



HOCHSCHULE RUHR WEST
UNIVERSITY OF APPLIED SCIENCES

Master Thesis

Supporting chatbot intent recognition with sentiment analysis

Master of Science in Computer Science at the Ruhr West
University of Applied Sciences

Mohamed Alaa Eddine Cherni

Supervision:

Prof. Dr. Anne Stockem-Novo

Prof. Dr. Fatih Gedikli

Bottrop, July 2022

Acknowledgment

I would like to express my sincere gratitude to my Directors of Memory, Prof. Dr. Anne Stockem-Novo and Prof. Dr. Fatih Gedikli. I thank them for supervising, guiding, assisting, and advising me. I address my sincere thanks to all the professors and persons who, through their words, writings, advice, and criticism, have guided my reflections and agreed to meet with me and answer my questions throughout my study. I thank my wonderful parents, who have always been there for me.

Abstract

Developing an intelligent chatbot that can imitate human-to-human interaction has become important in recent years. For this reason, many studies have been conducted to evaluate the quality of chatbots. Furthermore, various approaches and tools, such as sentiment analysis, have been created to improve the performance of chatbots.

This study examines previous research to identify the quality dimensions used to measure chatbots performance in order to develop a general chatbot assessment model that evaluates and compares chatbots quality. The developed evaluation model measures ten chatbot quality dimensions. This model is based on user experience. It requires human testers to interact with the chatbot to test its functioning and then a quantitative approach is used to collect data from user testing by conducting a survey with these testers. In this survey, they are instructed to evaluate the quality of the chatbot using a questionnaire that contains the items needed to evaluate each dimension.

This study also investigates whether sentiment analysis can improve the quality of chatbots and, if so, to identify the dimensions improved with sentiment analysis. For this reason, two chatbot versions are implemented using the Rasa framework (one that cannot detect sentiment and the other that analyzes sentiment and responds accordingly).

Following that, we used our evaluation model to evaluate and compare the two chatbot versions with two groups of participants by conducting a survey. In this survey, each group tested the functioning of one version. Then, both groups were instructed to use the items of the evaluation model to evaluate the version they tested. The goal of this survey was to evaluate the validity and reliability of the items used in the evaluation model to evaluate chatbots, and also to determine if sentiment analysis improved the chatbot quality by comparing survey results between the two groups.

The results show that items used in the assessment model to evaluate chatbots are valid and reliable. The findings also indicate that sentiment analysis improves the chatbot's quality. However, it improves the quality of some dimensions but not the majority of them.

Table of Contents

Acknowledgment	2
Abstract	3
Table of Contents.....	4
List of Figures	8
List of Tables.....	9
Acronyms.....	10
1. Introduction	11
1.1. Motivation	12
1.2. Problem statement:	13
1.3. Outline.....	14
2. Background	15
2.1. Dialogue system	15
2.1.1. Components of dialogue system	15
2.1.2. The Input Decoder	16
2.1.3. Natural Language Understanding (NLU)	16
2.1.4. The Dialogue Manager	16
2.1.5. Domain-Specific Component.....	17
2.1.6. Response Generator	17
2.1.7. Speech Generation	17
2.2. Classification of dialogue system.....	17
2.2.1. Chatbots	17
2.2.1.1. Rule-based chatbots.....	18
2.2.1.2. Corpus-Based chatbots	18
2.2.1.3. Hybrid Models.....	19
2.2.2. Task-based dialogue systems/Task-oriented dialogue systems	19
2.3. Relevant dialogue systems	20
2.3.1. Meena	20
2.3.2. Replika.....	20
2.3.3. Mitsuku	20
2.4. Rasa Framework.....	21
2.4.1. Architecture	21
2.4.1.1. Rasa NLU	22
2.4.1.2. Rasa Core	26
2.4.1.3. Action Server.....	28
2.4.1.4. The input/output channels, Tracker store, Lock store, File system	28

Table of Contents

2.4.2. Workflow / dialogue management flow.....	29
2.4.3. File structure	30
2.4.4. Training data	31
2.4.4.1. NLU Module Data.....	31
2.4.4.2. RasaCore data.....	31
2.4.5. Rasa server/Rasa Open Source HTTP API	32
2.5. Sentiment analysis.....	33
2.5.1. Sentiment analysis algorithms	33
2.5.2. Vader sentiment	34
2.6. Flask	35
2.7. Ngrok.....	35
2.8. SQLite	35
2.9. Chatbots evaluation	36
2.9.1. The PARAdigm for DIalogue System Evaluation (PARADISE)	36
2.9.2. Commercial Chatbot: Performance Evaluation, Usability Metrics and Quality Standards of Embodied Conversational Agents	37
2.9.3. Perspectives for Evaluating Conversational AI	39
2.9.4. Evaluating Quality of Chatbots and Intelligent Conversational Agents	39
2.9.5. On Evaluating and Comparing Open Domain Dialog Systems	41
2.9.6. Evaluating and Informing the Design of Chatbots.....	42
2.9.7. A Survey on Evaluation Methods for Chatbots	43
2.9.8. Can we Improve the User Experience of Chatbots with Personalisation?	43
3. The evaluation Model.....	45
3.1. Evaluation model Overview	45
3.1.1. Evaluate the quality of chatbots	46
3.1.2. Comparing the quality of chatbots	46
3.2. Evaluation model dimensions	47
3.2.1. Usefulness.....	47
3.2.2. Ease-of-Use.....	48
3.2.3. Effectiveness.....	48
3.2.4. Efficiency.....	48
3.2.5. Visual appearance.....	49
3.2.6. Responses in unexpected situations	49
3.2.7. Personality and humanity.....	50
3.2.8. Response time	50
3.2.9. Security and privacy	50
3.2.10. User satisfaction.....	51
4. Implementation.....	54
4.1. General use case	54

Table of Contents

4.2. System architecture	55
4.2.1. Rasa framework	55
4.2.1.1. NLU module	55
4.2.1.2. Rasa Core	63
4.2.1.3. Training data.....	68
4.2.1.4. Rasa HTTP API.....	69
4.2.2. Front end client	69
4.2.3. Database:	70
4.3. Main tasks and conversational user/bot flow	70
4.3.1. Inform	71
4.3.2. Booking	74
4.3.3. Cancel flight.....	77
4.3.4. Cancel return flight	79
4.3.5. Modify travel date.....	80
4.3.6. Modify return date	81
4.3.7. Make claim	82
5. Research methodology	84
5.1. The reliability and validity	85
5.1.1. Reliability	85
5.1.2. Construct validity.....	85
5.1.2.1. Convergent validity	85
5.1.2.2. Discriminant validity	86
5.2. Chatbot quality improvement with sentiment analysis	86
5.3. Data collection	87
6. Survey results.....	88
6.1. Reliability and validity	88
6.1.1. Reliability	88
6.1.2. Validity	89
6.1.2.1. Convergent validity	89
6.1.2.2. Discriminant validity	90
6.2. Chatbot quality improvement with sentiment analysis	91
7. Discussion	97
7.1. Chatbot evaluation.....	98
7.2. Chatbot improvement with sentiment analysis	98
7.2.1. Chatbot overall quality improvement	98
7.2.2. Dimensions improvement	99
8. Conclusion and outlook.....	101
8.1. Limitations	101

Table of Contents

8.2. Future Research..... 102

References..... 103

Statutory Declaration 107

List of Figures

Figure 2.1: Components of a dialogue system	15
Figure 2.2: Rasa architecture	21
Figure 2.3: NLU components life cycle	23
Figure 2.4: Rasa dialogue flow	29
Figure 3.1: The dimensions of the evaluation model	47
Figure 4.1: General use case diagram.....	54
Figure 4.2: The system architecture	55
Figure 4.3: The pipeline steps	59
Figure 4.4: Neutral user sentiment	60
Figure 4.5: Negative user sentiment	60
Figure 4.6: Discussion without sentiment analyzer.....	61
Figure 4.7: Discussion with sentiment analyzer.....	61
Figure 4.8: Conversational user/bot flow for sentiment analysis	62
Figure 4.9: The policies	67
Figure 4.10: The user interface.....	69
Figure 4.11: Conversational user/bot flow for Inform process	71
Figure 4.12: Inform about a flight	72
Figure 4.13: Multiple information entered by the user in a single message.....	72
Figure 4.14: Booking confirmation	73
Figure 4.15: Stop the process	73
Figure 4.16: Modify data during the process.....	73
Figure 4.17: Bad sentiment detected during the inform process	73
Figure 4.18: Conversational user/bot flow for booking process	74
Figure 4.19: Start of payment process	75
Figure 4.20: Payment and confirmation	75
Figure 4.21: Conversational user/bot flow for cancel flight.....	77
Figure 4.22: Cancel flight.....	78
Figure 4.23: Cancel flight when a negative user mood is detected	78
Figure 4.24: Conversational user/bot flow for modify travel date process	80
Figure 4.25: Modify travel date	80
Figure 4.26: Modify travel date confirmation	80
Figure 4.27: Conversational user/bot flow for the claim process.....	82
Figure 4.28: Make a claim.....	83
Figure 4.29: Claim confirmation	83

List of Tables

Table 3.1: The items of the evaluation model	53
Table 4.1: Main Intents.....	56
Table 4.2: Main entities	57
Table 4.3: Main slots	59
Table 4.4: Main forms	65
Table 4.5: Main custom actions.....	67
Table 6.1: Cronbach Alpha, Composite Reliability, Average Variance Extracted	89
Table 6.2: HTMT ratios of all dimensions	90
Table 6.3: Survey Results of the first group.....	92
Table 6.4: Survey Results of the second group	93
Table 6.5: The P-value of the unpaired t-test of overall averages between the two groups	94
Table 6.6: : The P-value of the unpaired t-test of overall averages per dimension between both groups	95
Table 6.7: Hypotheses and results	95

Acronyms

CR: Composite Reliability

AVE: Average Variance Extracted

HTMT: Heterotrait-Monotrait ratio

TAM: Test Acceptance Model

HTTP: Hypertext Transfer Protocol

CSS: Cascading Style Sheets

JS: JavaScript

SQL: Structured Query Language

SDK: Software Development Kit

1. Introduction

A chatbot is a computer program that interacts with a human user using natural language [1]. The first chatbot was developed in 1960. It follows a set of rules to generate responses based on user input [2]. Nowadays, many chatbots and virtual assistants have been developed and are available commercially, such as Amazon Alexa, Google Assistant, Apple Siri, and others. These assistants can mainly answer users' questions and can perform a variety of activities such as setting alarm, scheduling meetings, online shopping, etc¹. Furthermore, several businesses have developed their own chatbots and embedded them into their websites to assist their customers. Because, these chatbots provide support seven days a week and twenty-four hours a day and can save the cost of hiring a real human assistant². For this reason, developing an intelligent and an accurate chatbot that can imitate human-to-human conversations has become an important task in recent years. Because, a poorly developed chatbot, that cannot understand users and give them responses that they didn't expect, can easily lead to dissatisfaction and ruin the user experience [3].

Since the usage of chatbots has grown in recent years, many studies have been conducted to evaluate their quality, which permits to identify and fix gaps in order to enhance the performance of the chatbot. Furthermore, other studies have been conducted to evaluate and to compare the quality of different chatbots or versions of the same chatbot in order to determine the best among them. In fact, a high-quality chatbot may improve the user experience, but a badly designed one might lead to user's dissatisfaction. However, evaluating and comparing chatbot systems in terms of accuracy, efficiency, and the ability to satisfy users remains challenging [4].

Furthermore, several techniques and tools, such as sentiment analysis³ or grammar and spelling correction [5], have been developed to increase the quality of the dialogue between users and chatbots.

Sentiment analysis or Opinion mining is one of the most important tools used to improve the quality of chatbots. It is an approach used to detect a user's sentiment based on his

¹ <https://www.zdnet.com/home-and-office/smart-home/the-best-voice-assistant>

² <https://www.eatmy.news/2020/12/an-overview-of-chatbot-technology.html>

³ <https://www.revechat.com/blog/chatbot-sentiment-analysis/>

utterances. Sentiment analysis has grown in popularity in recent years, and it is currently used in a variety of fields and industries. Nowadays, it is primarily used in social media monitoring to determine users' opinions on specific topics and in business market monitoring to determine how users feel about a company's products or services. It is also integrated into many existing chatbots to enable them to understand human sentiments and respond accordingly⁴. The main objective of integrating sentiment analysis into chatbots is to give consumers the sense that they are speaking with a real person who understands their emotions. Because people are emotional beings, they expect the person with whom they are communicating to understand their feelings, which can enhance their experience and motivate them to use chatbots in the future⁵.

This thesis aims to develop a general evaluation model that covers several quality dimensions for evaluating and comparing chatbots' performance, and to investigate how sentiment analysis can improve the quality of chatbots.

1.1. Motivation

Despite their growing popularity, chatbots frequently face assessment and quality issues. Most existing studies aimed at evaluating the performance of chatbots focused on quality dimensions related to their research and the type of chatbot they were evaluating, rather than proposing a general evaluation model that is applicable to all types of chatbots. Nevertheless, many studies have been conducted to develop a general model to evaluate other types of software applications such as web apps (e.g WebQual [6]).

Sentiment analysis, is considered as a significant chatbot improvement tool since it enables the chatbot to comprehend user sentiment and react accordingly. Thus giving the user the sense that they are chatting with a real person which can enhance their satisfaction.

This study has two main goals. First, it focuses on developing and evaluating a multidimensional evaluation model that can be used to evaluate and compare the performance of chatbots. The second goal of this study is to investigate whether sentiment

⁴ <https://monkeylearn.com/sentiment-analysis/>

⁵ <https://www.revechat.com/blog/chatbot-sentiment-analysis/>

analysis can improve the quality of chatbots and, if so, to identify which quality dimensions are improved by sentiment analysis.

1.2. Problem statement:

In the previous section, we discussed the lack of previous research focusing on evaluating chatbots' performance, as well as the importance of sentiment analysis as a tool for chatbots enhancement. The purpose of this thesis is to first create a general model for evaluating and comparing chatbots' quality, and after that to investigate if sentiment analysis can enhance chatbots' quality. To reach the desired objectives, we will investigate the following research questions:

Research question 1: What are the quality dimensions that can be used to evaluate chatbots?

The purpose behind answering this question is to identify the quality dimensions that permit the evaluation of the performance of chatbots. To accomplish this, we will refer to the literature review to investigate the quality attributes used in the previous studies to evaluate the quality of chatbots. Following that, we will develop a general evaluation model based on user experience that requires human testers and surveys to measure and compare the quality of chatbots.

Research question 2: Can sentiment analysis improve the chatbot quality?

The goal of answering this research question is to investigate if sentiment analysis can increase the chatbot's quality by identifying the quality dimensions improved with sentiment analysis. For this purpose, we will implement two versions of the same chatbot (The first version is incapable of understanding user sentiment, while the second is an improved version that can analyze sentiment and reply accordingly) using Rasa framework. Following that, we will utilize our assessment model to evaluate and to compare the quality between the two versions.

1.3. Outline

This document is structured as follows: In Chapter 2, we present the background knowledge and the literature review required for this work. Chapter 3 details the chatbot evaluation model. Chapter 4 describes the chatbot implementation. Chapter 5 details the research methodology. In chapter 6, we present the evaluation results. Discussion on the evaluation results and the methodology are presented in Chapter 7. Finally, Chapter 8 concludes the thesis, specifies the limitations of the study, and presents the future work.

2. Background

2.1. Dialogue system

A dialogue system is a computer program that interacts with a human user using natural language. The conversation system provides an interface between the user and a computer-based application that allows natural interaction with the application. The dialogue system could be voice based or text based, and it can be used in phones, PDAs, vehicles, robots, and web browsers, etc. Although different Dialogue Systems have varied architectures, they all possess the same set of phases: input recognition, natural language understanding, dialogue management, response generation, and output rendering [1].

2.1.1. Components of dialogue system

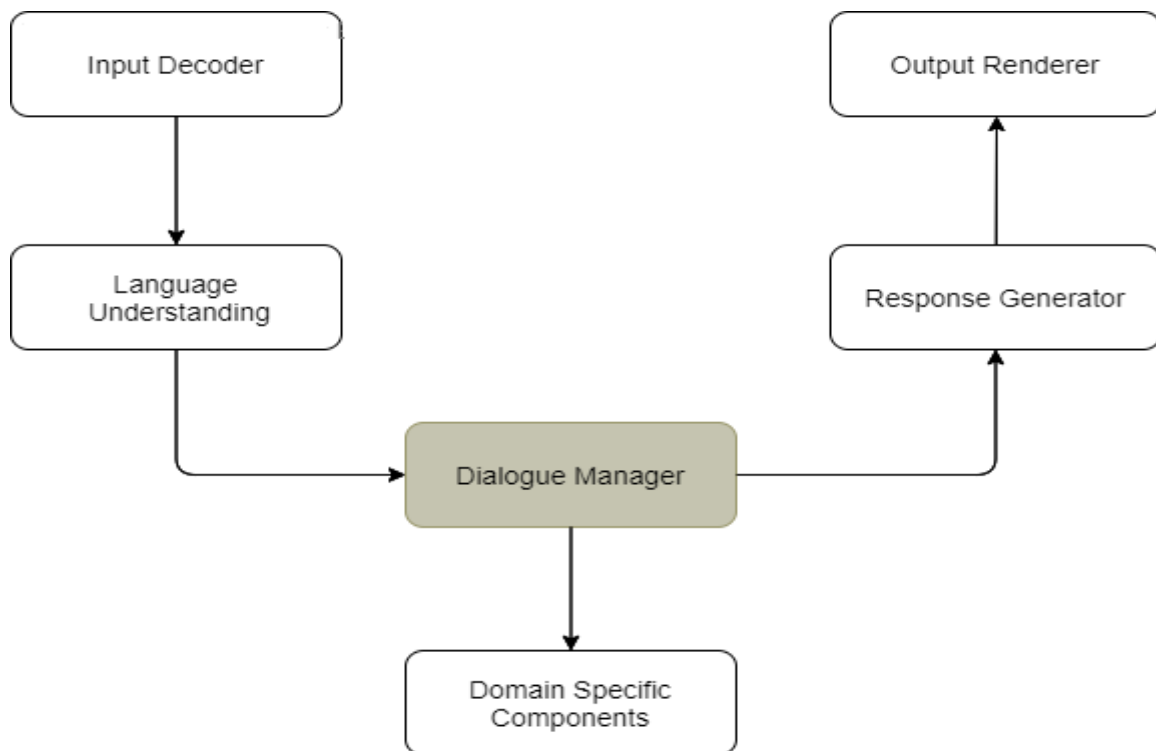


Figure 2.1: Components of a dialogue system

As shown in figure 2.1, there are seven key components to a dialogue system: Natural Language Understanding, Input Decoder, Domain-Specific Component, Dialogue Manager, Response Generator, and Output Renderer [1].

2.1.2. The Input Decoder

The Input Decoder is the component that detects the user input. It transforms the data into simple text. This component is only present in dialogue systems that aren't text-based. This component entails the translation of spoken sound (user utterances) to text (a string of words) [1].

2.1.3. Natural Language Understanding (NLU)

Natural Language Understanding (NLU) is a subfield of computer science that focuses on applying computational techniques to learn, understand, and produce human language content. NLU can be used for a variety of purposes, such as supporting human-human communication and improving communication between humans and machines. The extraction of structured, semantic information from unstructured natural language input, such as chat messages, is the overall purpose of NLU services. The two major pieces of information that NLU needs to extract when using a dialogue system are intents and entities. An intent is a mapping between what a user says and what action the dialogue system should perform. It represents the user intention of the entire message and is not restricted to a specific location within it. An entity, on the other hand, is a tool used to extract parameter values from natural language inputs. There is a corresponding entity for any important data that needs to be extracted from a user's message. The value of an entity is called a slot. For this message "What is the weather in Paris?" for example, the intent is "asking the weather," but "Paris" is a slot of a potential entity named "City" [7].

2.1.4. The Dialogue Manager

The Dialogue Manager is in charge of all parts of the conversation. It provides a semantic representation of the system response using a semantic representation of the user's input and determining how the text fits in the overall context. It accomplishes a variety of tasks, including: Maintains the discussion history, adopts certain dialogue approaches, deals with malformed and unrecognized text, retrieves information from files or databases, determines the best response for the user, manages initiative and system response, handles pragmatics issues, discourse analysis [1].

2.1.5. Domain-Specific Component

The Dialogue Manager needs to communicate with external software, such as a database or an expert system. As a result, the query or plans must be converted from the dialogue manager's internal representation to the format utilized by the external domain specific system (e.g. SQL). The domain-specific components are in charge of this interfacing. The Natural Language Query Processing system can handle this. From natural language, this system generates SQL queries [1].

2.1.6. Response Generator

This component is responsible for producing the response. It involves making decisions on what information should be included, how information should be arranged, word choice, and message syntactic structure. Simple methods, such as inserting retrieved data into specified slots in a template, are used by current systems [1].

2.1.7. Speech Generation

This component converts the message created by the response generation component into spoken language. There are two ways that can be utilized to generate speech. The first method is to employ prerecorded canned speech, which can be combined with retrieved or previously recorded samples, for example. "Welcome, how can I help you?" The second method is to employ text-to-speech synthesis. Text is used to generate speech in this case [1].

2.2. Classification of dialogue system

2.2.1. Chatbots

A chatbot, often known as a bot, is a computer program that simulates human conversation. Users interact with a chatbot using the chat interface or by voice, just as they would with a real person. Chatbots interpret the user's utterances and respond with a pre-defined response. They can be found on platforms such as Facebook Messenger, Whatsapp, Skype, Slack, Line, Kik, Wechat, and even in websites. Chatbots have an application layer, a database, APIs, and a User Interface. One of the benefits of chatbots

is that, unlike apps, they aren't downloaded, don't need to be updated, and don't take up memory on the phone. One of chatbots most appealing advantages is that, they are available 24 hours a day/seven days a week and a single chatbot may respond to several people at the same time⁶.

