

Technische Universität Berlin

Institute of Software Engineering and Theoretical Computer Science
Quality and Usability Lab

Fakultät IV
Einsteinufer 17
10587 Berlin
www.eecs.tu-berlin.de



Master Thesis

Implementation of a Framework for the Rapid Development of Conversational Interfaces

Muhammad Ahsan Shahid

Matriculation Number: 0389868
17.07.2020

Examiner:

Prof. Dr.-Ing. Sebastian Möller

Second Examiner:

Prof. Dr. Axel Küpper

Supervisors:

Dr.-Ing. Stefan Hillmann
M.Sc. Jan Nehring

Acknowledgment

Primarily, I would like to thank my supervisor Prof. Dr. Ing. Sebastian Möller for giving me the opportunity to accomplish the research under his excellent supervision. I am also grateful to Prof. Dr. Axel Küpper for the help and support as a second supervisor of my thesis.

I am deeply grateful to Stefan Hillman from Technische Universität Berlin and Jan Nehring from DFKI-German Research Centre for Artificial Intelligence, for arranging regular meetings to check with the progress and providing every possible help. I am really thankful to both of them for having constructive discussions and thorough guidance throughout a hard time, that pushed me to achieve this accomplishment.

I would like to thank my friends and all the people that have spared the time to read this thesis and provided me valuable feedback.

I hereby declare that the thesis submitted is my own, unaided work, completed without any un-permitted external help. Only the sources and resources listed were used.

Place, Date

Berlin, 17.07.2020

.....
(Signature [Muhammad Ahsan Shahid])

Abstract

The trend of developing chatbots at a corporate level has been raised for the last few years. The existing state of the art dialogue frameworks like IBM's Watson Assistant [1], RASA [2], and PLATO [3] are providing great support in this regard. But still there exist various hindrances that need to be addressed in order to make this emerging technology noticeable. It is necessary to keep the frameworks as simple as possible without compromising on its abilities so that a non-technical person could also be able to develop a chatbot according to his/her needs.

This thesis illustrates the design, implementation, and evaluation of the chatbot framework developed using novel Modular Architecture. The chatbot designed using this framework is named as "Frankenbot". The chatbot uses the RASA's Natural Language Understanding(NLU) Model for the semantical interpretation of the users' utterances in order to fetch the required fields for the chatbot.

Categorically, it provides the designers and developers to follow a simpler structure for creating a new chatbot with an additional added feature of modules re-usability. Furthermore, it has also provided support for users to chat on multiple topics at the same time. It means once the user switches the dialogue from one topic to another then instead of reinstating the chatbot, he/she can resume the chat for the previous topic where he/she left off.

This newly introduced strategy has been evaluated with the help of the research study conducted using different surveys. The users gave their opinion about what they experienced while interacting with the Frankenbot. They also provided a judgment for the quality of the chatbot. The majority of the users rated the overall experience with the chatbot as good. Also, the hedonic and pragmatic quality of the chatbot was graded as neutral but near to desirable.

Zusammenfassung

Der Trend zur Entwicklung der Chatbots auf Unternehmensebene hat sich seit einigen Jahren verstärkt. Bestehende hochmoderne Dialog-Frameworks wie IBM's Watson Assistant [1], RASA [2] und PLATO [3] bieten diesbezüglich hervorragende Unterstützung. Es gibt jedoch noch verschiedene Hindernisse, die angegangen werden müssen, um diese aufkommende Technologie erfolgreich zu verbreiten. Es ist notwendig, die Frameworks so einfach wie möglich zu halten, ohne ihre Fähigkeiten zu beeinträchtigen, damit eine nicht technische Person auch einen Chatbot nach ihren Bedürfnissen entwickeln kann.

Diese Arbeit veranschaulicht das Design, die Implementierung und die Evaluierung des Chatbot-Frameworks, das unter Verwendung einer neuartigen modularen Architektur entwickelt wurde. Der mit diesem Framework entworfene Chatbot heißt "Frankenbot". Der Chatbot verwendet das NLU-Modell (Natural Language Understanding) von RASA zur semantischen Interpretation der Äußerungen des Benutzers, um die erforderlichen Felder für den Chatbot abzurufen.

Grundsätzlich können Designer und Entwickler die einfachere Struktur zum Erstellen eines neuen Chatbots mit zusätzlichen Funktionen zur Wiederverwendbarkeit von Modulen befolgen. Darüber hinaus wurde den Benutzern die Möglichkeit geboten, gleichzeitig über mehrere Themen zu chatten. Sobald der Benutzer den Dialog von einem Thema zum anderen wechselt, kann er den Chat für das vorherige Thema fortsetzen, in dem er aufgehört hat, anstatt den Chatbot wiederherzustellen.

Diese neu eingeführte Strategie wurde mit Hilfe der Forschungsstudie bewertet, die anhand verschiedener Umfragen durchgeführt wurde. Die Benutzer gaben ihre Meinung zu dem ab, was sie bei der Interaktion mit dem Frankenbot erlebt haben. Sie beurteilten auch die Qualität des Chatbots. Die Mehrheit der Benutzer bewertete die Gesamterfahrung mit dem Chatbot als gut. Auch die hedonische und pragmatische Qualität des Chatbots wurde als neutral, aber nahezu wünschenswert eingestuft.

Contents

List of Figures	ix
List of Tables	xi
Listings	xii
1. Introduction	1
1.1. Motivation	2
1.2. Scope of the Thesis	3
1.2.1. Challenges	3
1.2.2. Contributions	4
1.3. Structural Outline	5
2. Foundations and Related Work	6
2.1. Chatbots History	6
2.2. Dialogue Systems	7
2.2.1. Task Oriented Agents	7
2.2.2. Chatbots	8
2.3. Communication Channels	9
2.4. Chatbots Components and Tasks	10
2.4.1. Natural Language Processing(NLP)	10
2.4.2. Response Generation	13
2.4.3. Knowledge Base Designing and Management	15
2.4.4. Dialogue Management	16
2.5. Applications	21
2.5.1. Virtual Personal Assistants(VPAs)	21
2.5.2. User Domain-Specialist Bots	21
2.6. Evaluation Methods	21
2.6.1. Qualitative Methods	22
2.6.2. Quantitative Methods	22
2.6.3. Cross-systems Comparison	23
2.6.4. Evaluation Metrics	23
2.6.5. Standard AttrakDiff Tool	23
3. Frankenbot: System Overview and Capabilities	25
3.1. Why Frankenbot?	25

Contents

3.2. Frankenbot: Experimental Chatbot	26
3.2.1. Domain	26
3.2.2. Design	26
3.2.3. Training for NLU	26
3.2.4. Response Generation	27
3.3. Frankenbot as Concept	27
3.4. Frankenbot: Approach	28
3.4.1. Module	28
3.4.2. Modular Architecture	29
3.5. Frankenbot: System Architecture	33
3.5.1. RASA Framework for NLU	34
3.5.2. Frankenbot Framework	36
4. Experiments and Evaluations	53
4.1. Conducted Surveys	54
4.2. Experimental Setup	55
4.2.1. Explanation of the Chatbot	55
4.2.2. Tasks	56
4.2.3. Interview	57
4.2.4. Frankenbot's Experience Questionnaire	57
4.2.5. AttrakDiff Single Evaluation	58
4.3. Quantitative Analysis	58
4.3.1. Frankenbot's Experience Survey	58
4.3.2. Evaluation via AttrakDiff	71
4.4. Qualitative Analysis	76
4.4.1. Natural Language Understanding(NLU)	77
4.4.2. Chatbot Guidance and Intelligence	77
4.4.3. Interaction	77
4.4.4. Switching Topics	78
4.4.5. Detective Game	78
4.4.6. User Interface	78
5. Discussion and Conclusion	79
5.1. Summary	79
5.2. Discussion	80
5.2.1. Framework	80
5.2.2. Evaluation	81
5.3. Future Work	82
Bibliography	84
Appendices	89
A. Hyperlink to Deployed Frankenbot	89
B. Information for Participants	89

C.	Instructions for Participants	89
D.	Request for Participants	89
E.	Frankenbot's Dialogue Structure (frankenbot.json)	90
F.	RASA NLU Training Data Stats	90
F.1.	Module 1(detective_tree.json)	90
F.2.	Module 2(general_tree.json)	90
G.	RASA Configuration File(config_spacy.yaml)	90
H.	Surveys	91
H.1.	Frankenbot's Experience Survey	91
H.2.	Frankenbot's Evaluation via AttrakDiff	103

List of Figures

2.1. Retrieval-based Methods Example [4]	13
2.2. Generative-based Methods Example [4]	14
2.3. Dialogue management Architecture and information flow [5].	17
2.4. Classification of Dialogue Management Approaches. Top most level shows three different approaches: handcrafted, statistical and hybrid. Next step illustrates different methods associated with each approach. Lastly, practical tools (white boxes), research tools (grey boxes), and research prototypes (blue boxes) are defined [5].	21
2.5. Main Concept of the AttrakDiff's Model [6].	24
3.1. Frankenbot Overview	28
3.2. Traditional dialogue tree containing two sub-trees.	30
3.3. Dialogue with two sub-trees using Modular Architecture	31
3.4. Module usability within a chatbot	32
3.5. Module usability within a chatbot	33
3.6. Module usability between different chatbots	33
3.7. Frankenbot's System Architecture	34
3.8. Frankenbot Framework Architecture Diagram	36
3.9. Frankenbot's User Interface(Client)	37
3.10. Flow chart to finalize and update JSON output.	44
3.11. Flow chart to find module with highest activation value.	46
3.12. Flow chart for the module's activation function.	47
3.13. Flow chart for the Rasa Intent and Entity Detector.	48
3.14. Flow chart for response generation.	50
4.1. Age stats of participants	53
4.2. Profession related details of participants	54
4.3. Overall impression of the interaction with the chatbot	59
4.4. Participants familiarity with the chat bots like Google Assistant, Apple's Siri etc.	61
4.5. Participants usage of the chat bots like Google Assistant, Apple's Siri etc.	61
4.6. Graphical representation of collected responses for achievement of goals.	62
4.7. Graphical representation of results collected for the users communication with the chat bot.	63
4.8. Graphical representation of results collected for the behaviour of the chatbot.	65
4.9. Graphical representation of results collected for the dialogue assessment.	66

4.10. Graphical representation of results collected for the users personal experience and impression.	68
4.11. Graphical representation of results collected for the chatbot's usability. . .	70
4.12. Description of the word pairs along with the mean values and standard deviation [7].	72
4.13. Average values for PQ, HQ-I, HQ-S and ATT [7].	75
4.14. Portfolio of the results for HQ and PQ of the chatbot [7].	76
.1. AttrakDiff's Questionnaire Page 1 [7]	103
.2. AttrakDiff's Questionnaire Page 2 [7]	103
.3. AttrakDiff's Questionnaire Page 3 [7]	104
.4. AttrakDiff's Questionnaire Page 4 [7]	104

List of Tables

2.1.	A frame showing each slot along with its type and related question [8]. . .	8
2.2.	Dialogue act detection example [9].	11
3.1.	Traditional vs. Frankenbot in terms of nodes and transitions.	31
3.2.	Representation for an Intent, Entity and Context Variable.	41

Listings

3.1. RASA NLU Training Data JSON Format.	35
3.2. JSON Objects for common_examples.	36
3.3. Frankenbot's JSON Structure.	39
3.4. Frankenbot's API JSON Output.	51
1. RASA Pipeline for Frankenbot.	91

1. Introduction

"You all know that chatbots are a new technology altogether. It's like the early age of the Web. Things are still shaky yet growing at the speed of light."

Rashid Khan (Build Better Chatbots)

As manifested by Khan and Das (2018) "*chatbots are a new technology altogether*" [10] which depicts that future belongs to the chatbots. So to cope up with the speed of advancement, it is the need of the time to make chatbots intelligent enough to communicate like humans. However, many states of the art chatbots are developed but still, they have room for improvement. And for this purpose, a lot of research is happening around the globe.

Mostly, already established chatbots can deliver for what they are designed and perform tasks for humans. Chatbots like Google Assistant, Apple's Siri, and Amazon's Alexa are among the most developing virtual assistants and becoming the need of everyday life. Whereas, IBM's Watson Assistant provides a cloud service for developers to include the service in their software and design the chatbots according to their choice and needs. In addition to it, there are several existing developed open source frameworks used widely for this purpose. RASA framework is one of these open source structures to provide ease in performing machine learning stuff and natural language understanding(NLU). Such frameworks play an important role in developing intelligent chatbots in the present era.

The prime purpose of this master's thesis is to develop a framework with a different approach i.e. modular approach as mentioned in Chapter 3 of this document. It assists developers to design their own conversational interfaces by using this framework. As an example, the chatbot named Frankenbot has been developed using this modular framework under the scope of this master's thesis requirements.