Chatbots can be classified into the following categories:

2.2.1.1. Rule-based chatbots

Rule-based chatbots use a set of rules to generate answers, e.g. if an input a is received, then perform action b and return response c. The Rule-based method consists of a pair of creating-pattern responses or templates. These templates may be created to handle a great range of inputs by using Natural Language Processing methods such as Semantic Role Labeling, Named Entity Recognition, and Part of Speech tagging, but it necessitates that the user enters entire sentences. The rule-based approaches may take more time than the other methods since they need the construction of many hand-written rules, but they may also be able to handle a broader range of topics as a result [8].

2.2.1.2. Corpus-Based chatbots

Corpus-based chatbots are AI-powered bots that combine the simplest features of Rule-based and Intellectually independent chatbots. Artificial Intelligence (AI) could be viewed as a computerized version of human intelligence. Artificial intelligence (AI) is a branch of computer science that focuses on developing intelligent machines that function and "think" like humans. AI-powered chatbots not only understand natural language, but also follow a predetermined path to ensure that they answer the user's problem. They can remember the conversation's context as well as the user's preferences. These chatbots can switch from one topic of conversation to another as needed, and they can respond to any user request at any time⁷.

AI chatbots are also divided into two categories:

⁶ <https://www.eatmy.news/2020/12/an-overview-of-chatbot-technology.html>

⁷ <https://www.eatmy.news/2020/12/an-overview-of-chatbot-technology.html>

Retrieval-based models:

A database of possible responses is required for retrieval-based models. This approach obtains the most relevant candidates from the database that match the current utterance, then selects the most appropriate response for retrieval [9].

Generative model:

Generative models use machine learning techniques to construct answers in real time. The model is used to produce responses by "translating" inputs into responses, and it is trained on a dataset of real dialogues. Some of the most current models for generating chatbot replies are Statistical Machine Translation (SMT) models [9].

2.2.1.3. Hybrid Models

Hybrid models are a combination of rule-based and deep learning-based design. They're created with the help of a collection of pre-defined rules and machine learning algorithms [10].

2.2.2. Task-based dialogue systems/Task-oriented dialogue systems

Task-based dialogue systems are AI chatbots that are more advanced than standard chatbots. Normal chatbots are utilized to respond to user questions in a question/answer process, whereas task oriented dialogue systems are meant to assist a user in completing a task, such as making an airplane reservation or purchasing a product. Every task-oriented conversation system is based around frames. A frame is a type of knowledge structure that represents the types of intentions that the system may extract from human sentences. It is composed of a collection of slots, each of which can take a set of possible values. This collection of frames is commonly referred to as a domain ontology. The set of slots in a task-based dialogue frame defines what the system needs to extract. For example, a slot in the travel domain could be of the type city, date, airline, etc. [10].

2.3. Relevant dialogue systems

2.3.1. Meena

Meena is a chatbot developed by Google. The model has 2.6 billion parameters and is an end-to-end trained neural conversational model. It was trained on 341GB of filtered social media conversations and employs a Transformer architecture known as The Evolved Transformer. Meena has 1.7 times higher model capacity and was trained on 8.5 times more data than the biggest GPT-2 model. The trained chatbot is considered as one of the most intelligent and specific existing chatbots. In comparison to other chatbots, the chatbot has amazing context understanding, although it still has limitations. Repetition and occasionally behaving as if the chatbot's prior response was from the user are two of its main weaknesses. Furthermore, while the work is centered on sensibleness and specificity, it does not take into account long-term memory or persona [8].

2.3.2. Replika

Replika is a chatbot that mixes neural generation and retrieval-based approaches. When producing responses, the neural generation takes into account the persona-based and the emotional embeddings. The retrieval-based model is trained using a hard negative mining technique, which forces the model to generate low matching scores for similar contexts and responses. This is done to avoid "echo-responses," which occur when a retrieval-based model retrieves the most semantically comparable response rather than the most suitable response. The system also takes into account conversation history and context by encoding it and providing it to retrieval and/or generating models. Furthermore, the agent can comment on and ask questions regarding images sent by the user [8].

2.3.3. Mitsuku

Mitsuku is a rule-based chatbot created with Artificial Intelligence Markup Language (AIML). Mitsuku has received the most Loebner prizes (5 times), where it was recognized as the most human-like chatbot in the competition. Mitsuku has a small memory where it keeps user information and certain contextual keywords. Some reported issues with the

chatbot include its repetitiveness, where it constantly utilizes the same template answer [8].

2.4. Rasa Framework

Rasa is a Python open source framework for chatbot development that uses machine learning approach. Rasa is designed to offer developers full control and customization over the development of their bots. Rasa creates all the baseline implementations, which facilitates developers work and allows them to concentrate on the domain-specific tasks. Rasa consists of two parts: Rasa NLU and Rasa Core. Rasa NLU is in charge of understanding a user's input, which can include intent classification and named entity recognition, slots extraction, etc. Rasa Core, on the other hand, manages the dialog flow using a neural network that predicts the next action depending on the current state. Alpacabot (Alex the Alpacabot: a virtual real estate agent 2021), Moltron (Moltron: educating users about machine learning 2021), Picpay (Connecting Brazilian Families with Emergency Government Assistance 2021), and others are examples of Rasa chatbots. The majority of these AI chatbots are text-based [11, 12, 13].

2.4.1. Architecture

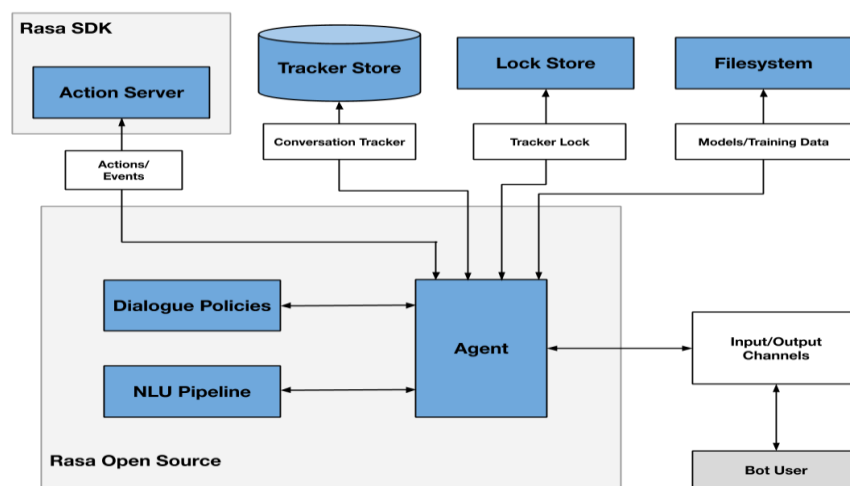


Figure 2.2: Rasa architecture⁸

⁸ <https://rasa.com/docs/rasa/arch-overview/>

The general architecture of the Rasa framework is seen in Figure 2.2.

2.4.1.1. Rasa NLU

Rasa NLU is the component in charge of NLU prediction, which can include intent classification and named entity recognition, slots extraction, etc. This is accomplished by providing training examples that indicate how the chatbot should understand user messages, and then a model is trained using these examples. These example are defined in the “nlu.yml” in the data folder of Rasa framework [8].

The three most important categories that Rasa NLU can recognize and extract are the following⁹:

The Intents: An intent represents the user intention of the entire message. All intents need to be defined in the “domain.yml” file of Rasa framework.

Intent example:

```
intents:
  - greet
```

The entities: An entity is a tool used to extract parameter values from natural language inputs. All entities needed are defined in the “domain.yml” file.

Example of entities:

```
entities:
  - PERSON
  - time
  - membership_type
  - priority
```

The Slots: Slots are quite important in Rasa. Because it functions as a key-value store, it can be used to store information provided by the user. They can support a variety of types, including text, boolean, float, list, category, custom, and unfeaturized. Slots can be filled not only from extracted entities (e.g. "Paris" is a slot of an entity named "City"), but also from intents, from text, or from a custom action. Filling the slots with intents allows to apply the filling regardless of the message's intent (e.g. a slot with a Boolean type can be

⁹ <https://rasa.com/docs/rasa/domain>

filled with true when an intent accept is detected and false when an intent deny is detected). The from-text filling will use the text of the most recent user utterance to fill the slot. The custom filling will use custom actions to fill the slot. All slots needed are defined in the “domain.yml” file. Using the keyword mapping, the type of filling (from intents, from entities, custom, or from text) is specified.

An example of slots:

```
slots:  
  cuisine:  
    type: text  
    mappings:  
      - type: from_entity  
        entity: cuisine
```

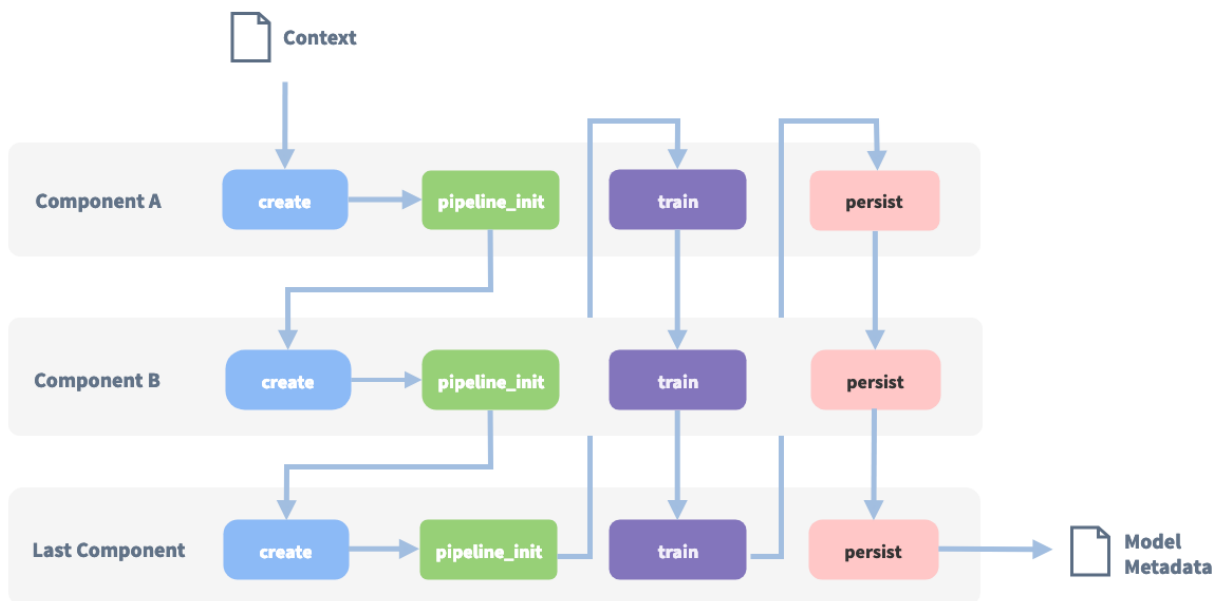


Figure 2.3: NLU components life cycle¹⁰

The NLU prediction is based on NLU components. These components are responsible of training the model for natural language understanding prediction, such as detecting user intent, extracting named entities and slots, and predicting sentiments [8].

¹⁰ <https://rasa.com/docs/rasa/tuning-your-model/>

The NLU module is implemented as a pipeline that processes the input text in a series of steps called as components. The lifecycle of Rasa NLU components is represented in Figure 2.3. A context object is passed to each component before the pipeline starts so that they can dissipate information. The output of one component can be utilized as the input of the next using this object. Each component of the pipeline is run in turn, and the output of each is available to the next. The Rasa SDK permits developers to implement their own components called Custom Graph Components or Custom Components. All pipeline steps are defined in the “config.yml” file of Rasa framework under the keyword “pipeline” [11].

- **Predefined NLU components:**

The following are some of the most important predefined NLU components¹¹:

The WhitespaceTokenizer: The WhitespaceTokenizer creates a token for every whitespace separated character sequence from the user sentence. Any character not found in the range a-zA-Z0-9_#& will be replaced with whitespace.

RegexFeaturizer: The RegexFeaturizer creates features for entity extraction and intent classification. It generates a list of regular expressions defined in the training data format during training. A feature will be set for each regex that indicates whether the expression was detected in the user message or not. To facilitate classification, all features will be fed into an intent classifier / entity extractor.

LexicalSyntacticFeaturizer: It provides lexical and syntactic features for a user message in order to assist entity extraction. It creates features for entity extraction and passes over each token in the user message with a sliding window, creating features according to the configuration.

CountVectorsFeaturizer: The CountVectorsFeaturizer creates intent classification and response selection features. It represents user messages, intents, and responses as a bag of words.

DIETClassifier: The DIET (Dual Intent and Entity Transformer) is a multi-task architecture for classifying intent and recognizing entities. The architecture is based on a transformer that serves both tasks.

¹¹<https://rasa.com/docs/rasa/components/>

EntitySynonymMapper: It Converts synonymous entity values to the same value. This component ensures that identified entity values are mapped to the same value if the training data contains defined synonyms.

Response Selector component: It's a dictionary with the key representing the response selector's retrieval intent and the value containing predicted responses, confidence, and the response key. It may be used to create a response retrieval model to predict a bot response straight from a list of possible responses. The dialogue manager uses the model's prediction to utter the predicted responses. It follows the exact same neural network architecture and optimization as the DIETClassifier and embeds user inputs and response labels in the same space.

The FallbackClassifier: If the intent classifier was unable to identify an intent with a confidence greater than or equal to the FallbackClassifier's threshold, the FallbackClassifier classifies the user message with the intent NLU fallback. It also creates a Fallback action that handles messages with uncertain NLU predictions.

- **Custom Graph Components:**

Rasa contains a variety of NLU components that can be used to train the NLU module. The Rasa framework permits developers to implement their own NLU components called Custom Component or Custom Graph Components in order to perform a specific NLU task for which Rasa does not have a pre-build component for example, a sentiment analyzer, or a word checker that corrects spelling errors, etc. The Custom Component is implemented as a Python class that contains all of the necessary methods to train the NLU module to perform that NLU specific task. The file containing the Custom Component class and the name of the Custom Component class must be referenced using the format “<file_name>.<CustomComponentClass_name>” under the keyword “pipeline” of the “config.yml” file of Rasa framework. There are two sorts of Custom Components: The pre-trained Custom Components (for example, trained on different datasets and packaged as python libraries, pkl files, etc.) and the Custom Components that need to be trained using the Rasa NLU training data¹².

¹² <https://rasa.com/blog/enhancing-rasa-nlu-with-custom-components/>

2.4.1.2. Rasa Core

The dialog management is the responsibility of the Rasa Core. It keeps track of a conversation and decides how the chatbot should respond. Based on previous user inputs, it generates a probability model that decides a set of actions to perform, based on the “rules” and “stories” defined in the training data. Rasa Core can execute three type of actions: “Simple response”, “custom action”, or “form”. Internally, Rasa Core uses the concept of policies to specify how the next action is chosen. All policies are defined in the “config.yml” of Rasa framework under the keyword “policies” [8,11].

- **Type of actions:**

Simple Response: The responses are simple messages or utterances that the chatbot can employ to reply to user input. These responses are defined in the “domain.yml” of Rasa framework under the keyword “responses”. A response can be a simple string, a button, or an image. Each utterance has one or more candidate responses, from which the chatbot will choose one randomly. Responses are simple messages that the chatbot uses to reply to user intent, but custom actions must be utilized when more complex tasks are required [8,13].

Example of a simple response:

```
responses:
  utter_greet:
    - text: "Hi there!"
  utter_bye:
    - text: "See you!"
```

Form: The form is how Rasa gathers a set of information for a certain goal. All forms are defined in “domain.yml” file under the keyword “forms”. Each form contains a set of slots that represents the data that must be gathered. This set of slots must be defined directly under the name of the form declared in the “forms” section in “domain.yml”. Once the form has been activated, the chatbot will use the slots’ utterances defined in the “responses” section in “domain.yml” to ask the user and collect responses so that the form's slots can be filled. The form is automatically stopped after all of the slots have been filled [13].

Custom action: While responses are simple messages that the chatbot can use to respond to user input, custom actions are used when more complex tasks are required, such as testing if the information entered by the user matches certain conditions and returning a specific response, modifying slots values to influence next actions, calling an external API, or storing and retrieving data from an external database. The “action.py” file of Rasa open source contains all customs actions' code. Every custom action is built as a class with two main methods: "name" and "run." The "name" method always returns a string with the custom action's name. The "run" section, on the other hand, provides the code for the custom action's task, which may include accessing an external API, saving and retrieving data from an external database, and etc. [8,14].

A custom action with the following name “validate <form_name>” can also be used to test and validate a form's required slots. As a result, the chatbot will only accept slots values that match specific conditions. A class for the given custom action must be built in the “action.py” file. This class mainly includes the "name" method, which returns the name of the given custom action, as well as additional methods for testing and validating the form's required slots values. Each of these methods contains the code that permits to test if the slot value meets the validation requirements. These methods are named in the following format “Validate <SlotName>”, where the “<SlotName>” field containing the name of the slot to be tested and validated. All custom actions need to be declared in “domain.yml” under the “actions” keyword [8,13].

- **Policies:**

Policies specify the next action to take based on user input. They are all defined in the “config.yml” file. Multiple policies can be defined at the same time. In this situation, the policy with the greatest confidence score prevails. Policies are a mix of machine learning (such as Transformer Embedding Dialogue Policy, Memoization Policy) and rule-based (Rule Policy) policies [11,13].

Some of the most commonly utilized Rasa policies are as follows¹³:

The Transformer Embedding Dialogue (TED) Policy: It is a multi-task architecture that predicts next actions and recognizes entities. The architecture is composed of different transformer encoders that are shared between the two activities. A Conditional

¹³ <https://rasa.com/docs/rasa/policies/>

Random Field (CRF) is used to predict a sequence of entity labels. The dialogue transformer encoder output and the system action labels are embedded into a single semantic vector space for the next action prediction.

The RulePolicy: The RulePolicy is a policy that deals with conversation parts that have a fixed behavior (e.g. business logic). It predicts the next action based on rules provided in training data. All rules are defined in the “rules.yml” files in the training folder of Rasa framework. To handle cases when the policies cannot predict the next action with high confidence, the RulePolicy can be configured to run a default action and to revert back to the state of the conversation before the user message that caused the fallback. Thus, it will not influence the prediction of future actions

Memoization Policy: It checks if the current dialogue corresponds to the stories defined in “stories.yml” file in the training folder of Rasa framework. It predicts the next action with a confidence of 1.0 based on the matching stories in training data. If no matching conversations are found, the policy predicts None with a confidence level of 0.0.

2.4.1.3. Action Server

The Rasa Action Server permits to execute all custom actions implemented in “action.py” file of Rasa framework. the Action Server runs independent of the NLU Module and the Rasa Core. Because Rasa Core's only requirement is to be able to communicate with the Action Server via a standardized API, the Rasa SDK forces an object-oriented pattern to ensure compatibility with the Rasa Core. The Rasa Action Server is called every time Rasa Core predicts a custom action to be executed¹⁴.

2.4.1.4. The input/output channels, Tracker store, Lock store, File system

The input/output channels: They are the chatbot application's frontend (e.g. Web browser, Facebook Messenger, etc.) [13].

The Tracker store: The tracker is an object that saves information about the conversation state. The conversation tracker is saved in the tracker store. The tracker is stored in memory by default, but an external database can be used to store the tracker [13].

¹⁴ <https://rasa.com/docs/rasa/custom-actions/>

The Lock store: Rasa employs a ticket lock system to ensure that incoming messages for a given conversation ID are processed in the correct order, and conversations are locked while messages are being processed. This means that many Rasa servers can run as replicated services in parallel, and clients don't have to submit messages to the same node for a given conversation ID. Lock store is used to save conversation locks. Conversation locks are kept in memory by default¹⁵.

The File system: Trained models can be accessible through a file system, such as a local hard disk, an HTTP server, or an external cloud, where they are stored [13].

2.4.2. Workflow / dialogue management flow

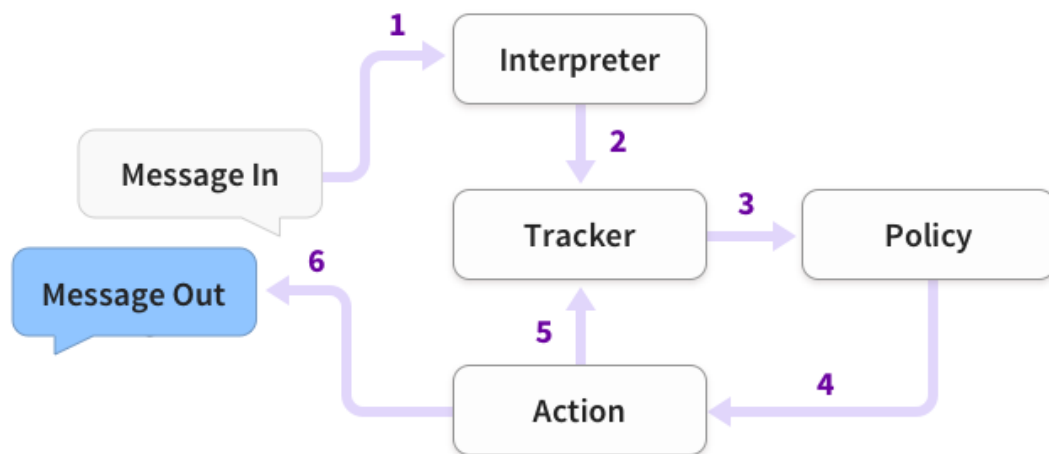


Figure 2.4: Rasa dialogue flow

The architecture or dialogue management flow consists of 6 steps, as shown in figure 2.4:

The received message is first delivered to the Interpreter, who converts it into a dictionary containing the original text, the purpose, and any entities recognized. This is handled by the NLU module. Second, the message is transmitted from the Interpreter to the Tracker, who keeps track of the state of the conversation. After that, each policy receives the current status of the tracker. Then, each policy determines the next action to take. Next, the tracker logs the chosen action. Finally, a response is sent to the user¹⁶.