The fundamental goal of this study is to introduce a framework and make a contribution to the currently rising field of the chatbots. The advanced modular framework has been developed using Python. It implements a web API to make a request to the server via a user interface. And a JSON response with desired information is returned to display it for a user. Web API has been developed using Python's library known as Flask. The chatbot implemented using this framework is named as Frankenbot. Which

1. Introduction

has been used for experimental purposes and evaluating the research accomplished in this master's thesis.

Frankenbot itself is a detective chatbot that communicates with users and on the basis of their respective answers makes a judgment whether a user is a culprit of a robbery or not.

1.1. Motivation

The concept of conversational agents known as chatbots have been there for decades. But Brandtzaeg and Følstad (2018) highlighted the year 2016 as a revolutionary year for it. And real rationales behind it were rapid growth in the field of artificial intelligence (AI) and a rise in a trend for messenger applications such as Facebook Messenger and Slack etc. [11]. Human-like conversational systems are one of the emerging hot topics nowadays. Retrospectives and advancements in artificial intelligence (AI) and drastic transformation of mindset i.e. humans communicating with some automated agent without being recognized are the main reasons for enhancement of interest towards chatbots. It is very important to make sure that chatbots are intelligent enough to understand user utterances and the semantics in order to communicate as humans do. The other important fact that can't be deprecated is unlike living beings, machines don't need refreshment and can perform 24/7.

The word chatbot is derived from "chat robot". It clearly states that it is an automated agent dealing with natural language user interfaces for data and services provided via dictation or writing. Users can query, command, or converse with the chatbots using regular language in order to get the required content in the form of data or service. As one messaging platform provider Kik, claims on its developer site: "First there were websites, then there were apps. Now there are bots." [11]. And no one can deny the fact that future belongs to the bots as you can easily notice that many companies are cutting out the man force in their customer support sector and replacing it with "Chatbots". With more advancement in these conversational agents, they can be utilized for many other departments at a corporate level.

The existing state of the art dialogue frameworks like RASA [2], PLATO [3], and IBM Watson [1] manage dialogues modeled as a single dialogue tree and contains only single state for all modules where module can be an utterance, response pair or a dialogue tree. It has several disadvantages like a complex tree structure, complex and lengthy transitions between nodes, difficult to keep track of dialogue, dependency on the last user utterance to stay in topic, and difficulty in switching and jumping to and fro between different topics. To overcome these issues there is a need for an invention that has been proposed by means of this master's thesis as explained below.

1.2. Scope of the Thesis

The goal of this master thesis is the development of a framework to rapidly develop conversational interfaces/chatbots.

To overcome the problems with already existing frameworks mentioned lastly, the framework is needed that follows and implements the modular architecture for conversational interfaces. Following new advancements has been added to this approach: (i) Dialogues are not modeled as a single dialogue tree but as multiple modules, (ii) A module can be a single utterance/response pair or a dialogue tree and (iii) The dialogue can have one state in each module instead of only a single state in all modules. These progressions exposed several benefits. Transitions between the dialogue states do not need to be modeled explicitly. Therefore dialogue trees can be simpler. Furthermore, the chatbot can develop a sense of “staying in the topic” instead of choosing the topic simply based on the last user utterance.

1.2.1. Challenges

This section highlights the major challenges to design a chatbot with modular dialogue manager.

- Recognition of NLU based intent and entity(s). Designing a tree structure for groups of intents, entities and assign parent and child node relation among them.
- Combining multiple trees based on user utterances to make sure that chatbot can stay in topic without changing its state to some new topic.
- Implementing a dialogue manager that favors recent modules. It means once the module is triggered, dialogue manager can save it as an active module. There are more chances of the next user utterance to belong to this active module which results in enhancing the performance.
- The modules should be able to create dialogue trees. Every node of the dialogue tree contains an answer followed by the JSON based file structure for storage.
- Implementing the web based frontend with a support of multiple user sessions at the same time, relies on a web API with a single endpoint which receives the user utterance as a parameter and returns the chatbot answer.
- Handling of Natural language processing by picking up the syntax. For example, If we query a chatbot ”what’s the weather?”. It will respond perfectly fine but what if we rephrase it and ask ”Could you please check the weather?” there is a chance of a glitch. These kind of programming problems resides under natural language processing category and are centre of attraction for the companies like Facebook, Google with Deep Text and Syntax Net respectively. [12]

1. Introduction

- When it comes to Machine Learning, it is another fact that can not be declined while designing and developing a chatbot. Efficient programming practices with artificial intelligence(AI) concepts is a key to achieve the best learning for a system to respond correctly [12].
- Choosing a pipeline for RASA's natural language understanding(NLU) to extract the best out of it.

1.2.2. Contributions

This document contains a review of the literature about dialogue managers in chatbots. Moreover, it also explains a software architecture and its implementation in addition to the necessary configuration files and the dialogue JSON file.

The following functionalities have been successfully implemented during the thesis:

- Natural language understanding(NLU) for user utterance performed using RASA to detect intent and entities based on data for what it has been trained.
- Dialogue trees; The modules are capable of creating dialogue trees. Every node of the dialogue tree contains an answer.
- JSON based file structure to store the dialogues.
- The modular dialogue manager. Its detailed functionality and working have been discussed ahead in Chapter 3.
- The NLU based intent recognition has been implemented using the RASA NLU interpreter.
- The chatbot also performs RASA's NLU based entity recognition. All detected entities are written to the whiteboard.
- Dialogue manager is responsible to generate a response for a user utterance based upon the detected intent with the highest confidence. Secondly, if there is an entity key that appears in a bot response then it gets updated with its value stored on a whiteboard. On the contrary, it just gets eliminated in case of no value stored for that key.
- Frontend: A prototypical web-based frontend has been developed as an interface for a user. This web-based interface relies on a web API with a single endpoint that receives the user utterance as a parameter and returns the chatbot answer.
- Multiple users can chat simultaneously using the web-based interface. Their dialogue states, whiteboards, and other session-based variables will not mix.
- The system generates a log file that helps to analyze why the dialogue manager comes to a certain answer for a user utterance and what information and results have been provided by RASA's natural language understanding(NLU).

1. Introduction

- To demonstrate the functionality of the framework a chatbot is implemented using the framework i.e. detective bot named Frankenbot.

Technical Requirements

The framework has developed using Python language. It includes all the above-mentioned functionalities and is usable, which means it is running without any technical bug or error. Lastly, it has no dependencies on libraries with problematic licenses. Only the libraries lie under open source license have been utilized during the implementation of a framework.

1.3. Structural Outline

The current chapter provides a general introduction to the thesis topic. Whereas, Chapter 2 i.e. Foundations and Related Work, discusses the history of the chatbots along with the detailed background. Additionally, it also illustrates the classifications of dialogue systems. Furthermore, the medium of communication, chatbot elements along with their respective tasks and conversational agents applications have been discussed. And finally, the chapter gets closed with an explanation of evaluation techniques.

Moreover, Chapter 3 explains the concept and design of the system. It elaborates on the system overview, architecture, and capabilities of a system. Also, the detailed description of the whole framework and dialogue manager developed in this thesis has been described in this chapter.

Next, Chapter 4 expresses the users' experience for a new approach. It also represents the results gathered by the completion of user surveys.

Lastly, Chapter 5 discusses and provides the analysis for the results of the evaluation portrayed previously in Chapter 4. Moreover, it also summarizes and concludes the work and tasks performed in this thesis. Finally, the chapter and this document get closed with an outlook for future work.

2. Foundations and Related Work

This section formally discusses the state of the art machine learning techniques and strategies that have been used practically for chatbots.

In the current era, a lot of research is happening to make the chatbots to communicate like humans without being recognized as a machine. But in reality, a lot of development is still pending in this regard. Technological elevation using machine learning (ML) and artificial intelligence (AI) is helping this technology to get pushed forward through learning. At the same time, there are emerging challenges and problems hindering the pace of the progress. Many major companies are working to make an ideal agent but still are unable to produce one. This chapter illustrates the initial purpose of introducing the chatbots and why there appeared a need for these agents? The basic and important chatbot classifications and techniques are introduced. In addition to that, the work and research accomplished so far under this area are featured. Let's start with a brief historical background about the chatbots.

2.1. Chatbots History

According to Shawar and Atwell (2007) initially, chatbots were introduced in the 1960s and the key idea behind it was to make the communication feel real like humans [13]. In addition to it, as stated by Zadrozny et al. (2000) to make humans interact with the computers it can only be led by making humans "to express their interests, wishes, or queries directly and naturally by speaking, typing, and pointing" [14]. For the accomplishment of such objectives, the chatbots were introduced [13]. But now their applications are extended just like their scope. Mostly these artificial agents have covered the entertainment sector, information search, and assisting humans in the e-commerce and business sector. But when it comes to replicate human beings it is still one of the biggest challenges in the present. As an understanding of natural language, grabbing feelings, verbal structure, gestures, and body language just like living beings are the main hindrances for a machine to act like a human [15].

Many companies claim for operating their chatbots and doing research on them. As mentioned on the website "botnerds.com" [16], the stats for spring 2017 are as follows:

- Facebook alleges over 100,000 bots on Messenger.
- Twitter claims approximately 48 million bot accounts.
- Kik confesses 20,000 bots on its platform.

2. Foundations and Related Work

- Wit.ai claims 21,500 workers for development.
- Small amount of skype bots are also available.

The next section provides the technical knowledge of the dialogue systems.

2.2. Dialogue Systems

The importance of human-computer interaction has raised drastically in the last few years. It is due to its rising potential towards solving daily life problems especially providing aid in commercial challenges. With the advancement of big data and machine learning, the self-functioning virtual assistant doesn't seem to be impossible. Jurafsky and Martin (2019) classified dialogue systems as (i) Systems designed to complete a task i.e. Task-oriented dialogue agents and (ii) Systems for extended unstructured conversation i.e. Chatbots [8].

2.2.1. Task Oriented Agents

These systems are designed to accomplish a task in some specific domain. The system interprets a message provided by the user, symbolizes it to the internal state, and processes it according to the state of the dialogue. Lastly, the final action is performed on it to produce a response [17]. Common examples of such virtual agents are Apple's Siri, Google Home, and Amazon's Alexa.

Mainly natural language understanding is done using statistical models. On the other hand, some systems still use human-designed pre-defined rules for slot filling and detecting an intent [17]. It not only results in more time consumption and makes deployment of such system costly, but also restricts the operational domain for the system. To overcome these challenges, deep learning played an important role.

Modern task-oriented dialogue systems are based upon the following architectures:

Frame-based Architecture

Jurafsky and Martin (2019) narrates that state-of-the-art task-oriented dialogue systems are composed of frames. The information extracted from the user utterances by the system to identify the intentions of the user is stored in the form of a knowledge structure which is known as a frame. The slot could be a key-value pair of some specific type and a frame is composed of a group of such slots. These slots indicate what information is required by the system. Once these slots get filled then the system can perform some action accordingly [8]. Example of a frame along with the slots and their types and question to fill each slot has been shown in Table 2.1.

2. Foundations and Related Work

Slot	Type	Question Template
ORIGIN CITY	city	"From what city are you leaving?"
DESTINATION CITY	city	"Where are you going?"
DEPARTURE TIME	time	"When would you like to leave?"
DEPARTURE DATE	date	"What day would you like to leave?"
ARRIVAL TIME	time	"When do you want to arrive?"
ARRIVAL DATE	date	"What day would you like to arrive?"

Table 2.1.: A frame showing each slot along with its type and related question [8].

Dialogue-state Architecture

It is the most refined form of frame-based architecture. It is also named as belief-state architecture [8]. Whenever there comes a new user utterance, the NLU component extracts the required knowledge to fill the related slots. Secondly, the dialogue manager includes dialogue state tracker which is responsible for tracking and handling the entire dialogue information and values for each slot of a frame. Whereas, the policy learning component is responsible for deciding on the further action that the system should perform or convey. Once the dialogue policy confirms that what action should be taken by the system according to the updated dialogue state. Afterward, Natural Language Generation(NLG) component takes an action to produce the textual response for the users [17]. Graphical representation of such architecture is shown in Figure 2.3.

2.2.2. Chatbots

Chatbots are considered to be the simplest form of the dialogue systems [8]. The main goal of such systems is to communicate with humans on any topic without restricting it to some specific domain, making them more independent and self-sustainable. Applications of such a system are entertainment or general conversation with meaningful responses [17].

Furthermore, chatbots can also be categorized based on their composition design and methodology used for language processing. Jurafsky and Martin (2019) classified the chatbots into two divisions: (i) Rule-based chatbots and (ii) Corpus-based chatbots [8].

Rule-based Chatbots

Such conversational agents are trained on simple rules to communicate. There are pre-defined actions for which the user is provided with a choice of selection. These conversational agents mainly include questions to which the user can agree or disagree. Other than that they are also capable of detecting a keyword or a group of terms to proceed further and generate a proper response for the user utterance. ELIZA and PARRY are the most common examples of the chatbots that followed the rule-based approach [8].