¹⁵ <https://rasa.com/docs/rasa/lock-stores/>

¹⁶ <https://rasa.com/docs/rasa/next/architecture/>

2.4.3. File structure

After installing Rasa, the “rasa init” command can be used to start a project. After that, the following files and folders are created: domain.yml, config.yml, actions.py, credentials.yml, endpoints.yml and a data folder that contains the rules.yml, stories.yml and nlu.yml files [13].

The domain.yml file: It represents the chatbot's domain of knowledge. It contains all the names of all intents, entities, slots, custom actions, forms, and the name and text of simple responses or utterances that the chatbot may execute.

The config.yml file: It defines the Rasa NLU's “pipeline” and the Rasa Core's “policies”. The pipeline is where the NLU components are listed. These components are used to extract features such as intents and entities from the user input. Policies is where the Rasa Core policies are defined to predict the next action to take based on user input.

The action.py: The actions.py is where all custom actions classes are implemented. These custom actions are executed by the action server every time Rasa core predicts that a custom action is required.

The credentials.yml file: It includes elements related to messaging platform authentication, such as Slack, Facebook Messenger, and others.

The data folder: The data folder consists of three files: nlu.yml, rules.yml, and stories.yml. These three files contain the chatbot’s training data. nlu.yml lists example texts with entities if needed for each intent. Rules represent combinations of intent and actions. The order in which these collections' intent and actions are listed determines the order in which they are executed. Stories are collections similar to Rules, the only difference between them is that a story has a starting and an ending points.

The endpoints.yml file: It contains endpoint information. For example, if the models are stored on a cloud server, an endpoint is needed to access them. It specifies also where “tracker store” saves the conversation tracker, whether in the memory or in a SQL database.

2.4.4. Training data

To make NLU prediction and predict the next action, both modules NLU module and Rasa Core need to be trained.

2.4.4.1. NLU Module Data

To make NLU prediction, the Rasa NLU module must be trained using NLU training data. NLU training data consists of example user utterances categorized by intent. This generally includes any entities contained in his message. The `nlu.yml` file in the data folder of Rasa framework contains all of the NLU training data [11].

Example of NLU training data¹⁷:

```
- intent: greet
  examples: |
    - Hey
    - Hi
    - hey there [Sara] (name)
```

2.4.4.2. RasaCore data

The Rasa Core can be trained to predict next action using rules and stories, which are a sort of training data.

The Rules: Rules describe short pieces of conversations that should always follow the same path. It links a user's intent to one or more actions. An action could be a simple response, a custom action or a form. The rules can recognize intents or actions (response, custom action or form) only if they are defined in the `domain.yml` file. If the action used in a rule is a form, the chatbot will keep requesting the user to fill up all of the form's required slots. All necessary rules must be defined in the Rasa's `rules.yml` in the data folder [11].

Example of a simple Rule¹⁸:

```
rules:
```

¹⁷ <https://rasa.com/docs/rasa/training-data-format/#example>

¹⁸ <https://rasa.com/docs/rasa/rules/>

Background

```
- rule: Say `hello` whenever the user sends a message with intent `greet`  
  steps:  
    - intent: greet  
    - action: utter_greet
```

The Stories: Stories are a larger collection of Rasa rules that describe a conversation from beginning to end. Stories can be divided into smaller stories that can be related using checkpoint keywords. After training the chatbot with the “rasa train” command, the models generated employ stories to predict the next action. All necessary stories must be defined in the Rasa’s stories.yml in the data folder [13].

Example of a simple story¹⁹:

```
stories:  
- story: story to find a restaurant  
  steps:  
    - intent: find_restaurant  
    - action: restaurant_form  
    - action: utter_restaurant_found
```

2.4.5. Rasa server/Rasa Open Source HTTP API

The command line “rasa run” permits to run the chatbot as a server. Running this Rasa server permits to interact with the chatbot over webhook endpoints using chat applications such as Facebook Messenger, What’s up, Slack or Telegram. However, it does not enable the interaction with a front-end client like a web app or a mobile app via HTTP. The Rasa Open Source HTTP API allows a front-end client to communicate with a running Rasa server via the API HTTP endpoints. The command line “rasa run --enable-api” activates the HTTP API, allowing the Rasa server to respond the front end client HTTP requests²⁰.

¹⁹ <https://rasa.com/docs/rasa/rules/>

²⁰ <https://rasa.com/docs/rasa/http-api/>

2.5. Sentiment analysis

Sentiment analysis is a technique of natural language processing (NLP) for identifying positive, negative or neutral sentiment in texts. Sentiment analysis focuses on a text's polarity (positive, negative, or neutral), but it can also identify particular feelings and emotions (angry, happy, sad, etc.), urgency (urgent, not urgent), and intents (interested vs. not interested). Sentiment analysis is quickly becoming an essential tool for monitoring and understanding sentiment in all types of data, as humans express their emotions and opinions more openly than ever before. Nowadays, businesses frequently utilize it to identify sentiment in social data, analyze brand reputation, and comprehend customers. Using sentiment analysis, companies can learn what makes customers happy or upset by automatically analyzing customer feedback, such as thoughts in survey responses and social media conversations, to tailor products and services to match their customers' demands²¹.

2.5.1. Sentiment analysis algorithms

There are three main types of algorithms for implementing sentiment analysis, depending on the amount of data to analyze and the accuracy of the model²².

Rule-based: A rule-based system employs a set of manually crafted rules to analyze sentiment. Various NLP techniques developed in computational linguistics may be included in these rules, such as stemming, tokenization, parsing, and part-of-speech tagging, and Lexicon (i.e. lists of words and expressions).

Automatic: Unlike rule-based systems, automatic approaches depend on machine learning techniques rather than manually constructed rules. A sentiment analysis task is modeled as a classification problem, in which a classifier is given a text and outputs a category, such as positive, negative, or neutral. To implement a sentiment classifier using the machine learning technique, two processes are required: training and prediction. Based on the test samples used for training, the model learns to associate a specific input

²¹ [https://monkeylearn.com/sentiment-analysis/#:~:text=Sentiment%20analysis%20\(or%20opinion%20mining,feedback%2C%20and%20undersand%20customer%20needs.](https://monkeylearn.com/sentiment-analysis/#:~:text=Sentiment%20analysis%20(or%20opinion%20mining,feedback%2C%20and%20undersand%20customer%20needs.)

²² [https://monkeylearn.com/sentiment-analysis/#:~:text=Sentiment%20analysis%20\(or%20opinion%20mining,feedback%2C%20and%20undersand%20customer%20needs.](https://monkeylearn.com/sentiment-analysis/#:~:text=Sentiment%20analysis%20(or%20opinion%20mining,feedback%2C%20and%20undersand%20customer%20needs.)

(i.e. a text) with the corresponding output (tag) during the training process. The text input is converted into a feature vector by the feature extractor. To generate a model, feature vectors and tags (such as positive, negative, or neutral) are fed into the machine learning algorithm. In the prediction process, the feature extractor is utilized to convert unseen text inputs into feature vectors. The model then uses these feature vectors to generate predicted tags such as positive, negative, or neutral.

Hybrid: Hybrid system combine both rule-based and automatic techniques into a single system. One of the major advantages of these methods is that the results are frequently more accurate.

2.5.2. Vader sentiment

VADER stands for "Valence Aware Dictionary and sEntiment Reasoner" and is open source. The tool was released in 2014 and focuses mostly on social media messages. It rates the input using a lexicon-driven approach along with additional heuristics. VADER provides consistent ratings and does not require any training data because it is rule based approach. It's included in the NLTK package and can be used on unlabeled text data. Its development was divided into 7 phases: Gather lexical features of established sentiment lexicons, gather lexical features characteristic for microblogging domains, Rate lexical feature candidates, Filtering, building human heuristics, Evaluate heuristics, and evaluation and results. VADER sentiment analysis is based on a lexicon that contains more than 7500 words. This lexicon maps lexical features to emotion intensities called sentiment scores. The VADER sentiment analysis produces a sentiment score or emotion intensity in the range -4 to +4, with -4 being the most negative and 4 being the most positive. The midpoint 0 indicates a neutral feeling. A text's sentiment score is computed by summing the intensity of each word in the text. In Python programming language, a normalization is applied to the total to map it between -1 and +1²³. VADER performed exceptionally well in a variety of domains, including tweets, movie reviews, and product reviews [15].

²³ <https://medium.com/@piocalderson/vader-sentiment-analysis-explained-f1c4f9101cd9>

2.6. Flask

Flask is a web framework written in Python that facilitates the development of web applications. Flask is typically referred to as a micro framework since it lacks functionality such as an ORM (Object Relational Manager). It is intended to keep the application's core simple and scalable. It has a lot of features, such as URL routing and a template engine²⁴.

2.7. Ngrok

Ngrok is a globally distributed reverse proxy that serves web services from any cloud, private network, or private workstation. Ngrok is the quickest way to put an app on the internet. It allows to test apps against a development backend as well as build webhook consumers and demo websites without needing to deploy them. It requires no setup and gets started with a single command. It makes it simple to connect to networks because no port forwarding, dynamic DNS, or VPN are required²⁵.

2.8. SQLite

SQLite is an in-process library that creates a transactional SQL database engine that is self-contained, serverless, and requires no configuration. SQLite's code is in the public domain, which means it can be used for any purpose, commercial or private. SQLite is the most widely used database on the world, with an uncountable number of applications, including some high-profile projects. SQLite does not have a separate server process. It reads and writes to regular disk files directly. The database file format is cross-platform, allowing to copy databases between 32-bit and 64-bit platforms easily²⁶.

²⁴ <https://pythonbasics.org/what-is-flask-python/>

²⁵ <https://ngrok.com/>

²⁶ <https://www.sqlite.org/about.html>

2.9. Chatbots evaluation

Articles and research papers are examined in this section to provide a comprehensive review of existing chatbot evaluation metrics. These papers are discovered mostly through the keywords: chatbot, chatbot evaluation, evaluation framework, evaluation metrics, quality, performance, and their combinations. These Papers were selected if they contained at least one of the mentioned keywords in their title or abstract, and if they discussed some element of chatbot quality. Following the refinement, we selected eight papers that were rated most relevant. Some of these publications are based on the combinations of previous studies, while others are general views on evaluation aspects and their applicability to a research. A summary of the contents of these papers is described below.

2.9.1. The PARAdigm for DIAlogue System Evaluation (PARADISE)

The PARAdigm for DIAlogue System Evaluation (PARADISE) framework [16] is one of the oldest frameworks for evaluating chatbots. It was developed by Marilyn et al. in 1997. The framework decouples task requirements from an agent's dialogue behaviors, supports for comparisons of dialogue strategies, calculates performance over sub dialogues and entire dialogues, specifies the relative contribution of various factors to performance, and allows the comparisons of agents performing different tasks by normalizing for task complexity.

PARADISE supports comparisons among dialogue strategies by offering a task representation that decouples what an agent has to do in terms of task requirements from how the agent carries out the task via dialogue. PARADISE employs a decision-theoretic framework to specify the relative contribution of multiple factors in an agent's performance overall. Performance is represented as a weighted function of a task-based success measure and dialogue-based cost measurements, with weights calculated by correlating user satisfaction with performance. Performance may also be calculated for sub dialogues as well as entire conversations. PARADISE combines a heterogeneous set of performance measures (i.e., user satisfaction, task success, and conversation cost) into a single performance evaluation function using decision theory methodologies. The use of decision theory necessitates the specification of both the decision problem's objectives

and a set of measures (known as attributes in decision theory) for operationalizing the objectives. Other novel components of PARADISE include the use of the Kappa coefficient [17] to operationalize task success and the use of linear regression to quantify the relative impact of the success and cost factors to user satisfaction. This Kappa coefficient is calculated from confusion matrix that summarizes how effectively an agent fulfills the information requirements of a specific task for a set of dialogues instantiating a set of scenarios.

2.9.2. Commercial Chatbot: Performance Evaluation, Usability Metrics and Quality Standards of Embodied Conversational Agents

In this research paper published in 2015, Kuligowska [18] developed eleven aspects to evaluate the performance of commercial virtual assistants in the B2C industry. Kuligowska used a standard measurement tool with a rating scale of 1 to 5, assigning ratings of 1-very poor, 2-poor, 3-satisfactory, 4-good, and 5-very good to each of the 11 dimensions. Finally, Kuligowska generated a simple average of all the evaluated dimensions, providing an overview of the chatbot's overall quality. She employed this approach to evaluate and compare the performance of seven commercial chatbots. According to her, the chatbot with the highest overall rating performed the best.

Kuligowska used the following quality dimensions to evaluate chatbots:

The visual look: According to kuligowska, the book is frequently evaluated by its cover. As a result, the outer appearance of a virtual assistant is an important factor that determines the quality of its implementation.

The form of implementation on the website: This aspect evaluates the visibility of a virtual assistant embedded on a website.

The text-to-speech unit: According to Kuligowska, the Text-To-Speech module, that converts written text into synthetic speech, can boost user trust. As a result, it is one of the quality components that must be considered while evaluating chatbots.

The knowledge base (basic knowledge): This aspect focuses on measuring the chatbot ability to respond simple questions such as its name, the current time, etc.

The knowledge base (specialized knowledge): This aspect measures the chatbot built-in specialized knowledge such as the products and services offered, company contact information, and advanced knowledge about the firm.

The presentation of additional knowledge and functionality: This quality aspect focus on the additional functionalities performed by chatbots to facilitate user navigation on the website (e.g., "Back" button or scrolling the chat history, Term "Help" or "Info" button, etc).

Conversational skills and context-sensitivity: This aspect measures the chatbot's ability to lead a coherent conversation, handle complex user input, and take control of the conversation introducing topics.

Personality traits: According to Kuligowska, commercial chatbots must be equipped not only with skills, but also with the ability to express personality. It is essential to add a number of psychological layers to a virtual assistant's knowledge base, such as personality traits, biographical details, and expressed emotions.

Personalization options: Personalization options have a significant positive influence on consumers' judgment of the quality of interaction with a commercial chatbot. Users considered a conversational agent as more likable, trustworthy, and useful when they can customize its characteristics and appearance.

Emergency responses in unexpected situations: Commercial virtual assistants should be able to manage emergency situations such as understanding a user's unclear statement, detecting a lack of information on a specific question, dealing with insults, recognizing multiple languages, etc. Any typos, misspellings, or colloquialisms used in the dialogue should also be also recognized by the chatbot.

The ability to classify the chatbot and website by the user: User feedback on the chatbot is important for the chatbot's owner. Every chatbot must allow users to rate their overall satisfaction with the chatbot using a rating method such as a five-star rating, for example.

2.9.3. Perspectives for Evaluating Conversational AI

In their paper published in 2017, Jadeja and Varia [19] proposed an evaluation model that contain four perspectives to evaluate the chatbot's performance. The perspectives are the following:

The User Perspective: This perspective focuses on the measurement user satisfaction, usability, and other factors. According to them, recognizing the user's expectations, maintaining security and trust when private/confidential user data are required, and understanding user strategies may improve the following criteria. However, the fundamental drawback of this perspective is that it is time and money consuming.

The Information Retrieval Perspective: The measurement of the accuracy of information provided by the chatbot as well as the reaction time, or how quickly a user's input is processed, is the main goal of this perspective. According to them, the Information Retrieval Perspective is an important evaluation factor, but high IR qualities do not necessarily make users happy and satisfied, because user experiences are impacted by a variety of factors.

The Linguistic Perspective: It focusses on the measurement of four factors related to a chatbot's linguistic ability. First, the quality, which refers to how accurate the agent's phrases are. Second, the quantity of information, which evaluates how much information the bot provides. The third relation is to analyze how closely the responses are connected to the topic. Finally, the manner, which examines how direct and straightforward the conversation is in general.

the AI perspective: the measurement of the chatbot's human like interaction abilities is the main objective of this perspective. It can be made using the Turing Test. This perspective permits to improve problem solving and influencing skills of a chatbot.

2.9.4. Evaluating Quality of Chatbots and Intelligent Conversational Agents

Radziwill and Benton [20] reviewed 32 paper and 10 articles concerning chatbots evaluation methods in their research released in 2017, to identify the quality attributes of chatbots. After reviewing these papers and articles, they concluded that the evaluation methods are generally linked with the ISO 9214 notion of usability. The ISO 9214 defined

the usability as the effectiveness, efficiency, and satisfaction with which specific users achieve particular goal in specific environments. the effectiveness of chatbots is related to the accuracy and completeness with which users achieve their goals. efficiency is defined as how well resources are applied to achieve these goals. Satisfaction is the need to guarantee that customers are satisfied.

Based on this three categories, they defined an evaluation model that they employed to evaluate and compare the quality of two different versions on the same chatbot, to verify which of them is better. The evaluation model is the following:

The Efficiency: They specified five quality attributes that permits to measure a chatbot's efficiency: Graceful degradation, Robustness to manipulation, Robustness to unexpected input, Avoid inappropriate utterances and Effective function allocation

The Effectiveness: The effectiveness is divided into two sub-groups:

The functionality: It can be measured using quality attributes such as accurate speech synthesis, accurate command interpretation, linguistic accuracy, overall ease-of-use, and on-the-fly problem solving.

The humanity: It contains quality attributes such as passing (or failing) the Turing test, being transparent to inspection, including error for increased realism, and answering particular questions.

The user satisfaction: It is also divided into three categories:

The affect: It includes attributes like greetings and expressing personality, offering conversational cues, providing emotional information, demonstrating warmth and sincerity, making tasks more fun and engaging, and reading the mood.

The Ethics and behaviors: This attribute can be evaluated using the following quality attributes: Respect, preservation of dignity, users' ethics and cultural knowledge, privacy protection and respect, non-deception, sensitivity to safety and social concerns, trustworthiness, and awareness of trends and social context.

The accessibility: It includes quality attributes such as a response to social cues, intent recognition, and response to diverse needs.

2.9.5. On Evaluating and Comparing Open Domain Dialog Systems

In their 2018 study, Venkatesh et al. [21] proposed six different metrics for evaluating the open-domain conversational agents (socialbots) built for the Alexa Prize. They rejected the Turing tests in their study because they did not believe that this was a suitable method to evaluate chatbots, as AI chatbots may not behave as humans in some cases, but they can lead a good conversation, and also because the primary objective is to evaluate the conversational experience that a chatbot can provide without a human likely behavior.

The six metrics are the following:

The conversational user experience: They defined four elements based on user experience to evaluate the conversational user experience. First, the user expectation, which includes the chatbot's friendly presence and accuracy in responding to user input. Second, the chatbot's behavior and sentiments. third, there is security and trust. Finally, the measurement of how effectively the chatbot manages the absence of Visual Cues and Physicality.

The engagement: It measures the degree of interest in a conversation. In other words, how engaged a user is in the conversation.

The coherence: It evaluates the chatbot's ability to understand user messages and provide accurate answers.

The domain coverage: This metric measures the number of domain a chatbot can perform. A domain-specific conversation agent may be more like goal-directed chats, where the output answer space is constrained. An agent that can operate across multiple domains is more likely to be consistent with human expectations.

The conversational depth: It computes the average number of consecutive turns on a given thematic domain. It is critical to detect the context and depth of the discussions while evaluating chatbots. Human conversations typically delve deeper into a specific topic. An agent who can capture topical depth may seem more natural.

Topical diversity/conversational breadth: This metric evaluates a conversational agent's capacity to detect topics and keywords from a given utterance, to have conversations around the same topics and to share relevant concepts, and to recognize appropriate user intent.

2.9.6. Evaluating and Informing the Design of Chatbots

Jain et al. [22] focused on an HCI human-computer interaction perspective in their study published in 2018, to evaluate chatbots. To do this, they recruited a group of people with no prior experience with chatbots and instructed them to interact with chatbots for three days. During these three days, Jain et al. gathered the quantitative data listed below:

The total interaction time: It focuses on calculating how much time a user spends interacting with chatbots.

The message count: It measures the total of messages exchanged between a participant and the chatbot.

The interactive elements: It focuses on analyzing the amount of inerractive features provided by the chatbot during the conversation, such as buttons.

At the end of the three days, they asked participants via interviews to answer certain questions on their experience with the chatbots, from which they highlighted the most important users' comments regarding the chatbots. Based on these comments, they were able to define the following attributes that permit to evaluate chatbots performance:

The functionality: This element focuses on how well the chatbot performed in completing its primary task.

The conversational intelligence: It measures the chatbot's human-like interaction skills as well as its accuracy in understanding user input.

The chatbot personality: It focuses on examining the personality traits of a chatbot. According to Jain et al., participants preferred chatbots with distinct personalities. They expected the chatbot's personality to be consistent with its domain.

The chatbot interface: This attribute evaluates the interface that participants used to interact with the chatbots.

2.9.7. A Survey on Evaluation Methods for Chatbots

Maroengsit et al. [23] reviewed 30 publications representing chatbots from various areas (such as e-commerce, health, and open-domain), 17 of which included evaluation methods, in their paper published in 2019. They concentrated on evaluation methodologies and came up with three key categories:

The Content Evaluation: It covers both automatic evaluation using text summarization methods, or expert evaluation where humans are needed to perform what scripts cannot.

The User Satisfaction: This category focuses on methods that ask users about their thought of their interaction with the chatbot. It is also divided into Turn Evaluation and Session Evaluation based on whether users are asked about their interaction after each question or only at the finish.

Functional evaluation: It includes other approaches such as task-based evaluation, which is popular with goal-oriented chatbots, usage statistics, and evaluation as a building block.

2.9.8. Can we Improve the User Experience of Chatbots with Personalisation?

Duijst [24] developed an evaluation model to assess and compare several versions of the same chatbot in her study published in 2017. The assessment approach was based on user experience. After recruiting a group of users to test the functionality of the chatbot versions, she utilized a combination of quantitative and qualitative research methodologies to evaluate and compare the quality of the different chatbot versions from user testing.

Quantitative data: Surveys were conducted to gather quantitative data. After testing the chatbot's functioning, participants were invited to use a questionnaire to evaluate three chatbot's quality dimensions (usefulness, usability, and user satisfaction) by responding to the list of items related to each dimension using Likert scales. Finally, the survey responses were statistically analyzed in order to evaluate and compare the quality of different chatbot versions.

Qualitative data: Observations and interviews were used to collect qualitative data. During the testing of the chatbot's functionality, the participants were observed by the

researcher, allowing the researcher to take notes on the users' experiences. After testing the chatbot's functionality, the researcher conducted a semi-structured interview with the tester.

3. The evaluation Model

Almost all of the reviewed research that aims to evaluate and compare the performance and quality of chatbots did not provide a general framework that can be applied to all type of chatbots. However, they developed different evaluation approaches that concentrated on quality dimensions related to their study and the type of chatbot they were examining. Although these studies used different approaches to evaluate the performance of chatbots, the majority of them defined the user experience perspective as an important chatbot evaluation perspective. Because, chatbots are mainly used to communicate with humans. This user experience perspective necessarily requires human testers to interact with the chatbot. Then, multiple approaches are used to collect data from user testing in order to evaluate the chatbot quality, such as the quantitative approach through surveys and statistical analysis, the qualitative approach through interviews, or the combination of both quantitative and qualitative methods.

Because most reviewed studies considered the user experience as an essential chatbot evaluation perspective, we decided to develop a general evaluation model based on user experience to assess and compare the quality of chatbots.

3.1. Evaluation model Overview

This evaluation model includes ten chatbots quality dimensions: Usefulness, Ease-of-use, Efficiency, Effectiveness, User satisfaction, Personality and humanity, Responses in unexpected situations, Response time, Security and privacy, and Visual appearance. It requires first the use of human participants to test the functioning of the chatbot. Then, in order to evaluate the quality of chatbots, a quantitative approach is used to collect data from user testing by conducting a survey with these testers, in which they are asked to evaluate each dimension via a questionnaire. In this questionnaire, a set of linked items with a Likert scale from 1 to 5 (1-Strongly disagree, 2-Disagree, 3-Neutral, 4-Agree, and 5-Strongly agree) are used to assess the quality of every dimension.

The goal of this evaluation model is to help developers, researchers, and companies not only to evaluate chatbots but also to compare different chatbots and different versions of

the same chatbot, allowing them to have a general overview about the quality of chatbots they wish to evaluate or to compare.

3.1.1. Evaluate the quality of chatbots

After the survey is done, statistical analysis must be applied on the survey findings to calculate the overall average of the entire chatbot system as well as the overall average for each dimension. The overall quality of the chatbot or the quality of each dimension is considered very poor when the average is less than 2, poor when the average is between 2 and 3, satisfactory when the average value is between 3 and 4, and good when the overall is more than 4.

3.1.2. Comparing the quality of chatbots

To compare the quality of many versions of the same chatbot or different chatbots, our evaluation model must be applied to each one of them with the same number of testers to evaluate their performances. Then, the overall averages and dimension averages can be compared between chatbots. T-test [25] or ANOVA test [26] must also be conducted to measure whether the difference in averages between bots is significant or coincidental. When the number of chatbots to compare and the number of groups to test each of them is equal to two, the t-test is required. However, the ANOVA test is needed when the number of chatbots to compare and the number of groups to test each of them is greater than two.

3.2. Evaluation model dimensions

In this section, the dimensions of the evaluation model are described.

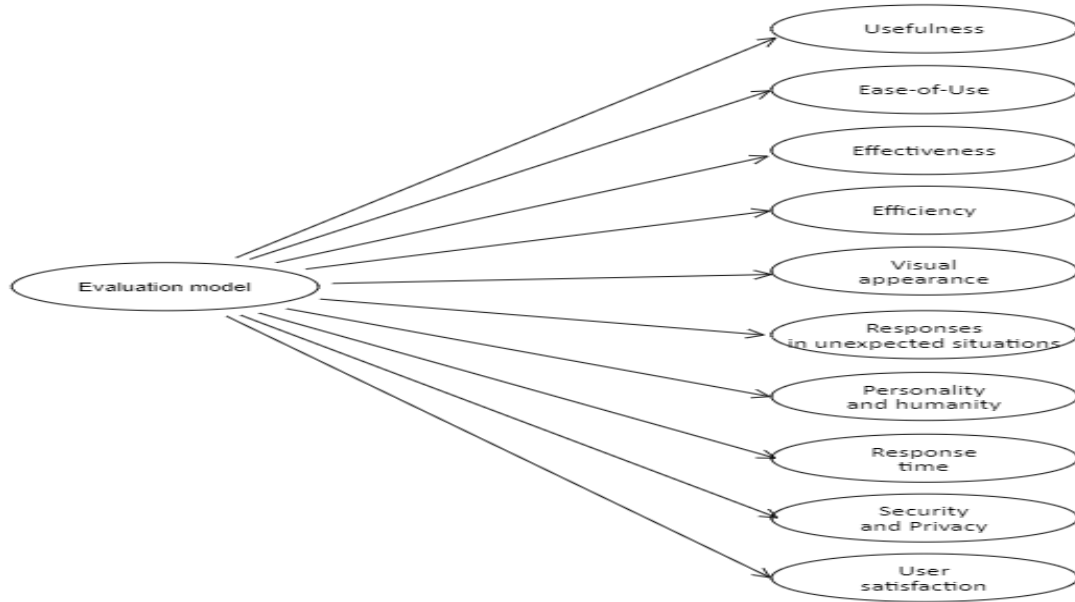


Figure 3.1: The dimensions of the evaluation model

3.2.1. Usefulness

The technology acceptance model (TAM) [27] developed by Davis in 1985 is one of the most well-known models of technology adoption. Perceived usefulness and perceived ease-of-use are the two major factors that determine technology acceptance, according to TAM. The extent to which a technology is considered to increase a potential user's performance is referred to as perceived usefulness. The usefulness is one of the dimension used by Duijst [24] to evaluate the quality of chatbots.

The TAM consists of ten items that can be used to measure a product's usefulness. To evaluate the usefulness of chatbots, we adapted from TAM the two items that are the most strongly related to our study.

Usefulness_1: Using this chatbot can help me complete tasks faster.

Usefulness_2: I find this chatbot useful.

3.2.2. Ease-of-Use

According to TAM [27], perceived ease-of-use is one of the two primary factors that affect technology acceptance. It is defined as the amount of effort necessary to effectively use a technology usefulness. The ease-of-use is one of the quality attributes that was defined by Radziwill and Benton [20] to evaluate chatbots.

TAM contains 10 items to measure a product's ease-of-use. We adapted the three items that are the most strongly related to our study from TAM to evaluate the Ease-of-use of chatbots.

Ease-of-Use_1: I find the chatbot easy to use.

Ease-of-Use_2: I find it easy to let the chatbot do what I want it to.

Ease-of-Use_3: Learning how to use the chatbot was easy for me.

3.2.3. Effectiveness

Radziwill and Benton [20] defined the effectiveness of chatbots as the accuracy and completeness with which users achieve their goals. We used the following items to evaluate the effectiveness of a chatbot.

Effectiveness_1: I felt that the chatbot understood all my intentions.

Effectiveness_2: I was able to reach my goal thanks to the chatbot.

Effectiveness_3: The chatbot understands exactly what I want and helped me to achieve my objective.

3.2.4. Efficiency

The efficiency refers to working as best as possible while wasting the least amount of time and effort [20]. We created the following items to evaluate a chatbot's efficiency.

Efficiency _1: The chatbot only provides me the amount of information I need.

Efficiency _2: The amount of information exchanged between me and the chatbot was adequate.

Efficiency _3: I reached my goal without too many exchanges.

3.2.5. Visual appearance

The visual look of a chatbot is an important factor that affects the quality standard of its implementation. Because, humans often judge a book by its cover [18].

We adapted the item used by Kuligowska [18] to evaluate the aesthetic look of a chatbot due to the importance of this factor.

Visual_appearance_1: I liked The visual look of the chatbot.

3.2.6. Responses in unexpected situations

According to Kuligowska [18], a chatbot should be able to respond to unexpected situations intelligently, politely, and patiently. Typos, misspellings, colloquialisms, and insults must all be recognized by a chatbot. It must also recognize the lack of information on a specific question and try a variety of creative solutions to overcome the user's ignorance.

The three first items used to measure chatbot responses in unexpected situations were adapted from those used by Kuligowska [18] to evaluate the chatbot responses in unexpected situations. We created the fourth item.

Responses in unexpected situations_1: The chatbot was able to recognize and overcome the lack of information on a specific question.

Responses in unexpected situations_2: The chatbot was able to overcome typos and misspellings.

Responses in unexpected situations_3: The chatbot was able to overcome insults and humiliations.

Responses in unexpected situations_4: The chatbot handled well the interruptions during the conversation.

3.2.7. Personality and humanity

According to Kuligowska [18], chatbots must be equipped not only with expertise, but also with the ability to express personality in order to become convincing in the eyes of users. The personality and humanity also were both defined in the evaluation model developed by Radziwill and Benton [20]. We developed three items to measure personality and humanity. The first item was used by Kuligowska [18] to evaluate personality in her research.

Personality and humanity_1: The chatbot has a very rich personality.

Personality and humanity_2: The chatbot was friendly.

Personality and humanity_3: I had the impression that I was chatting with a real person.

3.2.8. Response time

The response time is the amount of time it takes for a system to respond to a request or an interaction. In other words, the chatbot response time is the time that the chatbot takes to respond to the user's inputs. The response time, or how quickly a chatbot responds, was defined as an essential Information retrieval chatbot evaluation attribute in the research created by Jadeja and Varia [19].

We created the following items to evaluate the response time:

Response time_1: The chatbot responds quickly.

Response time_2: The response time was acceptable.

3.2.9. Security and privacy

According to Venkatesh et al. [21] and Jadeja and Varia [19] security is an important chatbot quality attribute. Because, security is vital for users especially when the chatbot is handling private or confidential data. Therefore, a higher level of trust can be built, if users believe that the chatbot is secure. Radziwill and Benton [20] defined the protection of privacy as an important quality attribute that can improve the user's satisfaction.

We created these two items to evaluate the security and privacy.

Security and Privacy _1: This chatbot was able to maintain my privacy.

Security and Privacy _2: The interaction with the chatbot seemed secure.

3.2.10. User satisfaction

Finally, the last dimension to evaluate in our evaluation model is the user's satisfaction or how the user feels about his experience with the chatbot. Most of the reviewed research defined the user satisfaction as one of the most important attribute to evaluate chatbots.

We adapted three items from the elements used by Duijst [24] to measure user's satisfaction in the questionnaire she developed.

User satisfaction_1: This chatbot is fun to use.

User satisfaction_2: I am satisfied with this chatbot.

User satisfaction_3: I would recommend this chatbot to a friend.

The dimensions and items of the proposed model are listed in the table below:

<i>Dimensions/Attributes</i>	<i>Items</i>
Usefulness	<ul style="list-style-type: none">• Usefulness_1: Using this chatbot can help me complete tasks faster.• Usefulness_2: I find this chatbot useful.
Ease-of-use	<ul style="list-style-type: none">• Ease-of-Use_1: I find the chatbot easy to use.• Ease-of-Use_2: I find it easy to let the chatbot do what I want it to.• Ease-of-Use_3: Learning how to use the chatbot was easy for me.
Effectiveness	<ul style="list-style-type: none">• Effectiveness_1: I felt that the chatbot understood all my intentions.• Effectiveness_2: I was able to reach my goal thanks to the chatbot.

	<ul style="list-style-type: none"> • Effectiveness_3: The chatbot understands exactly what I want and helped me to achieve my objective.
Efficiency	<ul style="list-style-type: none"> • Efficiency _1: The chatbot only provides me the amount of information I need. • Efficiency _2: The amount of information exchanged between me and the chatbot was adequate. • Efficiency _3: I reached my goal without too many exchanges.
Visual appearance	<ul style="list-style-type: none"> • Visual_appearance_1: I liked The visual look of the chatbot.
Responses in unexpected situations	<ul style="list-style-type: none"> • Responses in unexpected situations_1: The chatbot was able to recognize and overcome the lack of information on a specific question. • Responses in unexpected situations_2: The chatbot was able to overcome typos and misspellings. • Responses in unexpected situations_3: The chatbot was able to overcome insults and humiliations. • Responses in unexpected situations_4: The chatbot handled well the interruptions during the conversation.
Personality and humanity	<ul style="list-style-type: none"> • Personality and humanity_1: The chatbot has a very rich personality. • Personality and humanity_2: The chatbot was friendly.

	<ul style="list-style-type: none"> • Personality and humanity_3: I had the impression that I was chatting with a real person.
Response time	<ul style="list-style-type: none"> • Response time_1: The chatbot responds quickly. • Response time_2: The response time was acceptable.
Security and privacy	<ul style="list-style-type: none"> • Security and Privacy _1: This chatbot was able to maintain my privacy. • Security and Privacy_2: The interaction with the chatbot seemed secure.
User satisfaction	<ul style="list-style-type: none"> • User satisfaction_1: This chatbot is fun to use. • User satisfaction_2: I am satisfied with this chatbot. • User satisfaction_3: I would recommend this chatbot to a friend.

Table 3.1: The items of the evaluation model

4. Implementation

For our study, we developed a chatbot demo that allows flight company customers to purchase flight tickets and manage their reservations. Then, we created two versions of this chatbot to test if sentiment analysis can improve the chatbot quality. The first version cannot analyze the user sentiment, whereas the second is an improved version that can analyze the feelings and respond accordingly. Finally, we integrated these chatbots into two similar web applications that we implemented so that human testers could interact with both versions.

In this chapter, we will describe in details the implementation of the second version because both versions are extremely similar in terms of architecture and functioning, except that the first cannot analyze sentiment and respond accordingly, while the second can.

4.1. General use case

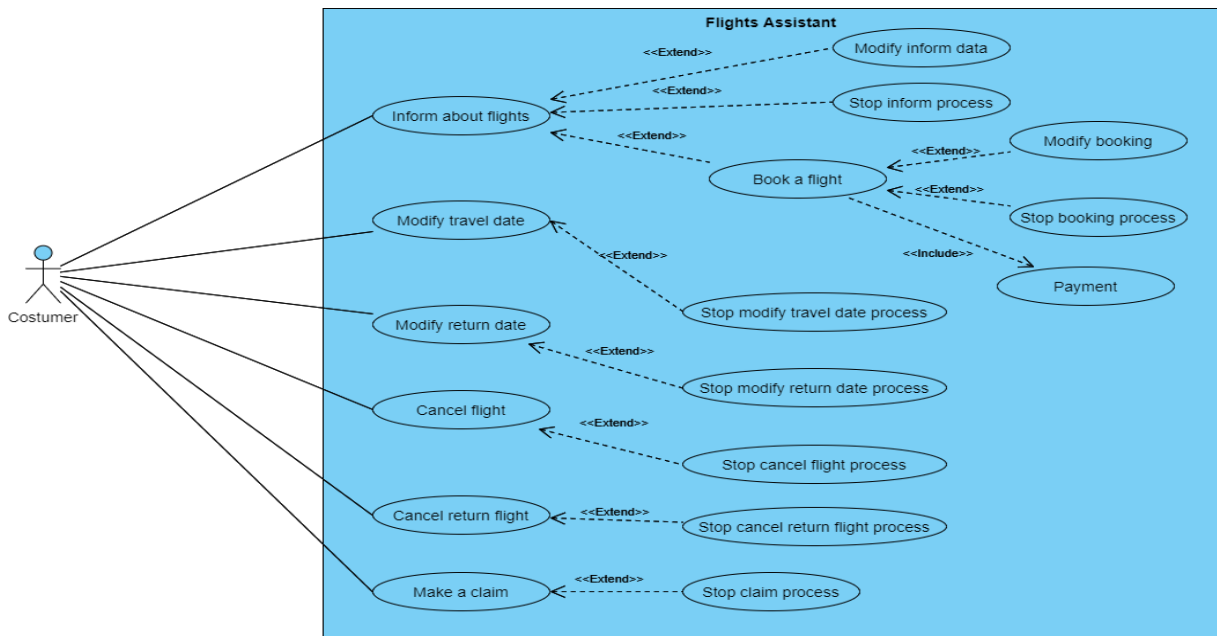


Figure 4.1: General use case diagram

4.2. System architecture

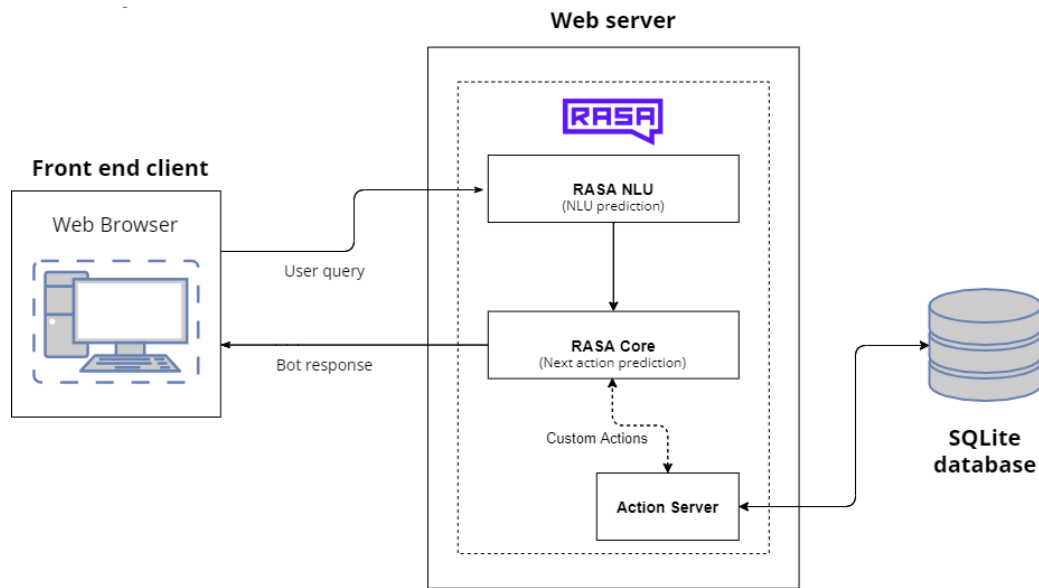


Figure 4.2: The system architecture

4.2.1. Rasa framework

The Rasa framework was used to develop the chatbot demo. The following sections cover all aspects of the demo's implementation.

4.2.1.1. NLU module

The NLU module is responsible for NLU prediction such as intents detection, entities recognition and slots extraction. All these intents, entities and slots are defined in the domain.yml file.

- **Data structure:**

The following are the most important Intents that we used in our implementation with the required entities/slots that need to be extracted when detecting these intents. Other intents, entities, and slots were also implemented, but they are not covered here for the sake of brevity.

Main Intents:

Intent	Description
inform_Book	This intent indicates that the user wants to be notified about potential flight opportunities. When this intent is recognized, the chatbot will provide all the necessary details and suggest the user book the flight if desired. If he confirms, the chatbot gathers his personal and his payment details in order to book this flight for him.
modify_travel_date	This intent indicates that the user wants to modify the travel date of a flight he has already booked.
modify_return_date	This intent indicates that the user wants to change the return date of a flight he has already booked.
cancel_flight	It means that the user wants to cancel his booking.
cancel_return_flight	It indicates that the user wants to cancel the return flight only.
Make_claim	Make_claim: This intent indicates that the user wishes to make a claim about something.

Table 4.1: Main Intents

Main Entities:

Entity	Description
city	City is the entity responsible for the extraction of the name of the city from user input.
date	This entity is responsible for the extraction of date from user messages.
sentiment	It is in charge of the extraction of user sentiment from his sentences.

Table 4.2: Main entities

Main slots:

Slot	Type	Mapping(extracted from)	Description
departure_city	Text	Entity: city	This slot stores the departure city entered by the user. It is filled when the entity city is detected.
arrival_city	Text	Entity: city	It saves the departure city sent by the user. It is filled from the city entity.
travel_date	Text	Entity: date	It stores the user travel date. It is filled when the entity date is recognized.
return_date	Text	Entity: date	It saves the user-specified return date. It is filled when the entity date in the user input is detected.
price	Text	Custom	This slot is automatically filled with the flight cost by a custom action after the user has already selected

			the travel date and return date.
passport_id	Text	From user text	It Is the slot that saves the user passport Id.
full_name	Text	From user text	It stores the user full name.
electronic_card_type	Text	From user text	It stores the user electronic card type.
electronic_card_number	Text	From user text	This slot saves the user electronic card number.
cvv_number	Text	From user text	It stores the user electronic card CVV number.
expiration_date	Text	Entity: date	This slot stores the user electronic card expiration date. It is filled when the entity date is recognized.
SMS	Text	From user text	This slot saves the secret code of the SMS sent to the electronic card owner in order to secure the payment.
ticket_id	Text	Custom/ From user text	<ul style="list-style-type: none"> • This slot is automatically filled by a custom action once the user purchases a ticket. • It is filled from user text when the user wishes to modify the travel date, modify the return date, cancel a flight , cancel the return flight, or make a claim.

claim_subject	Text	From user text	It stores the subject of the user's claim.
sentiment	Text	Entity: sentiment	This slot stores the user sentiment. It is filled from the entity sentiment.