2. Foundations and Related Work

Corpus-based Chatbots

Unlike rule-based agents, these are capable of learning and getting trained for general dialogue. For this reason, they need a humongous data source for getting trained enough to perform general conversation like humans. As stated by Serban et al. (2018) that a chatbot could need hundreds of millions or billions of words for sufficient training [18]. Once they are operational, these agents are also capable of self-learning, and user responses and utterances can also be used as additional data for training purposes [8]. The dialogue framework implemented under the scope of this master's thesis is capable of getting trained for the provided training corpus but the ability to use users' utterances as training data could be added in the future.

The next section explains the means that could be adapted by the users to interact with the dialogue systems.

2.3. Communication Channels

There are two types of communication channels that almost all the chatbots are using for communication i.e. messaging platforms for text messages like Slack and speech communication platforms like Alexa [19].

Text

These chatbots use text messages for communication. They have an extra advantage as the input and output are in the textual form which is easier to understand, store, and adapt.

Speech

Chatbots with voice or speech feature support is more user convenient and provide a more realistic and natural mean of communication as they take input using vocals and produce spoken output.

Text and Speech

There also exist such chatbots that have the conversational support using both voice and text channels. They provide the user with an option to configure the way of taking input and expressing the output [19].

The dialogue framework developed in this master's thesis only supports textual input for now. Support for interaction via both manners (text and speech) could be integrated into the next version of the system.

2.4. Chatbots Components and Tasks

Chatbots consist of various components and each component is responsible for performing a specific task. Usually, a chatbot is composed of the following components:

- Speech to Text Converter.
- Natural Language Processor.
- Response Generator.
- Knowledge Base.
- Dialogue Manager.
- Text to Speech Synthesizer.

As already mentioned in the previous section that the conversational agent developed in this master's thesis is only capable of taking input in textual form. The very first component i.e. speech to text and the last one i.e. text to speech are only applicable if there is an involvement of speech. So these two elements are overlooked in this research. Other than that rest of the components are discussed below.

Starting with natural language processing(NLP), it plays an important role in dialogue systems to understand the semantics of the inputted text and is also responsible for detecting the intents and entities. Moreover, intent and entity classification could be done using a variety of techniques that vary from Simple Keyword Extraction to Bayesian Inference. Afterward, the response could be generated using retrieval-based and generative-based methods [4].

Once the response has been generated, it is passed to the dialogue manager which is responsible to manage the dialogue as a whole. It decides how and what to respond to the user. Initial technical approaches used for dialogue management had been characterized as a dialogue grammar, a plan-based approach, or a collaborative approach [20][21]. Currently, advanced approaches known as statistical methods are helpful for the enhancement of strategies for dialogue management [22]. These models can get trained using substantial dialogues providing robustness to the system [22]. Knowledge management, dialogue state, and response handling are the fundamental tasks for a chatbot. The system's meta-communication (confirmation, clarification, repair, and recovery) strategies are also the major attributes of the dialogue manager [23].

2.4.1. Natural Language Processing(NLP)

It is an essential chatbot element. NLP is a technical process for a better understanding of a user's utterance. It makes the chatbots to comprehend the semantics and the user intention behind the conversation. It is used to add more humanly characteristics to a

2. Foundations and Related Work

chatbot and to make the dialogue feel more realistic. By adding NLP to a chatting agent, it becomes capable of learning and getting trained for any type of query or command provided by the user. Some methods for mastering the semantics of the text and for deep learning are discussed underneath.

Dialogue Act Detection

Determining the function of the text whether it is a question, command, suggestion or an offer is known as dialogue act detection. It is one way of deriving semantics from natural language [24][9]. For a better understanding of it, an example has been shown in Table 2.2.

Speaker	Dialogue Act	English
A	Conventional-opening	Hello!?
B	Conventional-opening	Hi Peter!
B	Statement	It's me, Michael.
B	Question	How are you?
A	Conventional-opening	Hello Michael!
A	Statement	Very well.
A	Question	And you?
B	Statement	I'm well too.

Table 2.2.: Dialogue act detection example [9].

Intent and Entity Classification

It is another way of understanding the meaning of the text represented in the form of natural language. Domain-specific dialogue acts are known as intents. An intent represents the user's real intention and goal of what he/she is trying to convey. Whereas, entities are modifiers for an intent. Entities could also be referred to as slots. Based on the detected intent the user can be directed further to available dialogue options [25].

Using NLP, a conversational agent can classify an intent and entity(s) in a user utterance. The main idea behind it for a chatbot is to detect the purpose of a talk that the user is trying to convey. For identification, the multi-classification problem is usually solved by labeling the utterances according to the possible user intentions and providing some relative names to them. Furthermore, there exist some techniques to complete this task varying from simple keyword extraction to Bayesian inference. These methods are meant to identify the user's request with the help of various messages. Well known LSTM[26] networks are used to perform this task due to their high performing and delivering capacity [27].

Information Extraction

Using this method for grasping the semantics, the unstructured text is transformed into structured grammar and passed to dialogue manager for further processing [9].

As stated by McTear et al. (2016) that the initial step for information extraction is to break down the sentence or utterance into tokens. This process is named as tokenization. It is not an easy task to perform due to several challenges:

- Phrases consisted of multiple terms e.g. the United States.
- Abbreviations e.g. U.S.A
- Contractions e.g. I'm

There are several techniques available to perform tokenization as mentioned in [9]: (i) Bag of words, (ii) Latent Semantic Analysis, (iii) Regular Expressions, (iv) Part of Speech Tagging, (v) Named/Relation Entity Recognition, (vi) Semantic Role Labeling and (vii) Creation of Grammatical Data Structures. There also exist some statistical methods to extract the semantical information for the text as stated below:

- Hidden Vector State Model [28].
- Support Vector Machine Model [29].
- Conditional Random Field Models [30].

Deep Learning Techniques

In this section, some commonly used deep learning will be explained that are used for Natural Language Processing(NLP). Word Embedding and Recurrent Neural Networks are frequently used techniques for deep learning [31].

Word Embedding Word embedding technique is used for transforming words in the form of vectors [4]. These mapped vectors can be directly featured in machine learning algorithms. Different methods have already been introduced to exercise this operation. These approaches varies from simple vector count to statistical methods such as Word2Vec [32], GloVe [33] and Skip-gram Model [34].

Recurrent Neural Networks Recurrent Neural Networks(RNNs) are the neural networks specifically designed for sequenced data. More precisely, such networks work recursively having an internal state denoted as C_t at a specific time t . Which is passed as an input to the neuron as upcoming time-step and it outputs a value h_t based on C_t at that time-step. But while implementing a simple RNN there comes gradient problems already proven by Mikolov et al. (2012) in “Understanding the exploding gradient problem” [35]. So to overcome this problem various revised methods have been introduced. According to [4] most commonly used ones are mentioned below:

2. Foundations and Related Work

- Long Short Term Memory Units [26].
- Gated Recurrent Units [36].

2.4.2. Response Generation

Once the NLP component is done with its part then the gathered information is passed to the response generator. It is responsible to generate some meaningful responses to make the communication effective. So, for this purpose, a bot must reply coherently according to the context of the conversation. This challenging task is executed using the following pair of modules [4]:

- Module to provoke a list of competitive responses.
- Module to select the most relevant reply based on some weighted value or specific metrics.

Moreover, there exist the following methods that are used practically to generate a response.

Retrieval-based Methods

These methods consist of a humongous database containing successive responses. Chat-bots developed employing such techniques contain the data structures like directed graphs and trees. The bot is competent enough to fetch the best response out of the numerous collection of responses. It is done by fetching the responses for the identified intents and entities related to the information provided in the form of a message. The information provided can usually be a regular expression searching for a specific structure of sentences or any output from the machine learning algorithm [4]. Whereas, such methods cannot self generate a natural language response like humans. Responses are either fed manually or via an already available knowledge base.

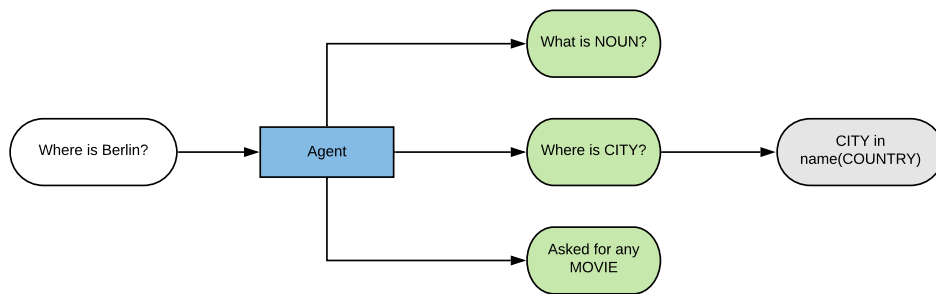


Figure 2.1.: Retrieval-based Methods Example [4]

The primary and fundamental step to achieve the retrieval-based method is message and response matching [17]. The basic purpose of matching algorithms is to minimize the

2. Foundations and Related Work

semantics difference between the responses and the messages [17]. Two such techniques are mentioned below:

Single Turn Response Matching In this type of matching the only responsible factor for response generation is the message itself. Message context and successive answers are represented as a vector [17].

Multi Turn Response Matching Unlike single turn matching, current utterance along with all past messages are responsible for producing the most suitable response that best matches the overall context of the conversation [17].

Generative Methods

Unlike retrieval-based methods, agents composed of generative models don't need pre-generated responses. They are capable of producing a new response based on the user's utterance using generative models. For generating a suitable response the model must be trained enough. Unfortunately, the performance of these models is still not sufficient enough to attract the companies due to various restrictions forced by the corporate [4]. The basic method for neural generative models has been briefly discussed below:

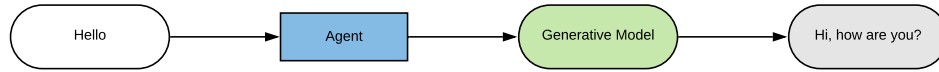


Figure 2.2.: Generative-based Methods Example [4]

Sequence-to-Sequence Models Neural networks are being used by these models to symbolize dialog history and to produce useful responses [17]. The systems based on these models don't need much knowledge base and pre-defined rules to generate natural language responses.

Furthermore, Jiliang et al. (2017) mentioned some challenges that need to be discussed for generative-based methods:

Dialogue Context; To keep track of the current state of chatbot and recent conversation, it is compulsory to take care of the dialogue history and recent user utterances. It helps in understanding the context of dialogues to generate an appropriate response. Recurrent Neural Networks(RNN) are mainly used for this purpose.

Response Diversity; To handle response diversity is a challenging task in the systems based on sequence-to-sequence models. As they can provide irrelevant, balanced, or completely related answers having some meaning. Inverse Document Frequency(IDF) is

2. Foundations and Related Work

mainly used for this purpose.

Topic and Personality; The topic and personality of the dialogues is another factor to enhance their diversity. By acquiring the fundamental characteristics of dialogues, diversity can be increased and also it leads to gain consistency.

Knowledge Base; For making the virtual assistant perform as humans do there is an essential need of a knowledge base. Not only using outside knowledge base we can cut the informational difference between machines and humans, but can also plant sensibility to artificial conversational agents.

Interactive Learning; Eventually the main objective of a dialog system is to be smart enough to do self-learning during an interaction with a user.

Evaluation; In the end, the final step is to assess the provoked response quality generated using response generators. Task-oriented dialogue systems can be graded using handcrafted methods like ratings from the user. Whereas, for non-task oriented dialogue systems it is a bit challenging task to evaluate them. METEOR, BLEU, and ROGUE are some word overlap metrics to weigh the produced response quality [17].

Hybrid Methods

Hybrid approaches by combining both the above-mentioned methodologies have recently been introduced. If response generation fails using retrieval-based methods then it should be produced using generative-based methods. Retrieval based systems give more accurate results but that could be slow and vague [17]. Whereas, neural generative systems can respond rapidly with the chance of giving useless results [17]. So by unifying both of these methods, the performance can be enhanced significantly. More studies about these methods can be found in [37].

2.4.3. Knowledge Base Designing and Management

The intelligence of a chatbot is directly proportional to the knowledge available for its training and related processing [9]. The knowledge base includes the bulk of training data for training of generative-based systems and the cluster of responses for retrieval-based conversational agents. Knowledgebase designing and management means how mature is the data gathered for training and managed to construct a corpus for the statistical dialogue systems. Making a chatting agent interact like humans also depends upon how well the corpus is designed and managed.

Cahn (2017) illustrates that at the beginning of this century humanly designed data corpuses served the purpose on a large scale. But over time, more advanced techniques have been introduced for collecting the data to design a knowledge base for chatbots.

2. Foundations and Related Work

Currently, data scraping technique¹ has provided ease by enabling data auto-collection over a large number of dialogues available on the web. By auto importing such a large amount of information, the knowledge corpora for the chatbots could also be auto-generated [9].

In present, Web API calls and optimized requests to the databases are extensively used to perform knowledge management [4]. Whereas, knowledge bases could also be represented as impressive graph-structured ontologies [38].

There could be many data sources available for collecting data to manufacture knowledge base for chatbots. Some of them discussed in [9] are:

Human-Annotated Collections

Human-annotated dialogues could be modified to Artificial Intelligence Markup Language(AIML)² and other formations suitable for chatbots.

Discussion Forums

Scraping online discussion forums can be helpful to serve the required purpose.

Email Messages

Gathering data using email conversations could also result in the designing of a knowledge base.

2.4.4. Dialogue Management

Once the response generator component is done with its part then it passes the response to dialogue manager. It is responsible to manage the whole dialogue and checks according to the current state of dialogue whether to respond or not and what to respond to the user. Detailed architecture for a dialogue management system and data stream has been displayed in Figure 2.3. Components involved in dialogue management have been already explained before in the section "Dialogue-state Architecture". In addition to that, a dialogue manager is also liable for applying communication strategies along with the shaping of generated response to grant it real human-like sensation [9].

Communication Strategies

Communication strategies firstly include the dialogue initiative plan. It is the basic elements that should be taken care of while designing the dialogue manager. The main goal for this interaction plan is to grant the leading role to the actors involved in a

¹<https://www.targetinternet.com/what-is-data-scraping-and-how-can-you-use-it/>

²<https://en.wikipedia.org/wiki/AIML>

2. Foundations and Related Work

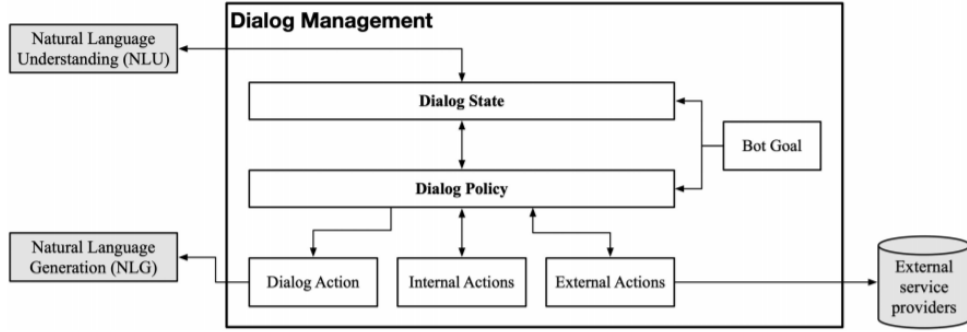


Figure 2.3.: Dialogue management Architecture and information flow [5].

dialogue and that could be the system, user, or both. McTear et al. (2016) classified it into the following categories [9]:

- User-directed initiative; It happens when the user leads the communication by posting any sort of query and the system has to respond. Pros: User is not restricted and more humanly conversation. Cons: The system might not be able to respond effectively to the user's query that is not relevant to the system. It means the system can be misguided easily.
- System-directed initiative; The system is provided with the leading role to direct the conversation while the user is responsible for replying. Pros: Fewer chances that the system could receive deluded inputs. Cons: Limit the users and less humanly.
- Mixed initiative; In this approach, both are provided with the dialogue initiating rights. Pros: More real and humanly. Cons: Possibility of diverged dialogue.

Furthermore, another important strategy that it should adopt is the confirmation approach to make sure that the parties involved in a conversation are on the same page. McTear et al. (2016) discussed two confirmation strategies:

- Explicit confirmation; System confirms again by restating the user's query as a response. Pros: Guaranteed appropriate response. Cons: Greater time complexity.
- Implicit confirmation; System confirms by adding the user's query in a response along with the next query. Pros: No time wastage. Cons: Chances of irrelevant response.

McTear et al. (2016) also pitched the following examples for better understanding of confirmation strategies:

- Explicit Confirmation - User: "I want to go to New York." System: "You want directions to New York?" [9].

2. Foundations and Related Work

- Implicit Confirmation - User: "I want to go to New York." System: "What type of transportation do you want to use to get to New York?" [9].

Domain

A specific dialogue field refers to a domain. It is composed of particular actions, states, and intents required in a particular dialogue domain [5].

Response Manipulation

All responses generated using techniques mentioned before containing confidence values assigned to them. Higher the confidence, the higher the receptiveness of the response. In case of responses with lower relevancy and confidence values system should be able to respond effectively. There are the following techniques highlighted by Yu et al. (2016) used to serve this purpose [39]:

- Topic Shifting; Instead of replying with the answer having a lower confidence index, an agent tries to divert the user by switching the context to some common interesting topic like sports, weather, or any other engaging topic. It could cause irrelevancy but users find it more logical as compared to the inappropriate response with lower confidence.
- Throwing Open Queries; Replacing simple fallback messages such as "I didn't understand you" etc. with some meaningful content like "Sorry, I can't say anything about it. Could you tell me something more interesting?". As both of the statements are serving the same purpose but the second statement is more helpful for dialogue continuation.
- Cracking a Joke; Instead of replying with an irrelevant response, a chatbot could crack a joke. It will not let the conversation stop.
- Additional Information Inquiry; In this technique chatbot responds with a message and asks the user to elaborate or devise his/her query again.

Dialogue Recovery Principles

Dialogue manager has to implement the following dialogue recovery principles stated by [40]:

1. Disambiguation; It refers to the ability of the dialogue system to clarify the ambiguous user utterances. For example, input: "Was Amadeus nominated for an academy award?" and response: "Is Amadeus the movie's title?" [9].
2. Relaxation; It relates to the dialogue system's capability to respond flexibly. For example, input: "Are there any flights today" and response: "No. Would you like me to find one tomorrow?" [9].

2. Foundations and Related Work

3. Confirmation; Dialogue system’s ability to confirm before executing critical tasks for the users. For example, response: “you want me to book you the 7 am flight from Philadelphia to San Francisco?” [9].
4. Completion; Capability to enquire about the missing information before taking any action. For example, input: “I would like to book the 7 am flight to SF” and response: “Which airport are you flying out of?” [9].

Dialogue Management Techniques

Techniques used for dialogue management can be classified as (i) Handcrafted (rule-based) approaches and (ii) Probabilistic (statistical) approaches [5].

Handcrafted Approaches As it could be apprehended from the name of the approach that the dialogue managers implementing such a technique need the set of pre-defined rules provided by developers and domain professionals. These rules are further used to compose the system’s state and policy [5].

Finite State Machines; Dialogue managers consisted of finite-state automata, hold a single state for the conversation [5]. And with each leading transition, the state gets updated [5]. Such dialogue managers have overall control over the communication by limiting user choices.

Frame-based Dialogue Management; Flexibility factor can be enhanced by adding the data model for slots tracking to the finite state machines. Such an approach is named as frame-based dialogue management [5]. Slots could get filled irrespective of any dialogue sequence empowering the user-directed initiative.

Model-based Dialogue Management; More practical approaches could be designed by appending user model, conversational model, or any related model with previously discussed data model [5]. Harms et al. (2019) provided an example of such a model and named it as an information state model. It consists of a state which is responsible to keep track of user goals and notions. It is also expected that the developer has been provided the state updating mechanism according to the possible dialogue acts [5].

Probabilistic Approaches Unlike handcrafted approaches, statistical approaches are more pragmatic and possess the ability to learn the rules on their own via real conversations.

Example-based Method; It was an initial statistical method that made the dialogue managers capable of getting trained using massive data collection to respond efficiently. It checks for the similarity between user utterance and the example in the data available for training purposes and selects the best response from this training corpus [5].

2. Foundations and Related Work

Partially Observable Markov Decision Process(POMDP); As explained by Harms et al. (2019), dialogue managers following such processes initially mark the dialogue state as unperceived. Secondly, there exists a state distributed over all possible states named as belief states. Whenever there comes an input it is evaluated and observed for all the possible states within a system. This observation yields the most fitting state for the user input. In the end, dialogue policy component practices reinforcement learning to take an action according to the current belief state [5].

Neural Networks; Presently, neural networks are exercised on the dialogue managers. The main idea behind it is to omit the limitations set by handcrafted methods [5]. As such networks are composed of memory units just like humans and they are capable of reading and writing the knowledge into their memory segments. They receive the information through old and recent input provided to them.

Hybrid Approaches Both of the lastly mentioned dialogue management techniques have been combined and practiced to develop data-driven conversational agents [5]. Rasa Core³ is an example of such dialogue agents. And the framework developed in this master's thesis is also based upon this approach.

Graphical representation of the classifications for the dialogue managers along with respective existing tools has been shown in Figure 2.4.

Challenges

There exist few demanding factors that need to be highlighted while talking about dialogue management.

Coding Strategies; Strategies used for the coding purpose are most often independent of a dialogue structure. The coding scheme and modeling strategy are responsible for the amount of information that can be delivered using a dialogue.

Dialogue Structure; Logical dialogue designing supporting state structure and transformations. Data structures like trees and graphs can be used for this purpose. While graphs can push the structure more towards complications.

Error Recovery; There is a high risk of task failure but it should be made sure that dialogue must go on. It should be highly prioritized that the dialogue manager must be able to catch the errors and handle them properly. This means it should be able to recover from faulty situations using different techniques as mentioned previously under the heading "Dialogue Recovery Principles".

³<https://legacy-docs.rasa.com/docs/core/>

2. Foundations and Related Work

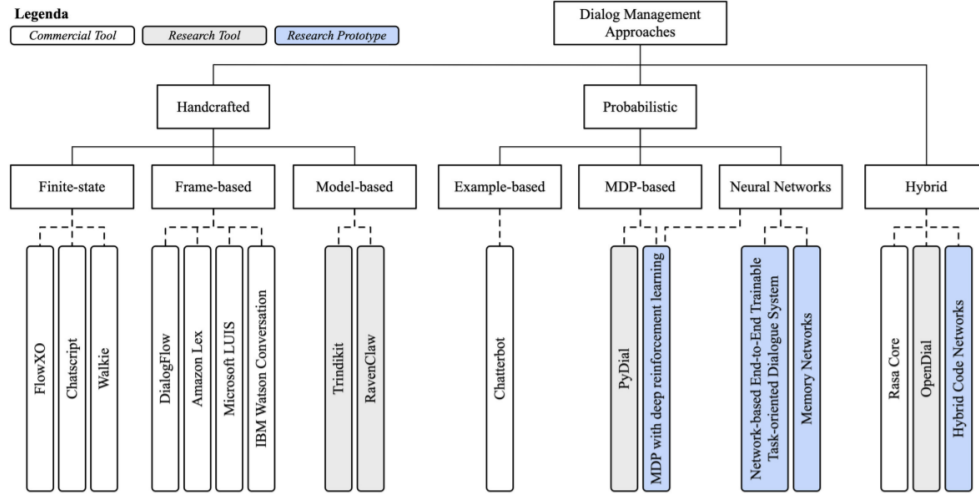


Figure 2.4.: Classification of Dialogue Management Approaches. Top most level shows three different approaches: handcrafted, statistical and hybrid. Next step illustrates different methods associated with each approach. Lastly, practical tools (white boxes), research tools (grey boxes), and research prototypes (blue boxes) are defined [5].

2.5. Applications

Cahn (2017) categorized the applications of the chatbots into the following two domains:

2.5.1. Virtual Personal Assistants(VPAs)

Virtual conversational agents are getting used extensively to assist the users at a personal level. Agents like Google Assistant, Apple’s Siri, Microsoft’s Cortana, and Amazon’s Alexa are playing an important role in this domain [9]. Such conversational agents seek command from the user and take steps accordingly to fulfill the desired task.

2.5.2. User Domain-Specialist Bots

Such bots are specialized in some user-specific domain for what they have been trained and provide instructions on user request. Some domains illustrated in [9] are: (i) Transportation, (ii) Dating, (iii) Meditation, (iv) Fitness, (v) Weather and (vi) Medical.

2.6. Evaluation Methods

Once the dialogue system is developed and ready to be launched, it must be undergone through some evaluation techniques. It never has been an easy task to evaluate dialogue systems. Usually, its complexity depends upon the criteria of what to evaluate and how

2. Foundations and Related Work

to evaluate it. In addition to that, assessment is also dependent on different features. The ability of a system to auto-recover itself in case of any error is known as robustness. It should be assessed beforehand during the implementation of a system. The other essential thing to evaluate is the effectiveness of a system that is, how effective, acceptable, and friendly a system is for managing a dialogue. According to [20], there are two of the following methods already being introduced i.e. Quantitative and Qualitative Approaches.

2.6.1. Qualitative Methods

It can be inferred from the name that such methods are used to evaluate the quality of the conversational assistant. These methods involve users to get the mission accomplished. So users' opinions play a major role to make assessment decisions for a system.