Table 4.3: Main slots

- **NLU components:**

The NLU module is implemented as a pipeline that processes the input text in a series of steps called components. Each component of the pipeline is run in turn, and the output of each is available to the next. All pipeline steps are defined in the config.yml file.

The pipeline steps or components that we used in our implementation are defined in config.yml file as shown in figure 4.3.

```
# Configuration for Rasa NLU.
# https://rasa.com/docs/rasa/nlu/components/
language: en

pipeline:
- name: "sentiment.SentimentAnalyzer"
- name: WhitespaceTokenizer
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4
- name: DIETClassifier
  epochs: 100
  constrain_similarities: true
- name: EntitySynonymMapper
- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
- name: FallbackClassifier
  threshold: 0.6
  ambiguity_threshold: 0.1
```

Figure 4.3: The pipeline steps

The first component, "sentiment analyzer", is a Custom Graph Component, while the others are predefined by Rasa. All of the components we used are described below.

Sentiment analyzer:

We had to create a new Custom Graph Component that allows the NLU components to extract the user sentiment from his messages. Because, Rasa NLU does not provide a predefined NLU component that allows to analyze user sentiment. For this reason, we

first created a Python file called “sentiment.py”. In this file, we implemented a class named “Sentimentanalyzer” that includes the Custom Component's code. This class contains two primary methods: “process” and “covert_to_rasa”. The “process” analyzes the user sentiment using the Vadersentiment sentiment analyzer included in the “NLTK” library. This method returns a string that can be “neutral”, “negative”, or “positive”, as shown in figure 4.4 and 4.5 below. The “convert_to_rasa” method, on the other hand, indicates to the NLU module that the sentiment is extracted as an entity. Finally, we referenced this Custom Component in the NLU pipeline of the config.yml file. As a result, after the training process is complete, the NLU module will be trained to predict the user sentiment based on his input and to extract this sentiment as an entity.

```
"text": "Hey, I want to book for a flight",
"parse_data": {
  "intent": {
    "name": "book",
    "confidence": 0.99991774559021
  },
  "entities": [
    {
      "value": "neu",
      "confidence": 0.794,
      "entity": "sentiment",
      "extractor": "sentiment_extractor"
    }
  ]
}
```

Figure 4.4: Neutral user sentiment

```
"text": "Stop everything damn !!",
"parse_data": {
  "intent": {
    "name": "stop",
    "confidence": 0.9411519765853882
  },
  "entities": [
    {
      "value": "neg",
      "confidence": 0.733,
      "entity": "sentiment",
      "extractor": "sentiment_extractor"
    }
  ]
}
```

Figure 4.5: Negative user sentiment

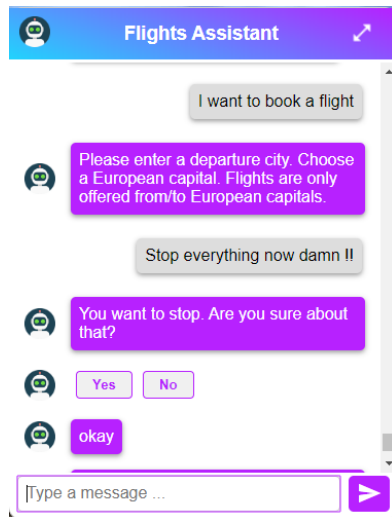


Figure 4.6: Discussion without sentiment analyzer

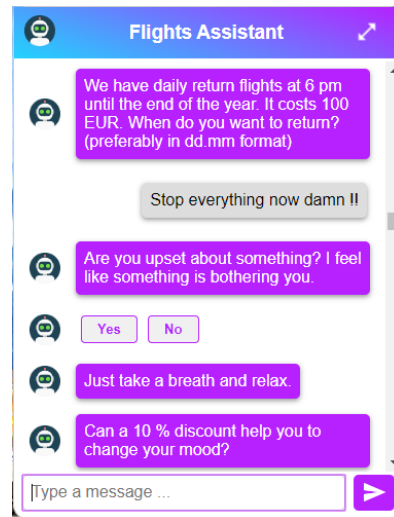


Figure 4.7: Discussion with sentiment analyzer

As seen in Figure 4.6, a chatbot without a sentiment analyzer will ignore the user's sentiment and continue with the process. On the other hand, if negative user sentiment is detected at any point throughout the conversation utilizing the sentiment analyzer, as illustrated in figure 4.7, the chatbot's answers will be more empathetic and the chatbot will take a new path as seen in figure 4.8. In this new path, it asks the user if he is unhappy about something. Then, it offers him a discount on booking or a discount code that can be used in free duty shops to try to change his mood.

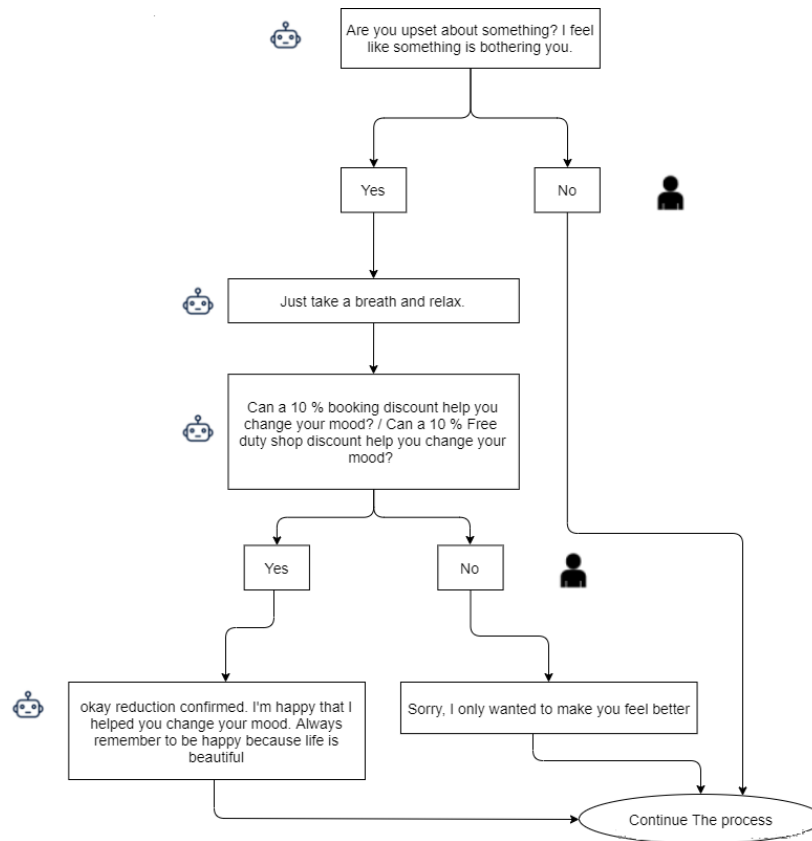


Figure 4.8: Conversational user/bot flow for sentiment analysis

Predefined components:

The predefined NLU components that we used in our implementation are:

- **WhitespaceTokenizer:** We used the WhitespaceTokenizer component to create a token for every whitespace separated character sequence from the user's sentence.
- **LexicalSyntacticFeaturizer:** We used this component to provide lexical and syntactic features for a user's message, in order to assist entity extraction.
- **RegexFeaturizer:** We used the RegexFeaturizer to create features for entity extraction and intent classification.
- **CountVectorsFeaturizer:** It was used to create intent classification and response selection features.
- **DIETClassifier:** We used the Dual Intent Entity Transformer (DIET) used for intent classification and entity extraction.

- **EntitySynonymMapper:** This component is used to ensure that identified entity values are mapped to the same value if the training data contains defined synonyms.
- **Response Selector component:** We utilized the Response Selector component for the creation of the dictionary with the key representing the response selector's retrieval intent and the value containing predicted responses, confidence, and the response key.
- **FallbackClassifier:** We used the FallbackClassifier to Classify a message with the intent nlu_fallback if the NLU intent classification scores are ambiguous. We set the threshold to 0.6 which means if the confidence on an intent is less 0.6, the chatbot will respond to the user that his message was unclear and that he needs to rewrite his message in a clear manner.

4.2.1.2. Rasa Core

Rasa core is the part of Rasa that is in charge of predicting and executing the next action based on the data retrieved by the Rasa NLU module. This action might be a simple response, a custom action, or a form. The most important forms and custom actions that we implemented are as follows:

- **Forms:**

A form contains a series of slots that represent the data that must be collected. Once activated, it will keep requesting the user to fill in all required slots using the list of slot utterances defined in the domain.yml until all are filled. Rasa forms are highly useful in operations such as booking that need the gathering of a set of data from the user.

The most important forms that we implemented are described below in more details:

Form	Description	Main Required slots
inform_book_form	This form is in charge of gathering the slots that need to be filled during the inform process and the booking process. Because,	<ul style="list-style-type: none">• departure_city• arrival_city• travel_date• return_date• price

	<p>every user must be first informed about the flights opportunities, the total price of the booking, and the payment opportunities. Then he will be asked if he is still interested about the offers and wants to book. When he confirms, the chatbot starts the booking process and asks the user to enter his personal data and the data related to the payment.</p>	<ul style="list-style-type: none"> • Passport_id • full_name • electronic_card_type • electronic_card_number • cvv_number • expiration_date • SMS
modify_travel_date_form	<p>This form is in responsible of gathering the slots that need to be filled during the modification of the travel date of an already booked flight.</p>	<ul style="list-style-type: none"> • travel_date • ticket_id
modify_return_date_form	<p>It is responsible for collecting the slots that need to be filled during the modification of the return date of an already booked flight.</p>	<ul style="list-style-type: none"> • return_date • ticket_id
cancel_flight_form	<p>This form gathers the slots that need to be filled during the cancel flight operation.</p>	<ul style="list-style-type: none"> • ticket_id
cancel_return_flight_form	<p>This form collects the slots that need to be filled during</p>	<ul style="list-style-type: none"> • ticket_id

	the cancel return flight operation.	
make_claim_form	It gathers the slots that need to be filled during the claim process.	<ul style="list-style-type: none"> • ticket_id • claim_subject

Table 4.4: Main forms

- **Custom actions:**

Custom actions are required when the next action is more complex than a simple action. All the custom actions are implemented in action.py file and they are executed by the Action Server. This Action Server runs independent of the NLU module and the Rasa core. It is called every time Rasa Core predicts a custom action to be executed.

The most important custom actions that we implemented are the following:

Custom action	Description
validate_booking_form	This is the custom action that was implemented to verify the values of all the required slots of the “inform_book_form” before validating them. If the value of the slot does not fit the constraints set in this custom action, the user will be asked to enter it again.
validate_modify_travel_date_form	It verifies all the required slots of the “modify_travel_date_form” and validates them only when they match certain constraints.
validate_modify_return_date_form	It checks all the required slots of the “modify_return_date_form” and validates them only when they match certain constraints.
validate_cancel_flight_form	It verifies all the required slots of the “cancel_flight_form” and validates them only when they match the constraints defined in this custom action.

validate_ cancel_return_flight_form	It checks all the required slots of the “cancel_return_flight_form” and validates them only when they match certain constraints.
validate_claim_form	It checks all the required slots of the “claim_form” and validates them only when they match certain conditions specified in this custom action.
submit_booking	This custom action is always executed once the booking operation is finished. It eventually generates the ticket Id, displays a booking confirmation with all booking details, and stores the booking in the database when the booking is confirmed by the user.
submit_modify_travel_date	This custom action is performed once the travel date modification process is over. It displays a confirmation message, and stores the modification in the database when the modification is confirmed by the user.
submit_modify_return_date	This custom action is executed once the return date modification operation is over. When the modification is confirmed by the user, it shows a confirmation message, and stores the modification in the database.
submit_cancel_flight	It is executed after the cancel flight process is over. Once confirmed by the user, it shows a confirmation message, and removes the booking from the database.
submit_cancel_return_flight	It is executed after the cancel return flight operation is finished. Once confirmed by the user, it shows a confirmation message, modifies the booking to a one- way flight and stores the modification in the database.

submit_claim	This custom action is executed once the claim operation is finished. When the claim is confirmed by the user, it shows a confirmation message, and stores the claim subject in the database.
---------------------	--

Table 4.5: Main custom actions

- **Policies:**

```
# Configuration for Rasa Core.
# https://rasa.com/docs/rasa/core/policies/

policies:

- name: MemoizationPolicy
- name: UnexpectTEDIntentPolicy
  max_history: 5
  epochs: 100
- name: TEDPolicy
  max_history: 5
  epochs: 100
  constrain_similarities: true
- name: RulePolicy
  core_fallback_threshold: 0.6
  core_fallback_action_name: "action_default_fallback"
  enable_fallback_prediction: True
```

Figure 4.9: The policies

Rasa Core uses the concept of policies to specify how the next action is chosen.

We used mainly the the RulePolicy to predict the next action based on the rules provided in training data. To handle cases when the policies cannot predict the next action with high confidence, we set the `core_fallback_threshold` to 0.6, as shown in figure 4.9. In that way, it sends a message to the user saying that his input was unclear and reverts back to the state of the conversation before the user's message that caused the fallback, when the next action confidence is less than 0.6.

We used also the Transformer Embedding Dialogue (TED) Policy to predict next actions and to recognize entities.

4.2.1.3. Training data

The data provided in the training data is used to train the NLU module and the Rasa core.

- **Rasa NLU Data:**

To make NLU prediction, the chatbot must train The Rasa NLU module using NLU training data. NLU training data consists of examples of user's utterances categorized by intent. This generally includes any entities contained in his message. The `nlu.yml` file, in the data folder that Rasa creates, contains all of the NLU training data.

To train the NLU module to make NLU prediction, we defined in `nlu.yml` file all intents that exists in the `domain.yml` with the possible user's utterance related to these intents. We defined also the possible entities that all these utterances could contain. Rasa uses a proprietary format for training data based on Markdown. We trained the NLU module using more than 1500 examples for different intents.

- **Rasa Core data:**

The Rasa Core can be trained to predict the next action using rules and stories. Rules describe short pieces of conversations that should always follow the same path. It links a user's intent to one or more actions. An action could be a simple response, a custom action or a form. The Rules can recognize intents or actions (response, custom action or form) only if they are defined in the `domain.yml` file. If the action used in a Rule is a form, the chatbot will keep requesting the user to fill up all of the form's required slots. All necessary rules must be defined in the `Rasa rules.yml` in the Data folder.

We used Rules to train our chatbot to predict the next actions. We utilized more than 150 rules that we defined in the `rules.yml`. We created rules that link all intents with all possible actions related to these intents. We employed these rules to manage all possible happy paths, user's interruptions and NLU Fallbacks.

4.2.1.4. Rasa HTTP API

After completing the implementation of the different modules of our demo and training both the NLU module and Rasa Core using the training data, we ran the chatbot using the command line “`rasa run –enable api`” to activate the Rasa HTTP API, allowing front end clients to communicate with the chatbot through HTTP.

4.2.2. Front end client

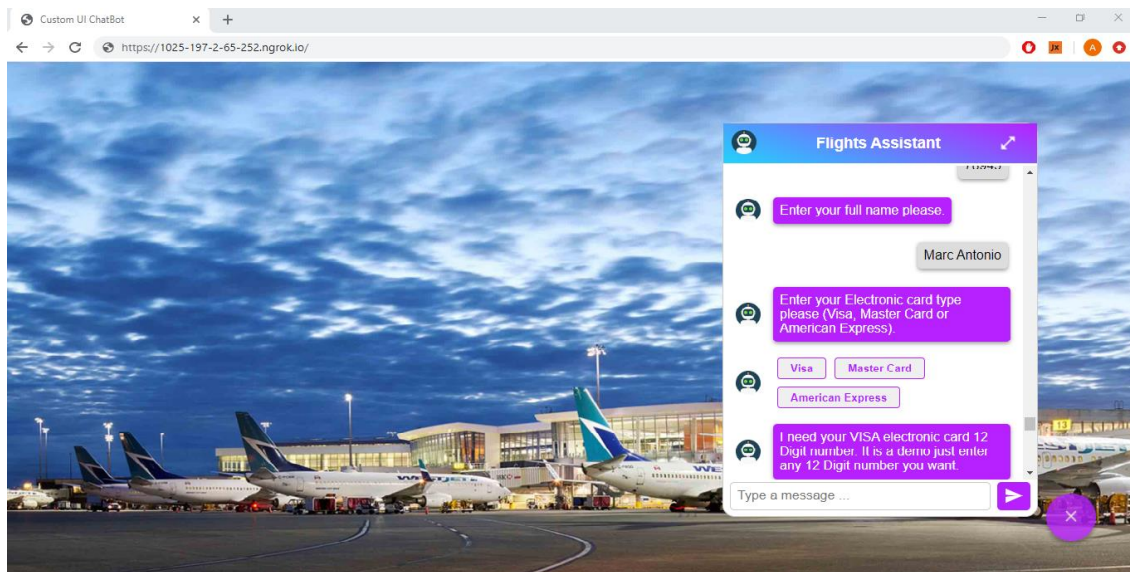


Figure 4.10: The user interface

The front end client uses the web browser to interact with the chatbot via HTTP. We utilized the Python web framework Flask to develop the necessary web services and to run the web server. This will allow users to interact with the chatbot via web browser using a web page named “`booking.html`”. We inserted the package `Rasa_Chatbot_UI`²⁷ in `booking.html`. This package consists of CSS and JS files. It creates a chat user interface and facilitates the interaction with a running Rasa server.

We executed Ngrok using the command line “`Ngrok<port where our web application is running>`” to create an URL address that can be used by users to remotely access our web application running on localhost to interact with the chatbot.

²⁷ <https://elysian01.github.io/Rasa-Chatbot-UI/>

4.2.3. Database:

For our implementation, we created a SQLite database named “bookings”. It contains a table named “Reservation” where all bookings are stored. Every line of this table contains all the information related to a booking such as the ticket Id, the customer full name, the passport Id, the electronic card type, the electronic card number, the departure city, the arrival city, the travel date, the return date, and the claim subject.

4.3. Main tasks and conversational user/bot flow

This section contains the description of every process presented in the use case. For each process, the main scenario and the corresponding conversational user/bot flow are explained in details.

4.3.1. Inform

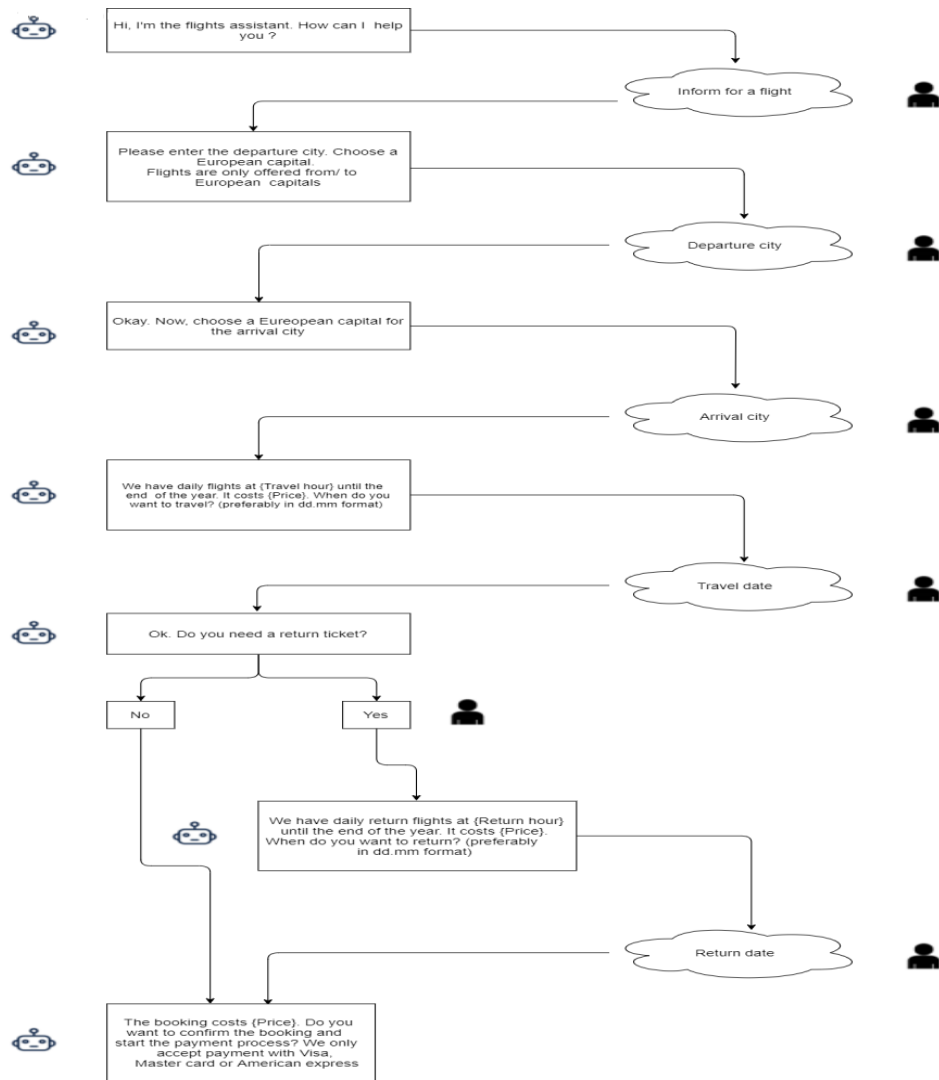


Figure 4.11: Conversational user/bot flow for Inform process

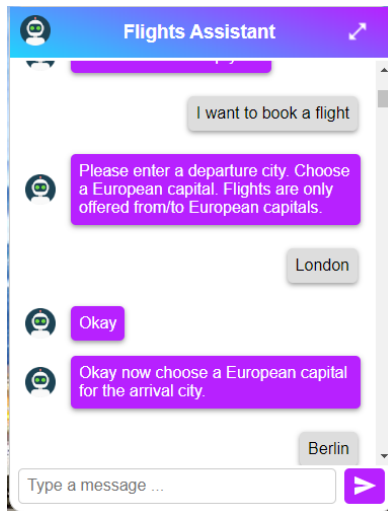


Figure 4.12: Inform about a flight

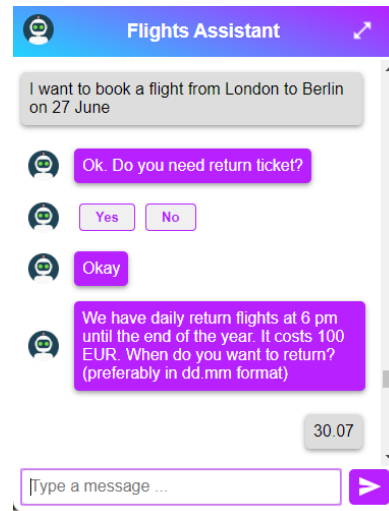


Figure 4.13: Multiple information entered by the user in a single message

The user may communicate with the chatbot to inform about potential flight possibilities, as seen in figure 4.12. At that point, the form “inform_book_form” is launched to gather all the information needed to fill the required slots related to the inform operation: `departure_city`, `arrival_city`, `travel_date`, and `return_date` if he desires to fly back.

As shown in figure 4.13, if the user provides a lot of information regarding many slots in a single sentence, the chatbot will fill these slots and then asks the user to enter data to fill the remaining slots.

When the user enters the value of a required slot, the custom action “validate_booking_form” is called to ensure that the value of that slot adheres to the restrictions defined in this custom action:

It validates the departure and arrival cities entered by the user only if they are both European capitals. The user may enter the travel and return dates in whatever format he wants, such as “tomorrow” or “21.05.2022”, etc. The methods responsible for the validation of the travel and the return date in the custom action “validate_booking_form” will convert them to the YYYY-MM-DD format using the Python “dateparser” package before validating their values. Only travel dates from the current year and at least one day after the current day are accepted. The return date is accepted only if it belongs to the current year and at least one day after the trip date.

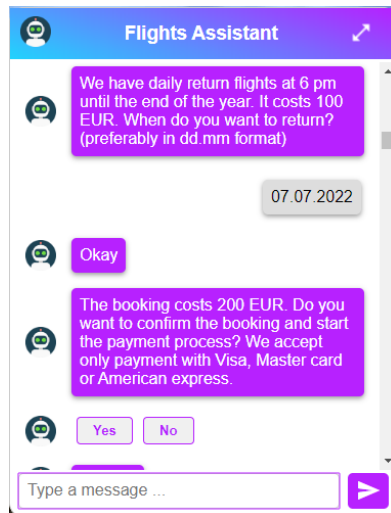


Figure 4.14: Booking confirmation

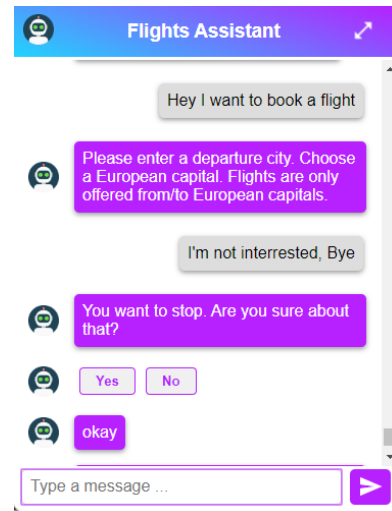


Figure 4.15: Stop the process

When all of these slots are filled, the chatbot displays the cost of the booking, payment options, and asks the user whether he wants to book that ticket and start the payment process, as seen in figure 4.14. If the user confirms, the booking process starts to collect user personal information and credit card details, and if he declines, the booking form is deactivated and the entire process is stopped.

As shown in figure 4.15, the user can also stop the process at any moment he wants. He can also modify all the data he previously entered, as seen in figure 4.16. If the user wants to modify the value of a previously filled slot, the chatbot will ask him to enter the data that allows him to fill that slot again instead of asking him to fill the next slot.

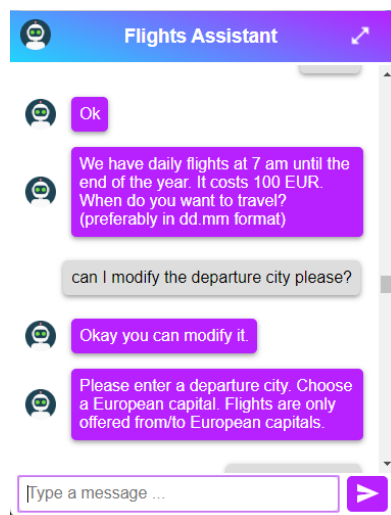


Figure 4.16: Modify data during the process

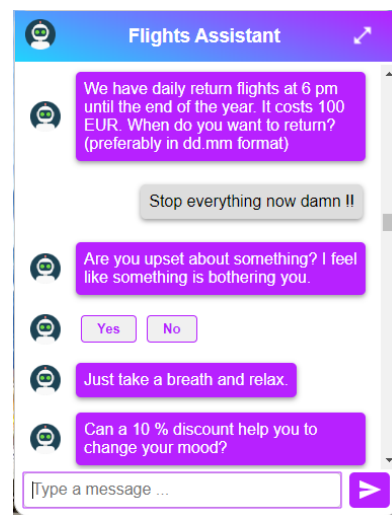


Figure 4.17: Bad sentiment detected during the inform process

As shown in figure 4.17, if negative user sentiment is detected at any point in the process, the chatbot's responses will be more empathetic, and the chatbot will not ask the user to provide the data required to fill the next slot, but will instead take a new path where it asks the user if he is unhappy about something and offers him a discount on booking to try to change his mood.

4.3.2. Booking

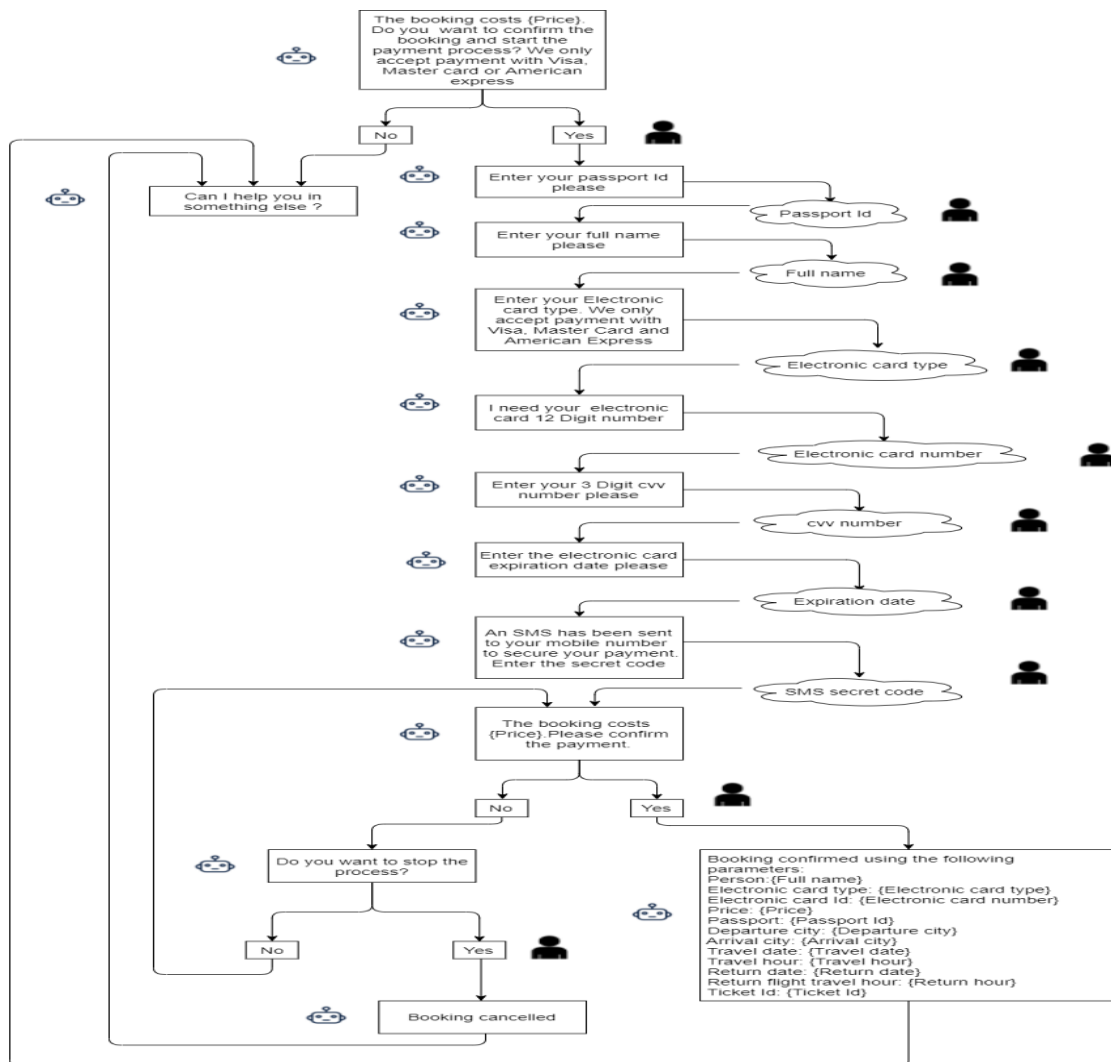


Figure 4.18: Conversational user/bot flow for booking process

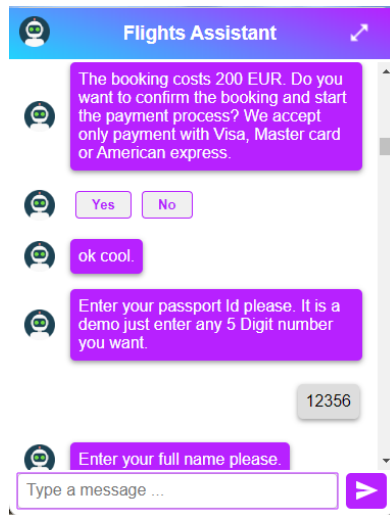


Figure 4.19: Start of payment process

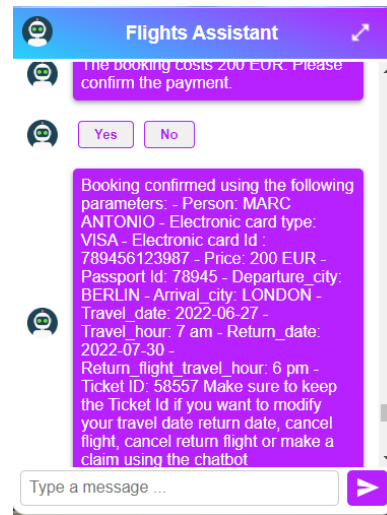


Figure 4.20: Payment and confirmation

When the user confirms that he wishes to purchase a plane ticket, the booking procedure starts to gather the user's personal information and payment information, as shown in figure 4.19. The “inform_book_form” will continue to ask the user to provide further information to fill the remaining slots: full_name, passport_id, electronic_card_number, cvv_number, expiration_date, and SMS. When a user enters the value related to a slot, the custom action “validate_booking_form” is called to ensure that the value corresponds to the constraints specified in this custom action.

The followings are the main steps of this process:

The procedure begins by requesting the user to provide his passport number in order to fill the slot “passport_id”. The custom action “validate_booking_form” is executed, when the user enters his passport number to check its value. The custom action validates the slot value only when the user’s text contains a five-digit number. otherwise, he will need to enter his passport number again.

Then, the user is asked to provide his name in order to fill the slot “full_name”. When the user enters his name, the custom action “validate_booking_form” is executed. It extracts the human name from a provided text using the Python “Spacy” package. If the user’s text is too short, or if the name entered by the user doesn’t exists in the human names dictionary of the Python “Spacy” package, the custom action will reject the slot value and the chatbot will ask the user to enter the name again.

After that, the user is asked to enter his electronic card number to fill the slot “electronic_card_id”. The custom action “validate_booking_form” is called, to check the

text entered by the user. The custom action validates the slot value only when the user text contains a twelve-digit number. Otherwise, he will need to enter his electronic card number again.

Next, the user is asked to enter his electronic card cvv number in order to fill the slot “cvv_number”. The custom action “validate_booking_form” is executed, to check the data entered by the user. This custom action validates the slot value only when the user’s input contains a three-digit number. otherwise, he will need to enter his cvv number again.

After the validation of the ccv number, the user is asked to enter the electronic card expiration date in order to fill the slot “expiration_date”. The custom action “validate_booking_form” is called, to check the date provided by the user. The expiration date is only accepted if the date is at least one day after the current day. Otherwise, he will need to enter the expiration date again.

Then, the chatbot asks the user to provide the secret code of the SMS sent to the electronic card owner’s phone in order to fill the slot “SMS” and secure the payment. Since we implemented a demo, the user needs only to provide a four-digit number to fill this slot.

Finally, a message that asks the user if he wants to confirm the payment is displayed. When the user confirms the payment, the whole booking process is successfully finished. When he refuses, a message that asks him if he is sure that he wants to stop the process is displayed.

When the whole booking process is finished, the “inform_book_form” is deactivated and the “submit_booking” custom action is executed. When the user has confirmed the payment, this custom action generates a ticket Id (5-digit number), saves the booking in the database and displays a booking confirmation message with all reservation details, as shown in figure 4.20. When the process is stopped by the user, it displays to him a message saying that the booking process is stopped.

Before the payment confirmation, the user can also stop the process at any stage he wants, or modify all the data he previously entered. If the user wants to modify the value of a previously filled slot, the chatbot will ask him to enter the data that allows him to fill that slot again instead of asking him to fill the next slot.

If negative user sentiment is detected at any point in the process, the chatbot's responses will be more empathetic, and the chatbot will not ask the user to provide the data required to fill the next slot. It will instead ask the user if he is upset about something and offers him a discount on booking to try to change his mood.

4.3.3. Cancel flight

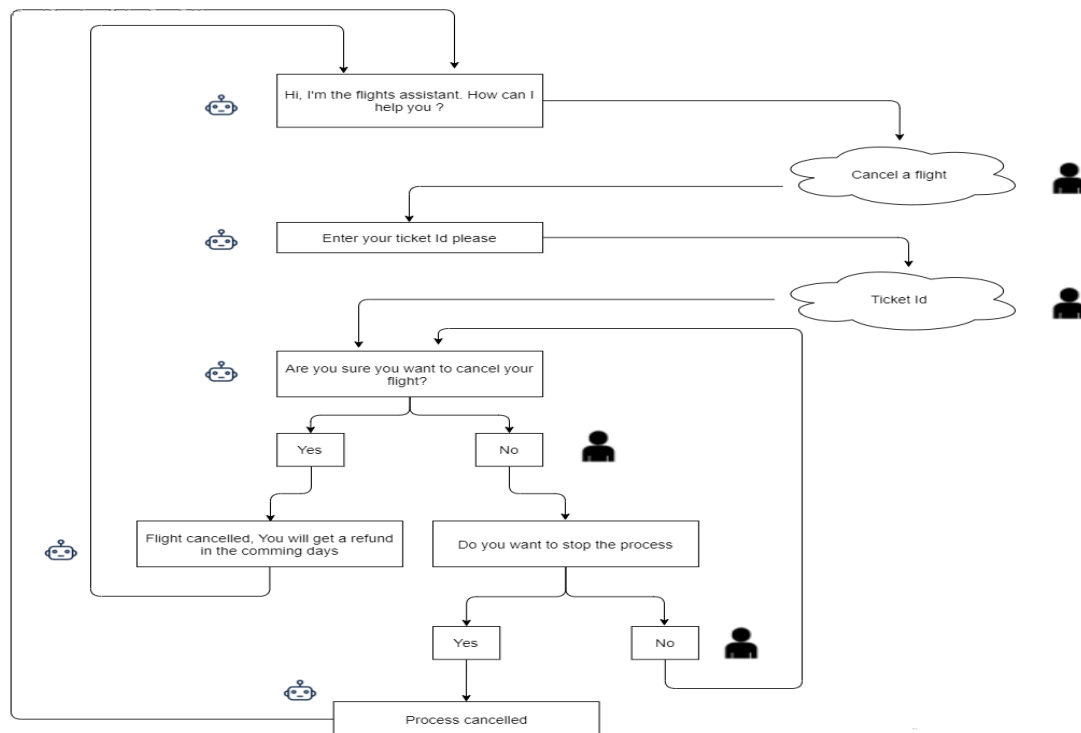


Figure 4.21: Conversational user/bot flow for cancel flight

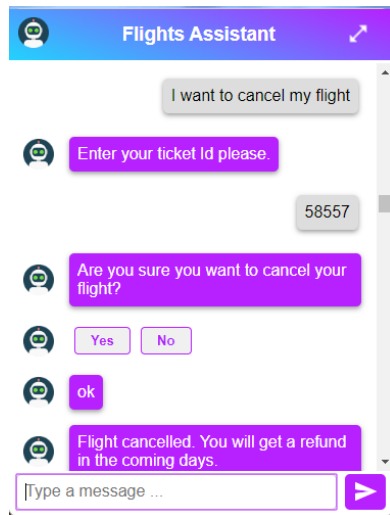


Figure 4.22: Cancel flight

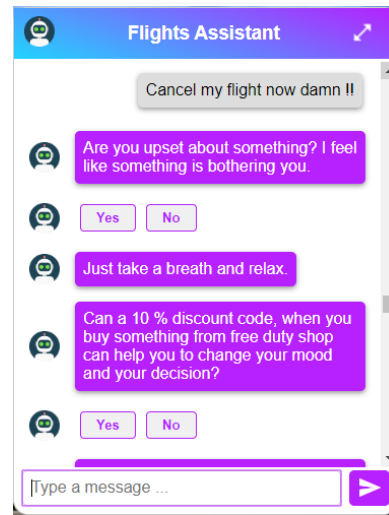


Figure 4.23: Cancel flight when a negative user mood is detected

As seen in figure 4.22, users can cancel flights they have previously booked by simply telling the chatbot that they wish to cancel their bookings. When the chatbot detects this intention, it activates the form “cancel_flight_form” and asks the user to enter his ticket Id in order to fill the required slot “ticket_id” of this form. The custom action “validate_cancel_flight_form” is executed when the user provides the ticket Id. This ticket Id entered by the user is validated by this custom action only if it exists in the database. Finally, a message is displayed asking the user to confirm the ticket cancellation with an affirm and decline buttons. When the user affirms, the operation is successfully completed and when he declines the chatbot will ask him, if he wants to stop the process.

When the whole process is finished, the form “cancel_flight_form” is deactivated and the custom action “submit_cancel_flight” is executed. When the user has confirmed the ticket cancellation, this custom deletes the booking from the database, alerts the user that his flight has been cancelled, and notifies him that he will be reimbursed in the coming days, as shown in figure 4.22. When the process is stopped by the user, it displays him a message saying that the process is stopped.

The user has the possibility to stop the process at any moment. If the chatbot recognizes that the user is upset at any point during the process, its responses will be more empathetic, and will take a new path where it asks the user if he is unhappy about something and offers him a discount code for free duty shops to try to change his mood and his decision about cancelling the flight, as seen in figure 4.23. The cancel flight process cancels the whole reservation, including the return flight.

4.3.4. Cancel return flight

Users can also cancel return flights. This process is too similar to the cancel flight one. But, it cancels the return flight only and not the whole reservation. Once this intent detected, the chatbot activates the form “cancel_return_flight_form” to ask the user to enter his ticket Id in order fill the required slot “ticket_id” of this form. The custom action “validate_cancel_return_flight_form” is executed to validate the ticket Id entered by the user.

When the whole process is completed, the custom action “submit_cancel_return_flight” is executed. If the user has confirmed the return flight cancellation, this custom action removes the return flight from the database, displays to the user that his return flight has been cancelled. Also, it tells him that he will be refunded in the following days. If the process is stopped by the user, the chatbot displays him a message saying that the process is stopped.

The user has the possibility to stop the process at any moment. If negative user sentiment is detected at any stage in the process, the chatbot's answers will be more empathetic, and will take a new path where it asks the user if he is unhappy about something and offers him a discount code for free duty shops to try to change his mood and his decision about cancelling the return flight.