For carrying it out, once the users have adopted and utilized a system, they are provided with a questionnaire to fill or an interview can be conducted for them with different questions regarding the performance of a system. Which also includes the questions about performance, natural behavior, user-friendliness of the system, and how well it performed under certain circumstances. It also varies from person to person that some persons respond positively whilst others can provide negative impressions. Even though the system is the same based on different user preferences. So, it is better to make a user well familiar with a system first and then ask him/her to give an opinion.

Some interviews had been conducted by Dybkjaer in 1995 after making the users familiar with a system that can be found in [41].

2.6.2. Quantitative Methods

To assess a dialogue system quantitatively, already two techniques have been introduced so far i.e. black box and glass box.

Black Box Testing

It is based upon the input and the respective output without caring about the design and other internal structure of a system. It is just an outer level or top-level evaluation and can be performed by the end-user.

Glass Box Testing

In this type of assessment, the system's internal individual components can be evaluated. And for this purpose, a developer must have the data from the past so that the current output of a component can be compared with the previous one. Such a comparison can provide one with information about the accuracy of a virtual dialogue agent. Such testing can't be performed using a black box technique as there is no trustful data available for comparing a dialogue. [20]

2.6.3. Cross-systems Comparison

In addition to the above-mentioned evaluation techniques, there also exist some other methodologies to evaluate the dialogue management systems such as objective performance evaluation. It is recommended to perform an objective evaluation for the dialogue to make its performance compared with other management systems. Also in the present era, there are several states of the art chatbots available like IBM Watson, etc. One can also easily evaluate self-created chatbots by doing comparison after designing the same dialogue on any other state of the art virtual assistants.

2.6.4. Evaluation Metrics

Also, there are following evaluation metrics used to quantify the performance of a chatbot according to [42]:

- Dialogue efficiency metrics in terms of matching type.
- Dialogue quality metrics based on the response type.
- Users' satisfaction assessment metrics.

2.6.5. Standard AttrakDiff Tool

AttrakDiff is a standardized tool designed to measure the attractiveness(usability and design) of an interactive product [7]. It provides the results gathered about the hedonic and pragmatic quality of the system. Both hedonic, as well as pragmatic qualities, have a great influence on the attractiveness of the system [6].

As shown in Figure 2.5 that attractiveness is based upon pragmatic and hedonic qualities of the system. Furthermore, hedonic quality is divided into two categories in the current version of the AttrakDiff: (i) Identification (HQ-I) and (ii) Stimulation (HQ-S). Pragmatic quality(PQ) includes the usability of the system i.e. the system delivered for what it has been designed. And as a consequence of PQ, user satisfaction gets collected [43]. Whereas, HQ-I is associated with the way products communicate important personal values as compared to other relevant products. Moreover, HQ-S is an extent to which a product is perceived as challenging or novel.

2. Foundations and Related Work

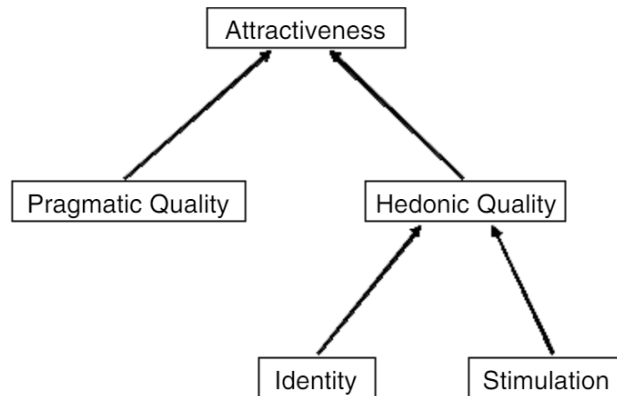


Figure 2.5.: Main Concept of the AttrakDiff's Model [6].

The next chapter illustrates the system architecture and overview of the chatbot named as "Frankenbot" along with its capabilities.

3. Frankenbot: System Overview and Capabilities

In this chapter, you can find all the details about the implementation of a framework utilized to design the bot known as "Frankenbot". Additionally, there is a complete description of the system's architecture, features, and abilities of the framework.

If you are wondering where does the name "Frankenbot" comes from and why? So firstly, let me remove your confusion. It is derived from the fictional character known as "Frankenstein". It was first introduced in a novel written by Marry Shelley in 1818. Later on, after getting the hype it was promoted using different media sources like films, T.V. series, and also adopted by the gaming industry. Furthermore, the most well-known edition for its representation was the movie renowned by its original name released in 1931 [44]. In this movie, Frankenstein was pictured as a haunted scientist who loved to dig the graves of humans and used to create new living beings by reassembling their expired body parts [45]. Likewise, Frankenbot is also a composition of different components as explained below in detail.

3.1. Why Frankenbot?

Currently, there are many states of the art dialogue frameworks available in the market like Rasa, Plato, and IBM Watson. But the question rises why Frankenbot and what makes it different from other well-known frameworks. There exist the following challenges that still need to be addressed.

Firstly, the question raises in mind that, is it possible to design a chatbot that is composed of several small chatbots? Secondly, what if the answer is yes? And one can build a giant chatbot using tiny virtual assistants then how can be the components and modules of a chatbot can be reused? Thirdly, how to make a platform-independent chatbot? In addition to that, another problem is the re-usability of a dialogue between different chatbots. Furthermore, there is another complication that can not be ignored is the rigidity handling for a chatbot. This proposes to inject the flexibility to a chatbot so that it can understand the dialogue consisting of multiple topics without displaying any alert message like "You are currently in the middle of the current dialogue. Are you sure to abort it?". Lastly, the biggest challenge is to design a chatbot that is capable of staying on the topic. This means it should be able to save the current dialogue state for each user instead of just responding with an answer that matches the user utterance

3. Frankenbot: System Overview and Capabilities

irrespective of the current context in the conversation.

For the above-mentioned challenges, there comes the Frankenbot to rescue and address them well. Additionally, the research completed under this master's thesis is an important step towards integrating different technologies. These technologies could include statistical dialogue managers, question answering, and slot-filler. The product implemented in this study provides the preparatory steps for it.

3.2. Frankenbot: Experimental Chatbot

The main idea behind the development of Frankenbot is to attest to the study of the modular framework implemented in this master's thesis.

3.2.1. Domain

The demo chatbot has been designed using entertainment mechanism so that the participants do not lose interest during communication. For this purpose, a detective conversational agent was designed to solve a robbery case. The task assigned to Frankenbot was to ask the user different questions to come up with a decision whether a user answering to those questions is a culprit or not. Whereas, it has been expected from a user to answer the questions or talk generally about the corona virus and its stats, humor, gossips, and other related questions about bot's profile.

3.2.2. Design

Frankenbot has been designed to handle the answers for the questions thrown by it to the users. Secondly, it also has to respond efficiently in case of general conversation. To cope up with these challenging situations, it is provided with two modules. One module for detective dialogue and the other for general conversation. Both modules are implemented using a dialogue tree. Each dialogue tree is composed of interrelated child nodes. And all nodes collectively forming a tree are set to be responsible for containing all the relevant information about the specific module, intents available, and corresponding responses.

3.2.3. Training for NLU

RASA NLU component has been adopted for the training of Frankenbot to understand the messages well. Additionally, it has been also used for detecting intent and entities in user utterances along with their respective confidence values. It has been discussed thoroughly under Section 3.5.1.

The detective model has been trained using 115 examples including 6 distinct intents ('#affirm', '#bye', '#robbery_time.info', '#purchasing', '#greet', '#negation'). Whereas, the general talk model has been trained by means of 205 statements including 9 distinct

3. Frankenbot: System Overview and Capabilities

intents ('#gossip', '#psychology', '#humor', '#emotion', '#coronaStats', '#botFood', '#general.talk', '#botProfile', '#corona'). The json data provided for training can be found in Appendix F.

3.2.4. Response Generation

Lastly, the response is generated based on the detected intent with the highest confidence for a user utterance among all the available modules. Moreover, Frankenbot also checks for the current dialogue state and considers different parameters like the last active module and node for each user before responding. And for each intent, there exists a list of responses. And a response gets selected based on the mode assigned to it which can be either random or sequential.

3.3. Frankenbot as Concept

This section highlights the theoretical functioning of the Frankenbot. Starting from the web-based interface designed for the users to provide input in the form of a text message. This input is passed to the chatbot by making simple get request using Web API. Once it is available to the chatbot for further processing then actual operation gets started.

Initially, the message is passed to a dialogue manager along with the session information and current dialogue state for a user. The dialogue manager holds the complete dialogue structure in the form of the dialogue trees. The number of dialogue trees represents the number of modules contained within a dialogue manager. Furthermore, each dialogue tree is composed of the tree nodes containing the modular information about the specific sub-state of a dialogue along with the intents and comparative responses.

Furthermore, a dialogue manager tries to find out the intent with the highest confidence value for the user utterance. For this purpose, all nodes of available dialogues trees are traversed. Each module contains a separate trained RASA NLU model and intent detection for response generation. A user utterance is processed through all of the generated NLU models to find out the intent and entities for input including the confidence values assigned to them. Later on, this information gets shared with a dialogue manager. It is responsible to keep a check for the highest activation value out of all the intents present in all modules. Additionally, it also keeps track of the module which includes the most confident intent. It also updates the node's activation information about the specific sub-state of a dialogue.

In the end, this information is delivered to a bot to generate a response based on the most confident intent and current state of dialogue for a user. Which is passed to the client as a response to an initial get request made using Web API by a user.

Session information for each user also gets stored along with the last active dialogue

3. Frankenbot: System Overview and Capabilities

state for that specific user. Additionally, context variables are saved to make the response and dialogue more humanly.

3.4. Frankenbot: Approach

The Frankenbot is a chatbot composed of multiple chatbots. The design composes modules that work together to form a chatbot. General graphical representation is shown in Figure 3.1.

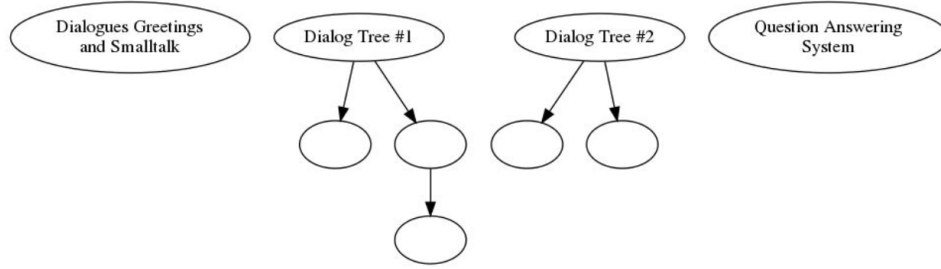


Figure 3.1.: Frankenbot Overview

As displayed, the example is composed of two modules: (i) "Dialogues Greetings and Small Talk" and (ii) "Question Answering System". And each module includes a dialogue tree demonstrating a complete dialogue structure for the respective module. So, "Dialogue Tree #1" belongs to the greetings module, and "Dialogue Tree #2" is linked with the Q&A module. The module and dialogue tree has been explained in the section below.

3.4.1. Module

A module can provide the answer to a user's utterance. Therefore it requires an activation function Z that maps the current user utterance and the system state to a real number.

$$Z : utterance, state \rightarrow R$$

For every user utterance, the dialogue manager calls all activation functions and chooses the model with the highest activation function. This module can then generate the answer. The module itself can be anything. It can be a classical intent-based system. Other systems are also possible. Some common examples of the modules are as follows:

- Bag of request/response pairs [46].
- Waterfall Dialogue [47].
- Slot-filler Dialogue [48].

3. Frankenbot: System Overview and Capabilities

- Dialogue tree [49].
- Question and Answering Dialogue [50].
- Knowledge Graph [51].

As a theory, a module can be more than a dialogue tree. Other systems (question answering, neural systems, etc.) can also fit in this framework, as long as they implement an activation function but it still needs to be tested.

The classic slot-filler/single dialogue tree-based architecture is a basic possible module of the Frankenbot. Therefore the Frankenbot is at least as powerful as the slot-filler.

Traditional Dialogue Tree

One of the data structures used to represent a dialogue is known as a dialogue or conversation tree. Usually, they consist of some sort of data stored in the hierarchical nodes. These nodes are also used to demonstrate the current state of the dialogue by pointing to the current node which has been processed lately. Furthermore, there exist to and from relations between all the nodes. So this approach acts more likely as Simple Directed Graphs ¹. Just, for example, consider a dialogue with two dialogue sub-trees as shown in Figure 3.2. In a traditional system we need to model:

- The transitions from the root node to the sub-tree.
- The transitions inside each sub-tree.
- The transitions from each node to the root node(dashed lines: “I want to cancel this dialogue”).
- To enable transitions between the sub-trees they would need to be modeled explicitly. In the example, this is omitted due to the complexity of the graph. Also, it is not possible to jump directly to the ”subtree #2” while being currently somewhere in ”subtree #1”. Switching is only possible by following the available path to it.