4.3.5. Modify travel date

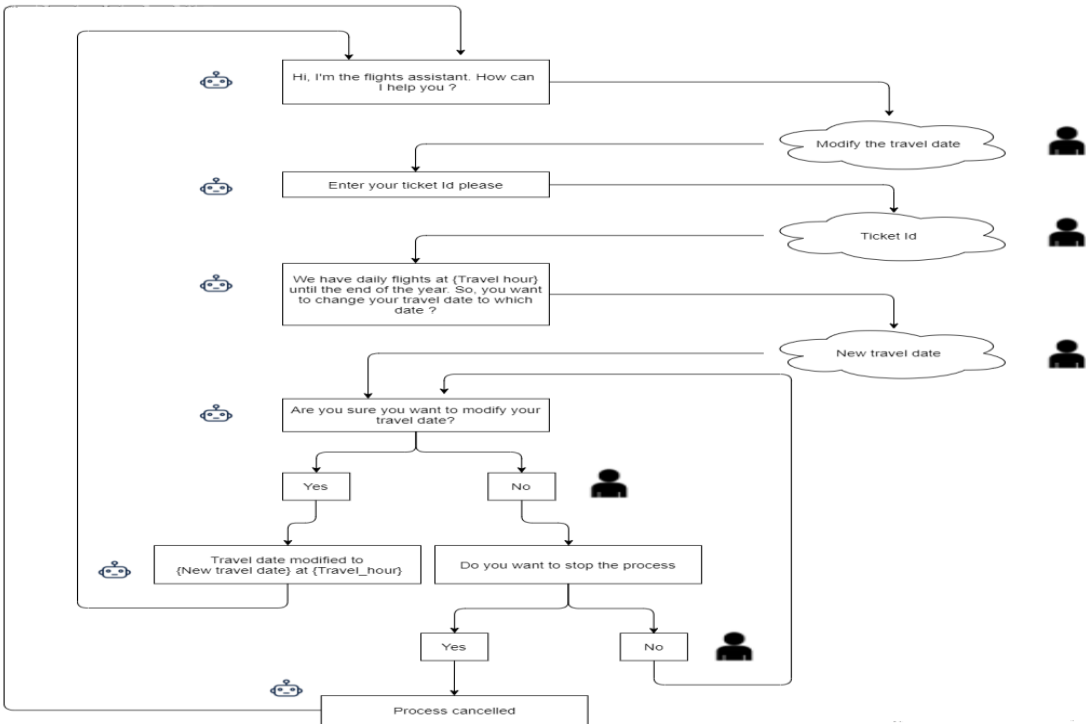


Figure 4.24: Conversational user/bot flow for modify travel date process

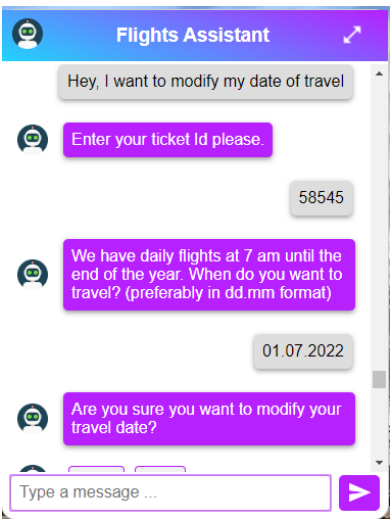


Figure 4.25: Modify travel date

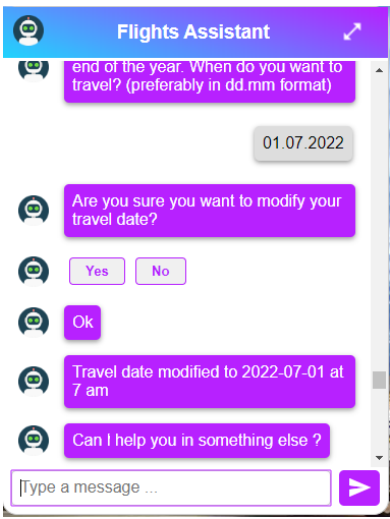


Figure 4.26: Modify travel date confirmation

Users can modify the travel date of a flight they have previously booked by simply telling the chatbot that they wish to modify the travel date, as shown in figure 4.25. When this user intention is detected by the chatbot, the “modify_travel_date_form” is activated to ask the user to enter his ticket Id and the new date of travel in order fill the required slots of this form: the “ticket_id” and “travel_date”. The custom action “validate_modify_travel_date_form” is executed when the user provides the data to fills these two required

slots. The ticket Id entered by the user is validated by this custom action only if it exists in the database. The new date of travel entered by the user is only validated if it is from the current year, at least one day after the current date, and at least one day before the return date if the booking has already a return date. Finally, a message is displayed asking the user to confirm the travel date modification procedure with an affirm and decline buttons. When the user affirms, the operation is successfully completed and when he declines the chatbot will ask him, if he wants to stop the process.

Once the whole process is completed, the “modify_travel_date_form” is deactivated and the custom action “submit_modify_travel_date” is then called. When the user has confirmed the travel date modification, this custom action updates the travel date of that booking in the database and displays the user that his travel date has been modified, as shown in figure 4.26. When the process is stopped by the user, it displays him a message saying that the process is stopped.

The user has the possibility to stop the process at any moment. If negative user sentiment is detected at any stage in the process, the chatbot's responses will be more empathetic, and will take a new path where it asks the user if he is unhappy about something and offers him a discount card for free duty shops to try to change his mood.

4.3.6. Modify return date

Users can also modify return flights date. This process is too similar to the modify flight one, but it modifies the return flight date. Once this intent is detected, the chatbot activates the form “modify_return_date_form” to ask the user to enter his ticket Id and the new return date in order to fill the required slots “ticket_id” and “return_date” of this form. The custom action “validate_modify_return_date_form” is executed when the user enters the data needed to fill these two required slots. The ticket Id entered by the user is validated by this custom action only if it exists in the database and the booked flight has a return flight. The new return date entered by the user is validated only if it is from the current year, at least one day after the current date, and at least one day after the travel date.

Once the whole process is completed, the custom action “submit_modify_return_date” is executed. When the user has confirmed the return date modification, this custom action

updates the return date of that booking in the database and displays to the user a message to notify him that his return date has been modified. When the process is stopped by the user, it displays him a message saying that the process is stopped.

The user has the possibility to stop the process at any moment. If the chatbot detects that the user is upset at any point during the process, its responses will be more empathetic, and it will take a new path where it asks the user if he is unhappy about something and offers him a discount card for free duty shops to try to change his mood.

4.3.7. Make claim

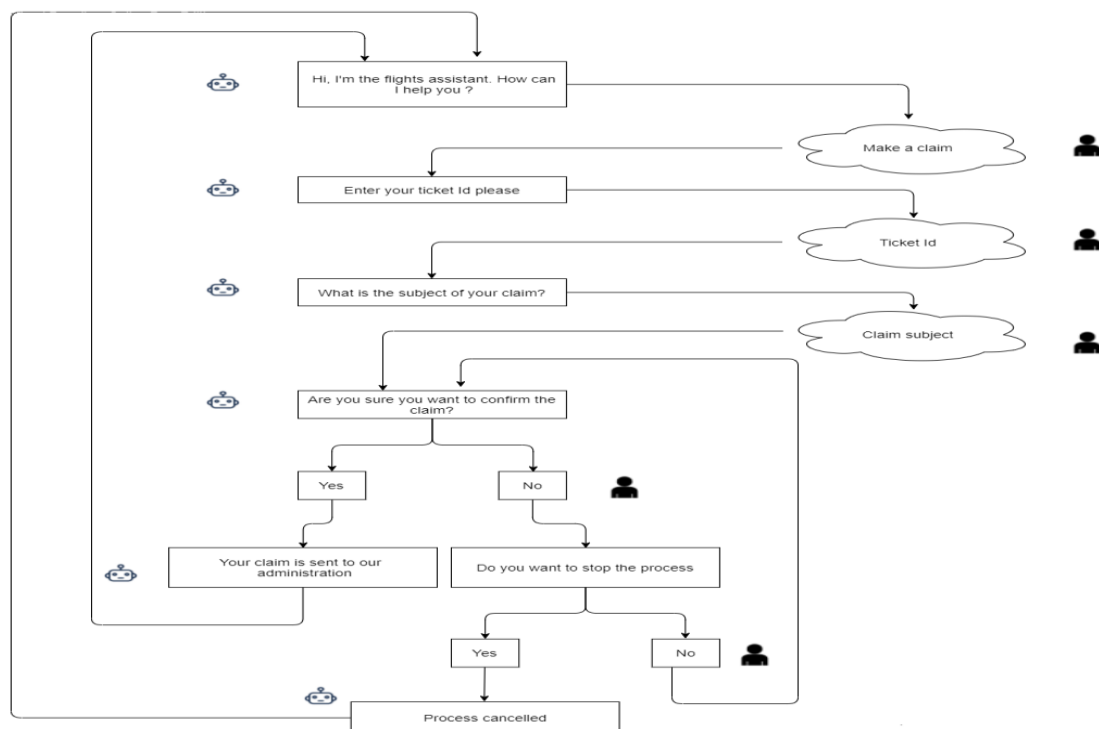


Figure 4.27: Conversational user/bot flow for the claim process

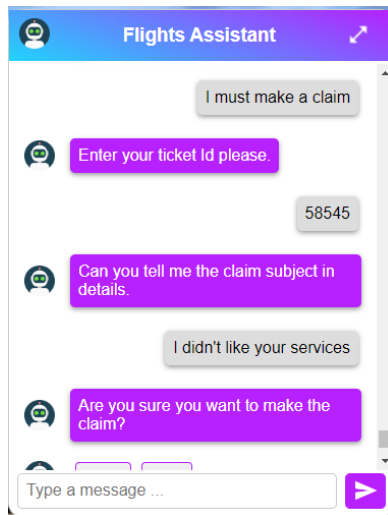


Figure 4.28: Make a claim

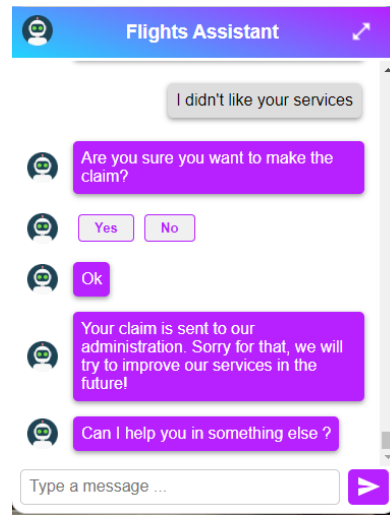


Figure 4.29: Claim confirmation

The chatbot allows users to make claims. As seen in figure 4.28, when the chatbot detects that the user wants to make a claim, the “make_claim_form” is activated to ask the user to enter his ticket Id and the subject of his claim in order to fill the required slots of this form: the “ticket_id” and the “claim_subject”. The custom action “validate_claim_form” is executed when the user enters the data needed to fill these two required slots. The ticket Id entered by the user is validated by this custom action only if it exists in the database. The claim’s subject entered by the user is validated only if it is not too short (more than 8 character). Finally, a message is shown asking the user to confirm the claim procedure with an affirm and decline buttons. When the user affirms, the operation is successfully completed and when he declines the chatbot will ask him, if he wants to stop the process.

Once the whole process is completed, the “make_claim_form” is deactivated and the custom action “submit_claim” is executed. When the user has confirmed the claim, this custom action stores the claim in the database and displays to the user a message saying that his claim is sent, as shown in figure 4.29. When the process is stopped by the user, it displays to him a message saying that the process is stopped.

The user has the possibility to stop the process at any moment. If negative user feeling is detected at any stage of the process, in the same way as when modifying a travel date, the chatbot's answers will be more empathetic, and will take a new path where it asks the user if he is upset about something and offers him a discount card for free duty shops to try to change his mood.

5. Research methodology

For our study, we conducted a survey with a set of participants. In this survey, participants were instructed to test the chatbot functioning, then to respond to the questions of the questionnaire. We divided these participants into two groups. The first group tested the first version of our chatbot while the second tested the second version that can analyze sentiment. During, the user testing, testers are instructed, to perform two tasks. The first task is a simple task in which they will try to book a plane ticket and answer to the question asked by the chatbot until the end of the process. The second task is complex task in which testers are instructed to interrupt the conversation, enter complex sentences and try to show their sentiments, when interacting with the chatbots. After completing these tasks, they were instructed to respond to the questions of the questionnaire. This questionnaire contains 27 questions. The first question is demographic that asks them if they had a previous experience with chatbots. The other 26 questions represent the items of every dimension of our evaluation model. Each tester used the Likert scale from 1” Strongly disagree” to 5 “Strongly agree” to respond all the items of each dimension in order to evaluate the quality of that dimension.

This survey at:

- First, measuring and analyzing the reliability and the validity of the items that we created to evaluate each dimension of our evaluation model, which will permit to evaluate the reliability and the validity of the entire evaluation model.
- Second, to determine whether sentiment can improve the chatbot overall quality and to identify the improved quality dimensions. Therefore, two hypotheses were formed:
 - H1: Users consider that the overall quality of a chatbot that can analyze sentiment is better than the overall quality of the same chatbot that does not analyze sentiment.
 - H2: If the overall quality of a chatbot has improved using sentiment analysis, then the quality of all dimensions improves as well.

5.1. The reliability and validity

Reliability and validity are linked but distinct. Indicators (i.e. items) might be reliable but not valid (i.e. accurate), and vice versa. The rate to which a group of two or more indicators (i.e. items in a questionnaire) contribute to the measurement of a construct is referred as reliability. In contrast, validity is a process that determines how well items measure the construct they were supposed to evaluate [28].

In this study, we will analyze the items reliability and validity for each construct (i.e. dimension in our study) we used in our evaluation model.

5.1.1. Reliability

The Cronbach alpha coefficient is the most frequently used measurement tool to assess a construct's reliability. In some researches, a Cronbach alpha coefficient more than 0.70 and less than 0.90 is recommended to confirm that the items used are reliable. Whereas in others, a Cronbach alpha coefficient greater than 0.7 and less than 0.95 is recommended. A Cronbach alpha less than 0.70 indicates that the items do not capture the construct that they were designed to assess, whereas a Cronbach alpha that is more than 0.95 indicates that the items of the construct are highly reliable, implying considerable redundancy and one or some of them should be deleted [29].

In this study, we will calculate the Cronbach alpha coefficient for each dimension to analyze the related items reliability.

5.1.2. Construct validity

Construct validity is composed of two distinct but related measures: convergent validity and discriminant validity. Both are required to evaluate the construct validity. However, neither is adequate by itself to demonstrate construct validity [28].

5.1.2.1. Convergent validity

The degree to which a measure correlates highly with other measures designed to assess the same construct is referred to as convergent validity [28]. The primary measurements used to measure convergence validity are Composite Reliability (CR) and Average

Variance Extracted (AVE). For each construct a Composite Reliability (CR) higher than 0.7 is recommended while an Average Variance Extracted (AVE) greater than 0.5 is recommended [30].

In this study, we will compute the Composite Reliability (CR) and Average Variance Extracted (AVE) for each quality dimension to analyze the convergent validity of the related items.

5.1.2.2. Discriminant validity

Discriminant validity is demonstrated when each measurement item is weakly correlated with all other dimensions except the one with which the item is associated [31]. Heterotrait-Monotrait ratio (HTMT) is one of most frequently used measure to assess the discriminant validity. If the HTMT values of all constructs is less than 0.9, the discriminant validity is proven [30].

In our study, we will compute the HTMT score for every quality dimension to assess the discriminant validity.

5.2. Chatbot quality improvement with sentiment analysis

In order to test the two previously set hypotheses, we will measure and compare between the two groups: The overall averages, the averages of each dimension, and the means and standard deviations of each item. However, being based on averages only does not allow us to confirm that one of the versions is better than the other, since the difference between averages could be insignificant and happen by chance. For this reason, we will use the t-test [25] to identify whether the difference of means between both groups is statistically significant to test our hypothesis.

A t-test is a statistical method to compare the means between two groups. It aims at determining if the difference in the means between these groups is significant or coincidental. The t-test generates a probability value known as the p-value. If the p-value is less than 0.05, it indicates that the difference in means is significant and did not occur by chance [25].

5.3. Data collection

The survey was created with the online tool “Google Forms”. It was accessible online from "01.05.2020" until "01.06.2020." The survey was posted in two Facebook groups. Both groups consist of computer science students, professors, developers, and researchers. In addition, a group of my acquaintances from various fields participated in the survey. During the first 15 days of the survey, participants tested the functioning of the first chatbot version before being allowed to reply to the questions of the questionnaire. During the survey's remaining time, participants assessed the functioning of the second chatbot version and then answered the questions of the questionnaire.

6. Survey results

This thesis's results chapter will deal with a quantitative research design. In this chapter, we present several statistical analyses of the data collected from our online survey.

A total of 54 participants participated in the survey. 20 of them answered yes to the question “have you chatted with a chatbot before”. The participants were divided into 2 groups. The first group consists of 27 participants who interacted with our chatbot's first version, whereas the second group consists of 27 participants who interacted with our chatbot's second version that can analyze sentiment.

First, based on all the responses of the 54 participants, the reliability and validity of the items used in our evaluation model to evaluate each dimension are analyzed to examine how effective these items are. Finally, we will measure the means obtained from both groups and investigate if the differences in means are statistically significant in order to test our hypotheses.

6.1. Reliability and validity

In order to measure the reliability and validity of the items used to evaluate each dimensions, we use the smartPLS tool.

6.1.1. Reliability

We used Cronbach Alpha to verify the reliability of the items used in our evaluation model to evaluate each dimension. A Cronbach Alpha coefficient between 0.7 and 0.95 is required for each construct or dimension to demonstrate items reliability. All dimensions, as shown in Table 6.1, have a Cronbach Alpha coefficient between 0.7 and 0.95, which is satisfactory and indicates that the items employed in each dimension are reliable.

The Cronbach Alpha is not computed to measure the reliability of the visual appearance dimension. Since, it consists of a single item and reliability can only be performed on two or more items. Furthermore, a single item is considered as very reliable according to Rost et al. [32].

Dimensions	Cronbach Alpha	Composite Reliability(CR)	Average Variance Extracted(AVE)
Usefulness	0.832	0.922	0.855
Ease-of-use	0.800	0.874	0.706
Effectiveness	0.840	0.900	0.750
Efficiency	0.817	0.878	0.709
Responses in unexpected situations	0.827	0.862	0.615
Personality and humanity	0.894	0.920	0.794
Response time	0.825	0.917	0.847
Security and privacy	0.731	0.881	0.787
User satisfaction	0.916	0.947	0.856

Table 6.1: Cronbach Alpha, Composite Reliability, Average Variance Extracted

6.1.2. Validity

It is composed of two related measures the convergent and discriminant validity. These two methods are required to demonstrate the validity of each dimension.

6.1.2.1. Convergent validity

We used both Composite Reliability (CR) and the Average Variance Extracted(AVE) to analyze the convergent validity of the items used in our evaluation model to evaluate each dimension. To demonstrate items convergent validity, each construct or dimension must have a Composite Reliability score higher than 0.7 and an Average Variance Extracted coefficient greater than 0.50. As shown in Table 6.1, all dimensions have a Composite Reliability score higher than 0.7 and an Average Variance Extracted coefficient greater than 0.50, indicating that the convergent validity of the items used in each dimension is proven. The Composite Reliability and Average Variance Extracted are not computed for

the visual appearance dimension. Since, it contains only one item and convergent validity can only be performed on two or more items. Moreover, a single item can be considered as highly valid according to Rost et al. [32].

6.1.2.2. Discriminant validity

We used the Heterotrait-Monotrait ratio (HTMT) to demonstrate the discriminant validity of the items used to evaluate each dimension. A HTMT values less than 0.90 for all dimensions is required to demonstrate discriminant validity. As shown in Table 6.2, all of the HTMT ratios are less than 0.80, indicating that each measurement item is weakly correlated with all other dimensions except the one with which the item is associated and that the discriminant validity is established for all dimensions.

	Usefulness	Ease-of-use	Effectiveness	Efficiency	Visual appearance	Responses in unexpected situations	Personality and humanity	Response time	Security and privacy	User satisfaction
Usefulness										
Ease-of-use	0.548									
Effectiveness	0.462	0.474								
Efficiency	0.646	0.627	0.512							
Visual appearance	0.400	0.335	0.334	0.390						
Responses in unexpected situations	0.503	0.470	0.430	0.641	0.399					
Personality and humanity	0.168	0.314	0.107	0.282	0.074	0.493				
Response time	0.153	0.317	0.488	0.262	0.253	0.205	0.101			
Security and privacy	0.273	0.323	0.231	0.320	0.433	0.277	0.221	0.399		
User satisfaction	0.513	0.360	0.423	0.561	0.284	0.693	0.733	0.173	0.394	

Table 6.2: HTMT ratios of all dimensions

6.2. Chatbot quality improvement with sentiment analysis

We used the MS-Excel tool to measure for each group: the overall average, averages of each dimensions, the means and standard deviations of each items. Also, MS-Excel was used in order to perform the t-test to compare the averages between the two groups.

<i>Dimensions</i>	Items	Mean	Std dev	Average
Usefulness	Usefulness_1	3.51	0.64	3.53
	Usefulness_2	3.55	0.57	
Ease-of-use	Ease-of-Use_1	3.66	0.55	3.53
	Ease-of-Use_2	3.29	0.66	
	Ease-of-Use_3	3.62	0.56	
Effectiveness	Effectivness_1	3.18	0.73	3.46
	Effectivness_2	3.62	0.56	
	Effectivness_3	3.59	0.69	
Efficiency	Efficiency _1	3.37	0.62	3.48
	Efficiency _2	3.37	0.62	
	Efficiency _3	3.70	0.60	
Visual appearance	Visual_appearance_1	3.18	0.68	3.18
Responses in unexpected situations	Responses in unexpected situations_1	3.33	0.55	3.13
	Responses in unexpected situations_2	3.00	0.62	
	Responses in unexpected situations_3	2.77	0.80	
	Responses in unexpected situations_4	3.40	0.74	
Personality and humanity	Personality and humanity_1	1.59	0.57	1.69
	Personality and humanity_2	2.14	0.71	

	Personality and humanity_3	1.33	0.48	
Response time	Response time_1	3.07	0.61	3.20
	Response time_2	3.33	0.62	
Security and privacy	Security and Privacy _1	3.18	0.62	3.38
	Security and Privacy _2	3.59	0.57	
User satisfaction	User satisfaction_1	2.85	0.53	2.71
	User satisfaction_2	2.70	0.60	
	User satisfaction_3	2.59	0.79	
Overall average = 3.13				

Table 6.3: Survey Results of the first group

As shown in Table 6.3, the overall average of the first group is 3.13 from a maximum value of 5, which is a satisfactory rate. The overall averages of all dimensions, except personality and humanity, and user satisfaction, are greater than 3.00, which could also be considered satisfactory. On the other hand, the average of the dimension personality and humanity is 1.69, which is considered as a very poor score, while the average of the dimension customer satisfaction is 2.71, which is considered a poor score.

<i>Dimensions</i>	Items	Mean	Std dev	Average
Usefulness	Usefulness_1	3.37	0.92	3.48
	Usefulness_2	3.59	0.88	
Ease-of-use	Ease-of-Use_1	3.70	0.54	3.60
	Ease-of-Use_2	3.37	0.56	
	Ease-of-Use_3	3.74	0.65	
Effectiveness	Effectivness_1	3.07	0.78	3.40
	Effectivness_2	3.77	0.93	
	Effectivness_3	3.37	0.74	
Efficiency	Efficiency _1	3.44	0.64	3.54
	Efficiency _2	3.44	0.75	
	Efficiency _3	3.74	0.71	

Visual appearance	Visual_appearance_1	3.14	0.90	3.14
Responses in unexpected situations	Responses in unexpected situations_1	3.51	0.64	3.42
	Responses in unexpected situations_2	2.85	0.71	
	Responses in unexpected situations_3	3.33	0.73	
	Responses in unexpected situations_4	4.00	0.78	
Personality and humanity	Personality and humanity_1	3.70	1.03	3.54
	Personality and humanity_2	3.88	0.84	
	Personality and humanity_3	3.03	0.97	
Response time	Response time_1	2.96	0.58	3.14
	Response time_2	3.33	0.62	
Security and privacy	Security and Privacy _1	3.22	0.50	3.42
	Security and Privacy _2	3.62	0.74	
User satisfaction	User satisfaction_1	3.48	0.80	3.59
	User satisfaction_2	3.59	0.84	
	User satisfaction_3	3.70	1.13	
			Overall average = 3.43	

Table 6.4: Survey Results of the second group

As shown in Table 6.4, the overall average of the second group is 3.43 from a maximum value of 5, which is a satisfactory rate. The total averages of all dimensions are greater than 3.00, which could also be considered satisfactory.

The total averages of the survey results of both groups are shown in Table 6.5. The overall average of the second group that tested the chatbot with sentiment analysis is 3.43, which

is greater than the overall average of the first group that tested the chatbot without sentiment analysis, which is 3.13. The t-test was also performed to determine whether there was a significant difference in averages between groups. The resulting p-value is 0.014, which is less than 0.05, indicating that there is a significant difference in overall averages between the two groups.

First group overall average	Second group overall average	P-value (unpaired)
3.13	3.43	0.014

Table 6.5: The P-value of the unpaired t-test of overall averages between the two groups

The survey results averages per dimension for both groups are shown in Table 6.6. The t-test was also performed on each dimension's averages to verify whether there was any significant difference in the dimensions' averages between the two groups.

The overall averages in the majority of dimensions of both groups are almost equal, and the resulting p-value for each dimension was much higher than 0.05, indicating that there is an insignificant difference between the two groups in the majority of dimensions' averages (Usefulness, Ease-of-use, Effectiveness, Efficiency, Visual appearance, Response time, and Security and privacy).

On the other hand, the overall average of the Responses in unexpected situations in the second group is 3.42, which is greater than the average of this dimension in the first group, which is 3.13. The resulting p-value is 0.063, which is close to but still more than 0.05. As a result, we cannot confirm that the difference in mean values on this dimension between the two groups is significant since 0.063 is greater than 0.05, implying that there is no significant difference. Furthermore, the overall average of the personality and humanity dimension in the second group is 3.54, which is much greater than the average of this dimension in the first group, which is 1.69. The resulting p-value is 1.594×10^{-12} , which is less than 0.05. As a result, we can confirm that the difference in mean values on this dimension between the two groups is statistically significant. Finally, the overall average of the dimension User satisfaction in the second group is 3.59, which is much greater than the average of this dimension in the first group, which is 2.71. The obtained p-value is 5×10^{-4} , which is less than 0.05. As a result, we can confirm that the difference in mean values on this dimension between the two groups is statistically significant.

Dimensions	First group average	Second group average	P- value (Unpaired)
Usefulness	3.53	3.48	0.847
Ease-of-use	3.53	3.60	0.661
Effectiveness	3.46	3.40	0.580
Efficiency	3.48	3.54	0.662
Visual appearance	3.18	3.14	0.866
Responses in unexpected situations	3.13	3.42	0.063
Personality and humanity	1.69	3.54	1.594×10^{-12}
Response time	3.20	3.14	0.718
Security and privacy	3.38	3.42	0.804
User satisfaction	2.71	3.59	5×10^{-4}

Table 6.6: : The P-value of the unpaired t-test of overall averages per dimension between both groups

Hypotheses testing:

Hypotheses	Support
H1: Users consider that the overall quality of a chatbot that can analyze sentiment is better than the overall quality of the same chatbot that does not analyze sentiment	Yes
H2: If the overall quality of a chatbot has improved using sentiment analysis, then the quality of all dimensions improves as well	No

Table 6.7: Hypotheses and results

The first hypothesis could be accepted for two main factors. First, the overall average of the first chatbot version is equal to 3.43 and higher than the overall average of the second version, which is equal to 3.13. Second, the p-value of the t-test that measures the difference in overall averages between the two chatbot versions is equal to 0.014 which is less than 0.05, indicating a significant difference between the two overall averages.

The second hypothesis could not be accepted since the resulting p-value in the majority of dimensions between the two chatbot versions was higher than 0.05, indicating an insignificant difference between the two versions in the majority of dimensions' averages. Furthermore, Only the overall averages of the dimensions User satisfaction, and Personality and Humanity are increased in the second chatbot version with a resulting p-values less than 0.05, indicating a significant difference.

7. Discussion

This chapter comprises of what can be observed from the survey findings and a comparison of these results with previous studies in order to answer our research questions.

The goal of this study was to first develop and evaluate a general assessment model with ten quality dimensions (Usefulness, Ease-of-use, Effectiveness, Efficiency, Visual appearance, Responses in unexpected situations, Personality and humanity, Response time, Security and privacy, Response time, and User satisfaction) to evaluate the quality of chatbots and compare the quality of different versions or types of chatbots. The second goal of this study was to implement two versions of the same chatbot (a first version without sentiment analysis and a second improved version that can analyze emotions and respond accordingly) and evaluate and compare their performances using the evaluation model we developed in order to demonstrate whether sentiment analysis can improve the overall quality of the chatbot, and if so to identify the quality dimensions that were enhanced with sentiment analysis.

We created the following research questions to guide us during our study:

RQ1: What are the quality dimensions that can be used to evaluate chatbots?

RQ2: Can sentiment analysis improve the quality of a chatbot?

We also set the following hypotheses to assist us in answering our second research question:

- Users consider that the overall quality of a chatbot that can analyze sentiment is better than the overall quality of the same chatbot that does not analyze sentiment.
- If the overall quality of a chatbot has improved using sentiment analysis, then the quality of all dimensions improves as well.

7.1. Chatbot evaluation

In this section, we will discuss the previous chapter's findings in order to answer our first research question.

As shown in Table 6.1, the results show that the Cronbach Alpha coefficient of each dimension was between 0.70 and 0.91 (a Cronbach Alpha score between 0.70 and 0.95 is required, indicating that every set of items used to evaluate each dimension was reliable).

The results also reveal that the composite reliability (CR) of each dimension was greater than 0.70 (A CR score greater than 0.70 is required) and the Average Variance Extracted (AVE) score of each dimension was greater than 0.5 (An AVE score greater than 0.5 is required), as shown in Table 6.1. Thus, the convergent validity of the items used to evaluate each dimension is demonstrated. As seen in Table 6.2, the Heterotrait-Monotrait ratio (HTMT) ratios of all dimensions were less than 0.70 (HTMT ratios less than 0.90 are recommended for each dimension), indicating that each measurement item is weakly correlated with all other dimensions except the one with which the item is associated, and demonstrating the discriminant validity of the items used to evaluate each dimension. Because both convergent and discriminant validity were proven, the validity of the items used to assess each dimension is confirmed, which indicates that all the items used in our evaluation model to evaluate the quality of chatbots are valid and reliable.

Our findings are the same as Lin et al. [30] that demonstrated the reliability and validity of the items employed to assess each construct.

7.2. Chatbot improvement with sentiment analysis

In this section, we will discuss the results of the previous chapter in order to respond to our second research question.

7.2.1. Chatbot overall quality improvement

Our results indicated that the first hypothesis we tested, “Users consider that the overall quality of a chatbot that can analyze sentiment is better than the overall quality of the same chatbot that does not analyze sentiment”, could be accepted. Because the second chatbot's overall average (3.43) was greater than the first version's overall average (3.13),

the resulting p-value was equal to $0.014 < 0.05$, indicating a significant difference in the overall averages between the two versions. Our findings support the theory that sentiment analysis improves the chatbot's quality when compared to the same chatbot without emotion analysis, which is consistent with the previous studies conducted by Almansor et al. [33], by Abedin et al. [34], and by Sutoyo et al. [35].

7.2.2. Dimensions improvement

Our findings showed that the second hypothesis we tested, “If the overall quality of a chatbot has improved using sentiment analysis, then the quality of all dimensions improves as well”, could not be accepted. Because, the results showed that the majority of dimension averages were almost similar between the two chatbots, and the resulting p-value of each dimension between the two chatbot versions was greater than 0.05, suggesting an insignificant difference. Furthermore, the results indicate that the overall average of the dimension Responses in unexpected situations of the second chatbot version (average equal to 3.42) was greater than the overall average of this dimension in the first version (average equal to 3.13). However, the p-value of this dimension between the two versions was equal to 0.06, which is close to 0.05 but still greater. As a result, our findings cannot confirm that the second version handled unexpected situations better than the first, and that this dimension is enhanced in the second version.

The overall average of the dimension Personality and humanity, on the other hand, has increased from 1.69 to 3.54 in the second version, with a resulting p-value equal to $1.594 \times 10^{-12} < 0.05$, indicating a significant difference in averages between the two chatbot versions. Furthermore, the average of the dimension User satisfaction increased from 2.71 to 3.59, with a p-value of $5 \times 10^{-4} < 0.05$, indicating a significant difference in averages between the two chatbots. As a result, our data demonstrated that the Personality and humanity, as well as the User satisfaction dimensions, have improved in the second version.

Our results showed that sentiment analysis did not increase the quality of the following quality dimensions: Usefulness, Ease-of-use, Effectiveness, Efficiency, Visual appearance, Responses in unexpected situations, Response time, and Security and privacy. These findings demonstrated that sentiment analysis increased just two dimensions: Personality and humanity and User satisfaction. These results are consistent

with the findings of Sutoyo et al. [35] who proved that sentiment analysis can improve chatbots' Personality and humanity, and increase the user's delight. However, our findings contradict the findings of Almansor et al. [33] which revealed that a chatbot that can analyze emotions handled responses in unexpected situations better than the same chatbot that cannot analyze sentiment. The difference between our findings and those of Almansor E et al. could be due to the difference in the evaluation methodologies. They used an automated evaluation technique that did not require human testers to evaluate and compare the chatbot without sentiment and the chatbot with sentiment, whereas the evaluation model we used was based on human testers. Furthermore, employing a larger sample of testers could help us to confirm whether sentiment analysis can improve the chatbot's ability to handle unexpected situations.

Furthermore, our findings contradict the findings of Abedin et al. [34] who proved that sentiment analysis enhanced the chatbot accuracy of understanding the user's intentions (i.e. The effectiveness in our evaluation model) when compared to the same chatbot that cannot detect sentiments. This difference in findings between our study and theirs may be due to the fact that they trained a model to predict the user's sentiment based on his input. Then, they trained another model to predict the user's intent based on the predicted sentiment. On the other hand, we trained a single model that predict both the user's intent and sentiment.

To summarize, our findings showed that sentiment analysis increased the overall quality of a chatbot, when compared to the same chatbot without sentiment analysis. Furthermore, sentiment analysis improved the Humanity and personality of the chatbot, and the User satisfaction. However, sentiment analysis did not improve the majority of dimensions such as the accuracy of understanding the user's intents.

8. Conclusion and outlook

This chapter will conclude the study by summarizing the important research results in connection to the research goals. It will also review the study's limitations and propose recommendations for future research.

In this thesis, we developed a general evaluation model based on user's experience to evaluate and compare the quality of chatbots by examining 10 chatbot dimensions. This strategy necessitates human testers interacting with the chatbot. Following that, the data from user testing is gathered quantitatively using surveys.

In this study, we also implemented two chatbot versions using the Rasa framework (one that cannot understand sentiments and one that can analyze sentiment and respond accordingly) and evaluated and compared them using our evaluation model to investigate whether sentiment analysis can improve chatbot quality and to identify the dimensions improved by sentiment analysis.

Our findings indicated that the items used in the evaluation model to evaluate the quality of chatbots, were valid and reliable. Furthermore, the results of the comparison between the two chatbot versions using the evaluation model showed that sentiment analysis increased the chatbot's quality. However, it did not increase the majority of dimensions such as effectiveness (i.e. the accuracy of understanding user intention) and improved only two dimensions: The Personality and humanity, and user satisfaction.

In this thesis, the chatbots general evaluation model that we developed might be a useful tool for developers, researchers and companies to have a general overview of the quality of chatbots they wish to evaluate or to compare, which can address the gaps of most of existing studies that have not presented a general evaluation framework to evaluate chatbots.

8.1. Limitations

Our results demonstrate that the questionnaire utilized by our evaluation model is valid and reliable, and that sentiment analysis improves chatbots quality when compared to the same version without sentiment analysis. This study, however, has some limitations:

First, due to time constraints, the data was collected from a relatively small sample, which means that we cannot generalize our findings to the entire population. To provide more conclusive results, a larger sample should be used.

Also, our evaluation model only includes a quantitative method to evaluate chatbots, which is based on statistical analysis, and does not include a qualitative approach, such as interviewing human testers, which can be more revealing since testers may have comments about some points concerning the quality of the chatbot.

8.2. Future Research

For future study, we propose using a large sample of human testers to increase the accuracy of the findings, and to be able to generalize the results to the broad population.

Furthermore, we recommend that our evaluation model include a qualitative approach that interviews testers about the quality of chatbots. As a result, the assessment model will include both quantitative and qualitative approaches, which may improve its efficiency to evaluate the performance of chatbots.

References

1. Suket, A., Kamaljeet, B. & Sarabjit, S. (2013). Dialogue System: A Brief Review. arXiv. <https://doi.org/10.48550/arXiv.1306.4134>
2. Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1), 36–45. <https://doi.org/10.1145/365153.365168>
3. Ashfaq, M., Jiang, Y., Shubin, Yu. & Loureiro S. (2020). I, Chatbot: Modeling the determinants of users' satisfaction and continuance intention of AI-powered service agents. *Telematics and Informatics*, 54. <https://doi.org/10.1016/j.tele.2020.101473>
4. Casas, J., Tricot, M., Abou Khaled, O., Mugellini, E., & Mauroux, P. (2020). Trends & Methods in Chatbot Evaluation. *ICMI '20 Companion: Companion Publication of the 2020 International Conference on Multimodal Interaction*, 280-286. <https://doi.org/10.1145/3395035.3425319>
5. Hao, S., & Hao, G. (2020). A Research on Online Grammar Checker System Based on Neural Network Model. *Journal of Physics: Conference Series*, 1651. <https://doi.org/10.1088/1742-6596/1651/1/012135>
6. Loiacono, E., & Watson, R. (1999). WebQual: A Web quality instrument. *AMCIS 1999 Proceedings*, 349. <https://aisel.aisnet.org/amcis1999/349>
7. Canonico, M., & Russis, L.D. (2018). A Comparison and Critique of Natural Language Understanding Tools.
8. Vinkler, M. L. (2020). Conversational Chatbots with Memory-based Question and Answer Generation [Master's thesis, Linköping University]. <https://www.diva-portal.org/smash/get/diva2:1510358/FULLTEXT01.pdf>
9. Kulatska, I. (2019). ArgueBot: Enabling debates through a hybrid retrieval-generation-based chatbot [Master's thesis, University of Twente]. https://essay.utwente.nl/79791/1/kulatska_MA_eemcs.pdf
10. Jurafsky, D., & Martin, J.H. (2021). Chatbots & Dialogue Systems. *Speech and Language Processing*, 12-18

11. Ahmed, S. (2019). An Architecture for Dynamic Conversational Agents for Citizen Participation and Ideation [Master's thesis, Technical University of Munich].
12. Härkönen, A.P. (2021). Computationally clarifying user intent for improved question answering [Master's thesis, Tampere university]. https://aaltodoc.aalto.fi/bitstream/handle/123456789/112845/master_H%C3%A4rk%C3%B6nen_Ari-Pekka_2022.pdf
13. Pavel, I. (2021). COMPARING CHATBOT FRAMEWORKS: A STUDY OF RASA AND BOTKIT [Master's thesis, Aalto university]. <https://trepo.tuni.fi/bitstream/handle/10024/132928/PavelImran.pdf>
14. Mamatha, M., & Sudha, C. (2021). Chatbot for E-Commerce Assistance: based on RASA. *Turkish Journal of Computer and Mathematics Education*, 12(11), 6173-6179.
15. Tyman, K., Lutz, M., Palsbröker, P., & Gips, C. (2019). GerVADER - A German Adaptation of the VADER Sentiment Analysis Tool for Social Media Texts. *LWDA*.
16. Walker, M. A., Litman, D. J., Kamm, C. A., & Abella, A. (1997). PARADISE: a framework for evaluating spoken dialogue agents. *ACL '98/EACL '98: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, 271-280. <https://doi.org/10.3115/976909.979652>
17. Carletta, J. (1996). Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, 22(2), 249-254.
18. Kuligowska, K. (2015). Commercial Chatbot: Performance Evaluation, Usability Metrics and Quality Standards of Embodied Conversational Agents. *Professionals Center for Business Research*, 2(02), 1–16.
19. Jaded, M., & Varia, N. (2017). Perspectives for Evaluating Conversational AI. *arXiv*. <https://doi.org/10.48550/arXiv.1709.04734>
20. Radziwill, N. M., & Benton, M. C. (2017). Evaluating Quality of Chatbots and Intelligent Conversational Agents. *arXiv*. <https://doi.org/10.48550/arXiv.1704.04579>

21. Ventakesh, A., Khatri, C., Ram, A., Guo, F., Gabriel, R., Nagar, A., ... Raju, A. (2018). On Evaluating and Comparing Open Domain Dialog Systems. arXiv. <https://doi.org/10.48550/arXiv.1801.03625>
22. Jain, M., Kumar, P., Kota, R., & Patel, S. N. (2018). Evaluating and Informing the Design of Chatbots. DIS '18: Proceedings of the 2018 Designing Interactive Systems Conference, 895-906. <https://doi.org/10.1145/>
23. Maroengsit, W., Piyakulpinyo, T., Phonyiam, K., Pongnumkul, S., Chaovalit, P., & Theeramunkong, T. (2019). A Survey on Evaluation Methods for Chatbots. ICIET 2019: Proceedings of the 2019 7th International Conference on Information and Education Technology, 111-119. <https://doi.org/10.1145/3323771.3323824>
24. Duijst, D. (2017). Can we Improve the User Experience of Chatbots with Personalisation?[Master's thesis, University of Amsterdam].
25. Kim, T.K. (2015). T test as a parametric statistic. Korean journal of anesthesiology, 68(6), 540–546. <https://doi.org/10.4097/kjae.2015.68.6.540>
26. Mouritsen, M.L., Davis, J.F., & Jones, S.C. (2016). ANOVA Analysis of Student Daily Test Scores in Multi-Day Test Periods, 12(2), 73–82.
27. Davis, F. D. (1985). A Technology Acceptance Model for Empirically Testing New End-User Information Systems: Theory and Results. Massachusetts Institute of Technology.
28. Tomiuk, D. (2005). The Impact of Site-Communality on the Attitudinal and Behavioural Components of Site-Loyalty: A Cross-Sectional Study [Ph.D, McGill University]. pp. 208-2014
29. Tavakol, M., & Dennick, R. (2011). Making sense of Cronbach's alpha. International journal of medical education, 2, 53–55. <https://doi.org/10.5116/ijme.4dfb.8dfd>
30. Lin L, Huang Z., Othman, B., & Luo, Y. (2020). Let's make it better: An updated model interpreting international student satisfaction in China based on PLS-SEM approach. PLoS ONE 15(7): e0233546. <https://doi.org/10.1371/journal.pone.0233546>

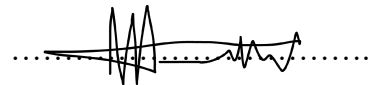
31. Gefen, D., & Straub, D. (2005). A Practical Guide To Factorial Validity Using PLS-Graph: Tutorial And Annotated Example. *Communications of the Association for Information Systems*, Vol. 16 (5). <https://doi.org/10.17705/1CAIS.01605>
32. Rost, D.H. & Sparfeldt, J.R. & Buch, S. (2008). Kann denn Kürze Sünde sein? - Erfassung schulfachspezifischer Interessen mit nur einem Item [single-item-assessment of school-subject-specific interests]. 225-237.
33. Almansor, E.H., Hussain, F.K. & Hussain, O.K. (2021). Supervised ensemble sentiment-based framework to measure chatbot quality of services. *Computing* 103, 491–507. <https://doi.org/10.1007/s00607-020-00863-0>
34. Abedin, A. F., Mamun, A. I., Nowrin, R. J., Chakrabarty, A., Mostakim, M., & Naskar, K. (2021). A Deep Learning Approach to Integrate Human-Level Understanding in a Chatbot. *arXiv*. <https://doi.org/10.48550/arXiv.2201.02735>
35. Sutoyo, R., Chowanda, A., Kurniati, A., & Wongso, R. (2019). Designing an Emotionally Realistic Chatbot Framework to Enhance Its Believability with AIML and Information States. *Procedia Computer Science*, 157, 621-628. <https://doi.org/10.1016/j.procs.2019.08.226>

Statutory Declaration

I hereby declare that I am writing this work independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. Neither the current version nor any other version of this thesis has already been submitted to another department of the Ruhr West University of Applied Sciences or to another academic university.

Gelsenkirchen, 11.07.2022

(Date)

A handwritten signature in black ink, consisting of a series of loops and a final flourish, positioned over a dotted horizontal line.

(Signature)