3.4.2. Modular Architecture

There are various benefits for using modular architecture. Utilizing this architecture many features can be unlocked. Usability and performance can also be enhanced as mentioned below:

¹https://en.wikipedia.org/wiki/Directed_graph

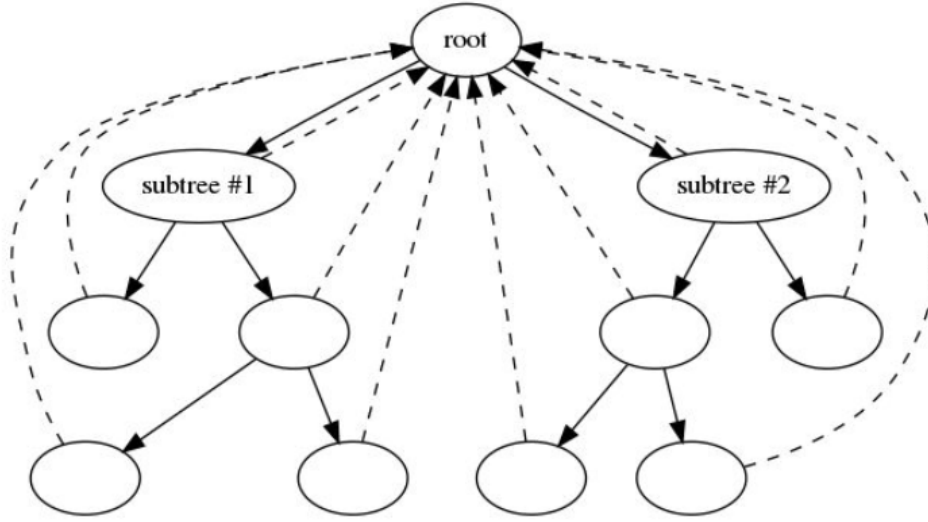


Figure 3.2.: Traditional dialogue tree containing two sub-trees.

Simpler but robust dialogue trees The same dialogue as mentioned above but with the Frankenbot’s framework modular architecture is shown in Figure 3.3. As you can notice the dialogue tree is less complicated and more powerful.

- Jump back to the root node is not necessary.
- Switching between trees is possible without explicit modeling.
- Going back after switching is possible without explicit modeling.

Unification of Smaller Chatbots In this framework, all modules remain independent of each other. They are not tied together. Using this architecture the complexity of larger chatbots can be declined by dividing them into smaller liberated bots. In other words, the complexity of larger chatbots could be reached using the divide and conquer rule. This modular architecture provides a better framework for larger chatbots to grow. It also provides more customization options to a bot as one can easily add and remove any module from any bot at any time without caring for its dependencies.

Modules Usability Modules can be reused within the same chatbot or different chatbots:

- **Usability within a Chatbot:** Modules can be reused within a chatbot. Just consider an example as shown in Figure 3.4.

3. Frankenbot: System Overview and Capabilities

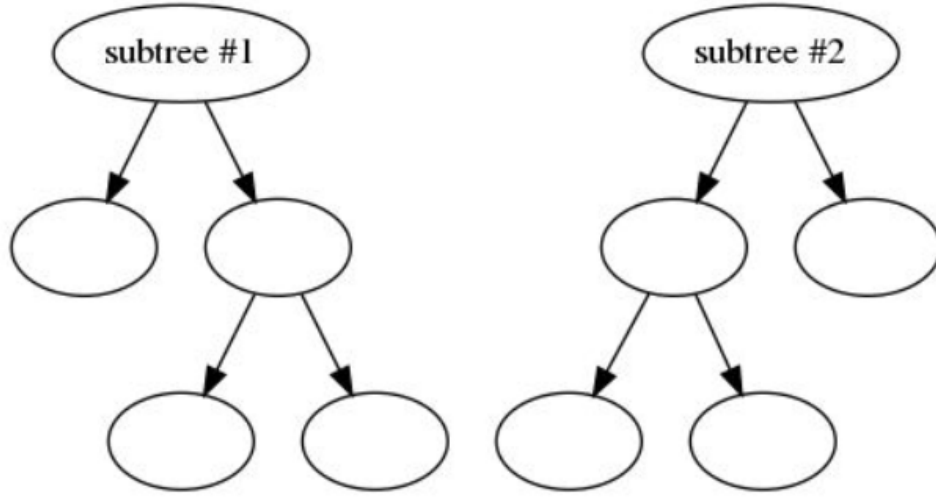


Figure 3.3.: Dialogue with two sub-trees using Modular Architecture

Imagine a chatbot from the smartphone support domain. Two dialogues require how to find out the model of the smartphone. The smartphone model will be stored as an environment variable. Transitions with a diamond tail are only possible when this environment variable is set. The transition with the dashed line does not leave the current node. It only activates the node in the next tree.

The same dialogue graph with a traditional dialogue tree is shown in Figure 3.5 which has several restrictions and disadvantages. The module should be defined twice, more explicit relations needed to go back from one state to another and the dialogue trees are more complicated. Also the number of nodes and transitions will be higher as considered to Frankenbot's architecture resulting to higher processing as shown in the Table 3.1.

	No. of Nodes	No. of Transitions
Traditional	21	26
Frankenbot	15	14

Table 3.1.: Traditional vs. Frankenbot in terms of nodes and transitions.

- **Usability among different Chatbots:** Usability can be enhanced by using

3. Frankenbot: System Overview and Capabilities

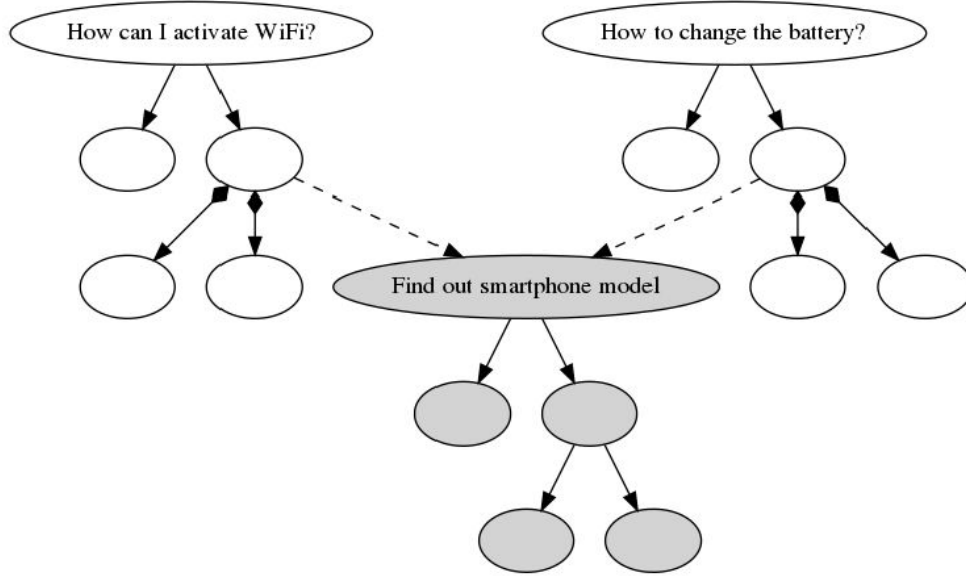


Figure 3.4.: Module usability within a chatbot

modular architecture. Multiple chatbots can share common modules. The module needs to be defined only once. Changes in the module will automatically be integrated in all chatbots. Figure 3.6 shows visuals for better understanding of it.

Sense for Staying in the Topic The dialogue manager chooses the module with the highest activation function. But we can also add a term to stay in the topic:

$$Activation(module) = Z(utterance, state) + History(module)$$

Z is the activation function. $History(module)$ is a term that gets bigger if the module has been active before. Therefore the module that was used before has a higher probability of being chosen which refers to the chatbot staying in the topic characteristic.

3. Frankenbot: System Overview and Capabilities

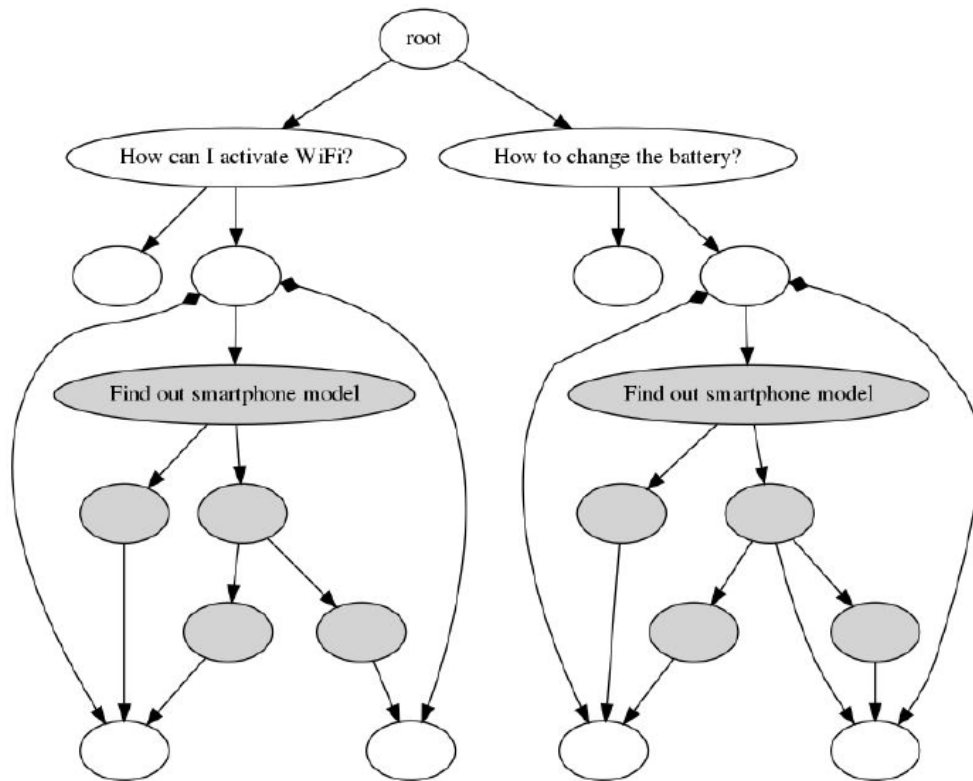


Figure 3.5.: Module usability within a chatbot

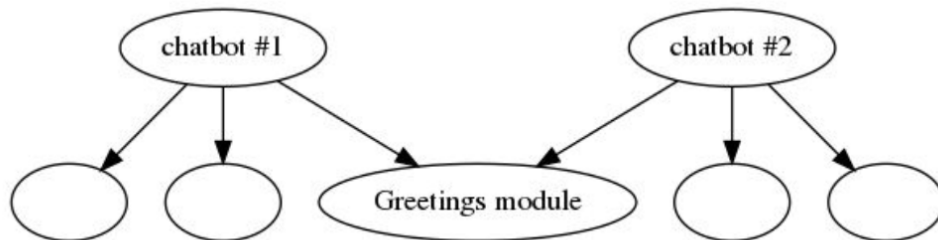


Figure 3.6.: Module usability between different chatbots

3.5. Frankenbot: System Architecture

Frankenbot's system architecture has been sketched in Figure 3.7. It consists of a server and server-side web API. The user communicates using a browser and sends a request using web API to a server. Furthermore, the server contains the deployed Frankenbot's

3. Frankenbot: System Overview and Capabilities

framework built using Python 3. Web server loads data for a chatbot from the JSON file for now but will be integrated with some database in the future. Whereas, a frontend for user's interaction has built simply using javascript, HTML, and CSS. Web API has been built using python library known as Flask [52].

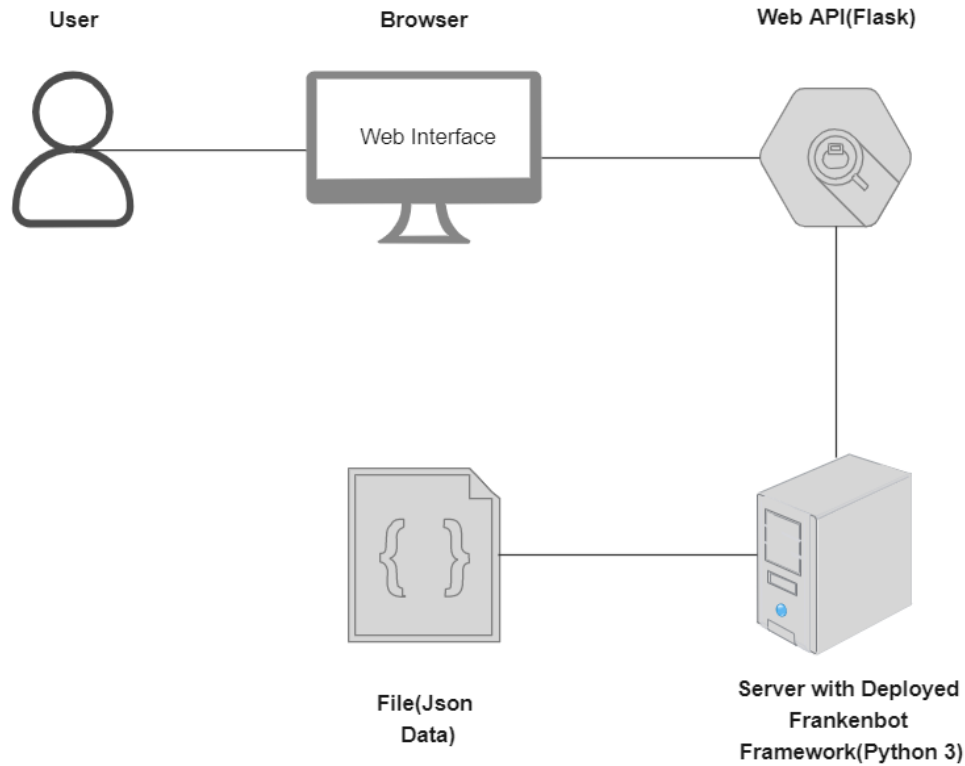


Figure 3.7.: Frankenbot's System Architecture

As it is clear from Figure 3.7 that the communication between the client and the server is possible only via web API composed of a simple get request. In addition to that when you dig deep into the framework, it is composed of several different components.

3.5.1. RASA Framework for NLU

RASA is an open-source renowned conversational artificial intelligence framework for designing contextual virtual agents. It is a machine learning framework designed to communicate through automated text and voice-based techniques. [2]

For making it work, the following steps must be taken into account:

1. Provide training data [53].

3. Frankenbot: System Overview and Capabilities

2. Provide configuration file with pipelines for training [54].
3. Provide directory's path to save trained models.
4. Generate rasa interpreter by loading it from the trained models' directory.

For utilizing it in a Frankenbot, necessary steps have been mentioned below under the heading of Configurations.

Training for RASA NLU

It has been used for natural language understanding in Frankenbot. It needs to get trained first to produce some useful results and for that purpose, one needs to provide it with the training data. It can be provided using different formats as mentioned on its website [53]. For Frankenbot, JSON format has been adopted but it can be changed without any issue based upon the developer preferences. The format listed in Listing 3.1 consists of a top-level object called `rasa_nlu_data`, with the keys `common_examples`, `entity_synonyms`, and `regex_features`. The most important one out of these all is `common_examples`.

```
1 {  
2   "rasa_nlu_data": {  
3     "common_examples": [],  
4     "regex_features" : [],  
5     "lookup_tables"  : [],  
6     "entity_synonyms": []  
7   }  
8 }
```

Listing 3.1: RASA NLU Training Data JSON Format.

Moreover, `common_examples` are a list of objects with the keys `text`, `intent`, and `entities` as shown in Listing 3.2. Whereas, `entities` can be an empty list or the objects list with the keys `start`, `end`, `value`, and `entity`.

3. Frankenbot: System Overview and Capabilities

```
1 {
2   "text": "...",
3   "intent": "#...",
4   "entities": [
5     {
6       "start": ...,
7       "end": ...,
8       "value": "...",
9       "entity": "@..."
10    },
11    ...
12  ]
13 }
```

Listing 3.2: JSON Objects for common_examples.

3.5.2. Frankenbot Framework

This framework is composed of various components. Each of which is responsible for performing a specific task. It has been majorly divided into the following two divisions: (i) The web server(backend), which is responsible for the creation of the whole chatbot and producing desired responses for a user and (ii) A client(frontend) also known as a user interface to interact with a chatbot. Both of these are explicitly explained below.

The detailed diagram for Frankenbot's Framework Architecture has been portrayed in Figure 3.8. All the components have been discussed in detail.

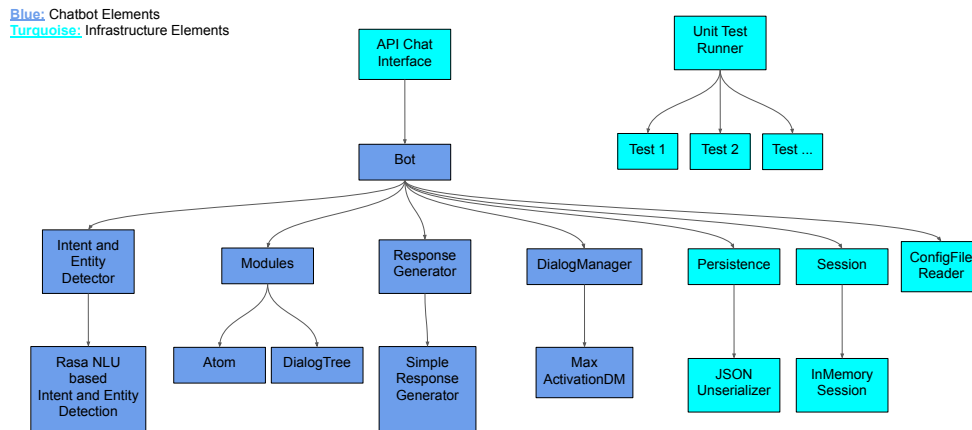


Figure 3.8.: Frankenbot Framework Architecture Diagram

3. Frankenbot: System Overview and Capabilities

Client

It is implemented using simple JavaScript, Html, and CSS. The template can be found under [55]. It is a simple user interface for a chat which allows a user to type a message and send it to the server by making simple get request utilizing the web API.

Furthermore, it is also responsible for storing the user's session id to storage on a browser. For that purpose browser's "localStorage" [56] is used. If the user is connected to the server the first time to the server then it requests an empty session id and user utterance to the server. And as a response chatbot sends a welcome message and assigns an id to a user. Which is saved in the browsers local storage for every user. Whenever there comes a new request from this user to a server, the session id is being sent along with a user utterance to a server. In return, the server responds with a suitable response according to the user utterance which is being displayed next to the last user utterance. The response from the server is in the form of a JSON object which is being parsed to extract the chatbot's reply and displayed for a user. The important point to take into account is that the user id will be stored until the page is not reloaded or closed. If you want to restart your session and get away with all your previous chat, then you can simply refresh the web page and it will make a new instance of the chatbot. But for the future, a server will have the record of all dialogues that happened in a past stored in a database.

For better understanding, the graphical representation of an interface is displayed in Figure 3.9.

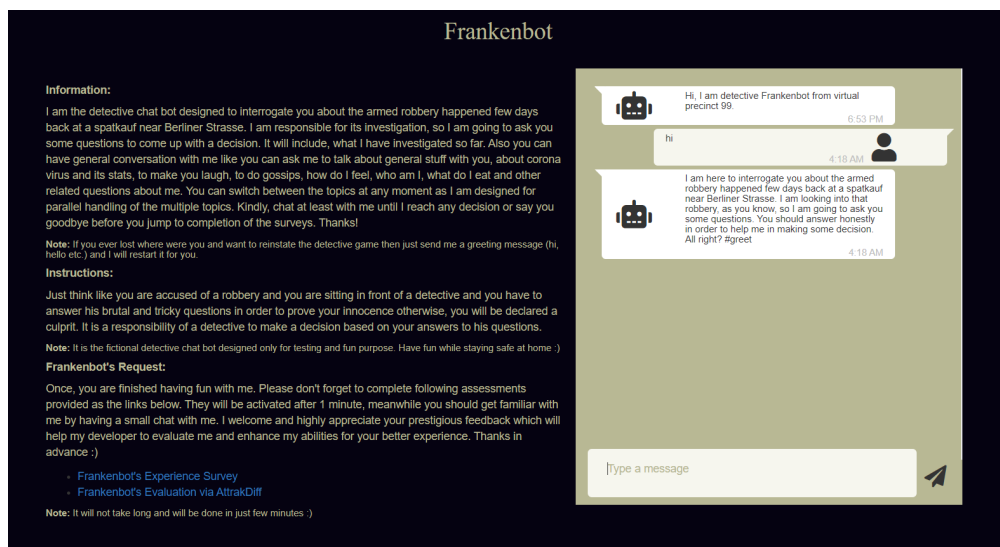


Figure 3.9.: Frankenbot's User Interface(Client)

As demonstrated, the left side of the screen contains important information and instructions for a new user. Whilst, on its right, there exists a chat window showing the first

3. Frankenbot: System Overview and Capabilities

most message as a welcome message for the newly arrived user. After that chat between the user and a bot initiated. The user can type a message in the bottom-most text box and press sends button or enter button from keyboard. The user's message will be displayed on the left side of the screen while the response will be shown on the right side of the screen's display and vice-versa according to your viewpoint.

Web Server

The web server itself is a composite of multiple classes performing specific tasks and connected to a client using web API. It is written in Python version 3. Abstractly, it is responsible for performing all tasks required to make a chatbot functioning. Firstly it receives user utterance as a parameter and initializes a chatbot from data provided as a source file of the JSON format. After initializing a chatbot, it undergoes various actions for a user utterance to generate a response.

Infrastructure Elements

Web API It is implemented using the python library known as Flask. It is responsible for making a connection between the user and a server. It makes a user communicate with a server. Whenever there comes a new request from the client, firstly it passes from the function implemented for its handling.

Important tasks performed by the web API are:

1. Receiving the user utterance and session id as a parameter from the get request made by the client.
2. Checking whether the session id already exists or its a new user. If it happens to be a new user then there will be the empty user utterance and user session-id received as a parameter from the client's get request. It will initialize a chatbot using the data provided in the JSON file (discussed in detail below). Afterward, it assigns a new user session id and sends it along with a welcome message as a response to a client.
3. If a session already exists, then it checks for the current dialogue state for an active user. And instead of initializing a new chatbot, the chatbot with the last stored state is being traversed with a new user utterance to get an appropriate response. And before responding to the client the current state gets updated again and stored for the current user.
4. The response is being sent as a JSON object.

Persistence Persistence includes the Json Unserializer. It takes the JSON file as an input and parses it for generating an initial chatbot.

3. Frankenbot: System Overview and Capabilities

The JSON file contains data as a JSON dictionary object which includes the name of a bot, welcome message, fallback message in case of no response, and a dialogue manager. Inside a dialogue manager, there exists a list of modules consisting of several modules having unique module ids. Additionally, each module contains a dialogue tree having a list of tree nodes. Whereas, each tree node includes a unique node id, information about its parent node, name of intent to which it belongs, and respective response generator. Furthermore, a response generator includes a mode for the responses that can be sequential or random and a list of responses. And all of the above-mentioned elements are uniquely identified using a key "type". For better understanding, just have a look at a JSON sample shown below in Listing 3.3.

```
1 {
2   "type": "bot",
3   "name": "...",
4   "welcome_message": "Hi, ...",
5   "fallback_message": "any desired message for bot's failure.",
6   "dialog_manager": {
7     "type": "max_activation_dialog_manager",
8     "modules": [
9       {
10        "type": "dialog_tree",
11        "module_id": "module1 or any id",
12        "dialog_tree": [
13          {
14            "type": "tree_node",
15            "node_id": any integer e.g 1,
16            "parent_node": null for making root or any node id to
17                          assign it a parent,
18            "intent_name": "#...",
19            "response_generator": {
20              "type": "simple_response_generator",
21              "mode": "sequential or random",
22              "responses": [
23                ...,
24                ...
25              ]
26            },
27            ... ,
28          ]
29        },
30        ... ,
31      ]
32    }
33 }
```

Listing 3.3: Frankenbot's JSON Structure.

3. Frankenbot: System Overview and Capabilities

Theoretically, a bot is composed of a dialogue manager that contains information about all the modules representing the entire dialogue structure. Moreover, a module includes a dialogue tree that has been developed to design a dialogue. Each dialogue tree is made up of the relational tree nodes. And each tree node holds all the technical information about the intent and other required fields for a dialogue to produce the respected response.

Now coming towards practical implementation, a bot is generated recursively based on unique type key. The JSON data has been iterated and checks for the type of each element recurrently. And by taking that type of element into account, it returns the object for different respective classes. Let suppose the JSON object contains types in the following sequence:

1. bot; recall the same function with a JSON for a dialogue manager returns a class Bot's object afterward.
2. max-activation-dialogue-manager; recall the same function with a JSON for each module in a list of modules and returns a class MaxActivationDialogueTreeManager's object afterward.
3. dialogue-tree; recall the same function with a JSON for a dialogue tree and returns a tree consisting of tree nodes afterward.
4. tree-node; recall the same function with a JSON for each element of a list dialogue tree and returns a class Atom's object afterward.
5. simple-response-generator; recall the same function with a JSON for a response generator and returns a class SimpleResponseGenerator's object afterward.

Now the function initiates and finds the very first type "bot" in a JSON data shown above. And then the recursive call to the same function is sent with "dialogue_manager" JSON object. Likewise, sequential recursive calls happen for types "modules", "dialog_tree" and "response_generator" with their specific JSON objects. Once it reaches the last key type "simple_response_generator" then it starts returning the corresponding class object related to each type as listed before. So, the function starts executing from the first step mentioned above but starts returning recursively from the last step. It means, the class SimpleResponseGenerator's object is passed as a parameter to the constructor of the upper-level class object i.e. class Atom in this case. The same goes for all other classes. In this way, class SimpleResponseGenerator's object gets appended to the class Atom's object, and so on. Finally, the class Bot object should contain the class MaxActivationDialogueManager's object as an attribute and that is how the chatbot will be generated recursively.

Also, there is a convention used to represent the intents, entities, and context variables. It has been taken from the state of the art chatbot IBM Watson as the intent is denoted by prefix "#", an entity is denoted by "@" and "\$" symbol is used to denote a

3. Frankenbot: System Overview and Capabilities

context variable. Same convention has been used in Frankenbot as shown in the Table 3.2.

	Key	Value
Intent	#intent_name	intent_value
Entity	@entity_name	entity_value
Context Variable	\$context_var_name	context_var_value

Table 3.2.: Representation for an Intent, Entity and Context Variable.

Session It is responsible for session handling. It is subjected to establish "InMemory-Session" which is meant to store session id in a list of session variables, context variables, and sequential response counter for each user. For each user, there exists a different chat-bot's state.

Whenever there comes a new user, the new dictionary object has been declared to store session id for a user along with its bot's state, active module id, and active node for each module. In addition to that, the last user utterance also gets stored for generating the last bot's message to the user if the chat window has been refreshed or reopened in a new tab of browser by a user(can be implemented in future). And then that dictionary object gets appended to a list of session variables. Which means there exists a separate object for each user in a list. It gets updated on every request from the client.

Another element present to save the context of dialogue for each user is known as context variables. They are independent of modules and get detected by NLU within a user utterance based upon the entities provided within the training data. It is a dictionary having a user session id as a key, and each key represents a key-value pair for entities that appeared in a dialogue. Each entity name appeared in dialogue has been saved as \$EntitytName as a key and a value of the entity as its value. Its value gets updated every time if the same entity has been triggered or a new entity has been discovered in a dialogue for the same user. These context variables are used to understand the context of dialogue and update the entity value in a response if it is intended.

Another important attribute available is the response sequential counter. Its main task is to keep track of all intent's response counter for each user if its mode has been set to sequential. It means if an intent contains multiple responses and the bot has to produce a response for it based on the user utterance then each time the next response should be produced for a user. It is intended to give a more realistic impression to a user and human-like interaction behavior to a dialogue.

Configurations To make chatbot work it is really important to provide it with proper configurations. Frankenbot needs three following directories to function:

3. Frankenbot: System Overview and Capabilities

1. bot directory; It should be provided with the bot's directory name as a command-line parameter while running the project. For example, the running command in my case is "python app.py detective". In this command "detective" is the name of a directory which should contain JSON file for initializing a bot. And it should be named "frankenbot.json" (see Appendix E to have a look at actual JSON data used for the structuring of the Frankenbot).
2. Furthermore, the bot directory must contain training data directory named as "rasa". Firstly, this sub-directory must contain a configuration file named as "config_spacy.yaml" defining pipelines required for Rasa NLU Training[54] (see Appendix G to have a look on actual pipelines used). Secondly, JSON files must be available with training data for each module in "frankenbot.json". And the names for those training files should be the same as module id assigned to a module for which it contains training data. For example, "frankenbot.json" contains two modules, and ids assigned to them are detective_tree and general_tree. So, the training files should be named "detective_tree.json" and "general_tree.json" (see Appendix F to have a look at actual JSON training data used for both training of modules in the Frankenbot).
3. Lastly, the model directory is required to store the rasa models produced after the model gets trained for each module. It can be any directory but one has to provide an appropriate path to it.

Chatbot Elements

Bot Once the initial bot is generated, it is the very first element that has been instantiated. Originally it contains the complete definition for a chatbot. Which includes attributes such as chatbot's name, welcome message, fallback message, and most importantly dialogue tree manager (discussed below in next heading).

Whenever a request from a client has been made it firstly gets processed by this component. As it receives the user utterance, active tree nodes for each module for a particular user and session information as the parameters.

Complete flow can be observed in Figure 3.10. The parameters are further passed to the dialogue tree manager to find a module with the highest intent's activation value, detected intent, intent's activation value, detected entities, entities activation values, active tree node, active module id, and JSON output object. This JSON object has been returned to a client containing all important information and respective response.

Before returning the response as a JSON object, it is also responsible for performing some major tasks. Firstly, it checks for detected entities received after the processing of its dialogue tree manager. If there exists any detected entity then entity(s) in the context variables for a particular user gets added or updated depending upon the current

3. Frankenbot: System Overview and Capabilities

state of a bot.

In addition to that, it also checks for the detected intent's activation value for a user utterance. As Rasa's NLU has been used to detect the intent and there are the chances of getting a false result with very low activation value. So it checks for it and responds only if its value lies above some threshold. The range of activation value lies between 0 and 1. If the value is below the threshold then it checks for the last active node for a user whether it exists or not. If it doesn't exist then, it means the user started a chat with some meaningless utterance as an input for a bot. So, the chatbot should respond using node with intent `anything_else` if it exists in a module with the highest activation obtained from RASA's NLU result. Secondly, if there exists some chatbot's active state for a user, then it checks for the last active module for a user and also checks for a node with an intent name `anything_else` in its children. If it is available then the response has been generated using it. Otherwise, it simply responds with a fallback message.

Alternatively, if it is above the threshold then the highest module received from its dialogue tree manager has been used to generate an appropriate response(explained ahead).

Dialogue Manager Dialogue manager is an abstract class whereas the max activation dialogue tree manager is an inherited class that should implement the abstract functions of a dialogue manager. Max activation dialogue manager originally contains the dialogue tree, once the chatbot has been initialized. Now the question arises what is the dialogue tree composed of? So, the dialogue tree gets generated while chatbot is loaded from the JSON file. And it is composed of the list of tree nodes, unique module id, data for RASA's NLU training and interpreter loaded from model directory once the training has been finished. Whereas, each tree node contains unique node id, information about its parent node, intent, activation flag which gets set if it is responsible for generating a response, and a module object which itself consists of the intent detector and response generator. The dialogue tree has been generated using the python library "any tree" [57].

A detailed visual chart for a dialogue manager is portrayed below in Figure 3.11. It is responsible for finding a module with the highest activation based upon the detected intent's activation value for the user utterance. So for this purpose, each tree must be traversed for each of its nodes. As each node contains a module object which is further utilized for this purpose. Module's activation function is being called bypassing user utterance, RASA interpreter for the current module, module id for a recent module, and session data as parameters. And it returns activation value for each intent along with its name, recognized entities along with their activation values and the JSON object for output. As activation function for all nodes in each tree has been called and meanwhile dialogue manager keeps on checking for the activation value received for each node's module. And keeps on updating it whenever the last received value is greater than the previously stored value. In this way, the module with the highest activation is detected.

3. Frankenbot: System Overview and Capabilities

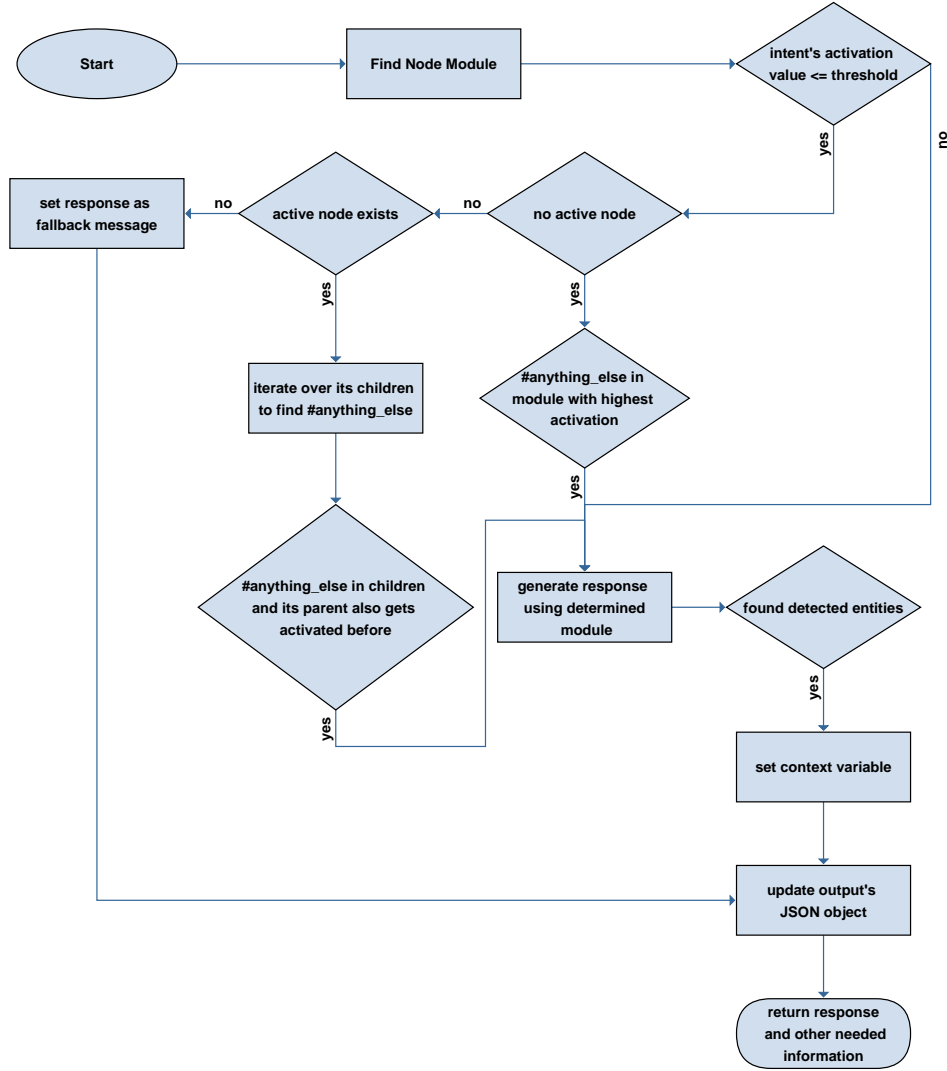


Figure 3.10.: Flow chart to finalize and update JSON output.

Responsibility of a dialogue manager not only ends here but it is also responsible to make the chatbot follow the modular architecture which is the main goal behind the designing of the Frankenbot. As explained above about the modular architecture that how should it work, let's discuss it here in a technical perspective. Whenever, the recently received activation value is greater than the last stored activation value and the identified intent is same as the current node's intent name then there comes following scenarios, if the recently traversed node is a root node or not:

1. If it is a root node then its activation flag should be set so that for the next time its

3. Frankenbot: System Overview and Capabilities

children can be accessed for response generation. Also, all the required information must be stored and returned to the previously discussed bot component. Which is responsible for returning the response to the client via web API and the active state for the current user must get updated beforehand.

2. Secondly, if the recently processed node is not a root node then what should be its behavior? It is a bit complex strategy to follow. Firstly, it should check if its parent's activation flag is set or not. If it is set then it should proceed further otherwise it should not. Let's suppose that it is set and it steps ahead and finds multiple nodes with the same intent in its children nodes then it will not respond correctly. As it is the main idea behind the modular architecture that modules can be reused but different modules with the same intents do not need to have the same responses. Responses can differ at different levels in a tree whilst having the same intent. So for removing this problem, an active state for each module has been stored separately. This means if there exist two modules for a chatbot then there should be two separate active states, one for each module, available for a user. So by using these active states, it can match the last active node's id with the current node's parent id and if it is true then it is good to go. Also what if the user starts to enter the same user utterance again and again. It will again cause a problem as it will be searching for intent in the last active node's children but it will not be able to find it. So to overcome this issue, the manager matches the current node's id with the last active node's id and again if this condition has been satisfied then it should respond correctly.
3. Lastly, irrespective of the above conditions whether any of them is satisfied or not. It should return to a bot component which is meant to handle all scenarios and responsible to produce an appropriate response if there exists any highest detected module. Otherwise, it uses anything_else node or fallback message to respond depending upon the state and design of the bot.

In the end, it also checks for identified entities and their activation values and stores and updates them accordingly. And all the essential information is returned to a bot component. Which is responsible for returning the response to the client via web API and the active state for the current user should be updated beforehand.

Modules Modules class is responsible for generating activation value for each intent which is passed to the dialogue manager for further processing as described above. In addition to that, it is also pledged for generating responses and JSON objects for API output. So these three functionalities should be implemented in a class inherited from abstract modules class. So, class Atom is inherited from it and provides all functional definitions for abstract class.

There are two attributes needed for the atom class. One is natural language understanding(NLU) which is an object for rasa intent and entity detector and second is a

3. Frankenbot: System Overview and Capabilities

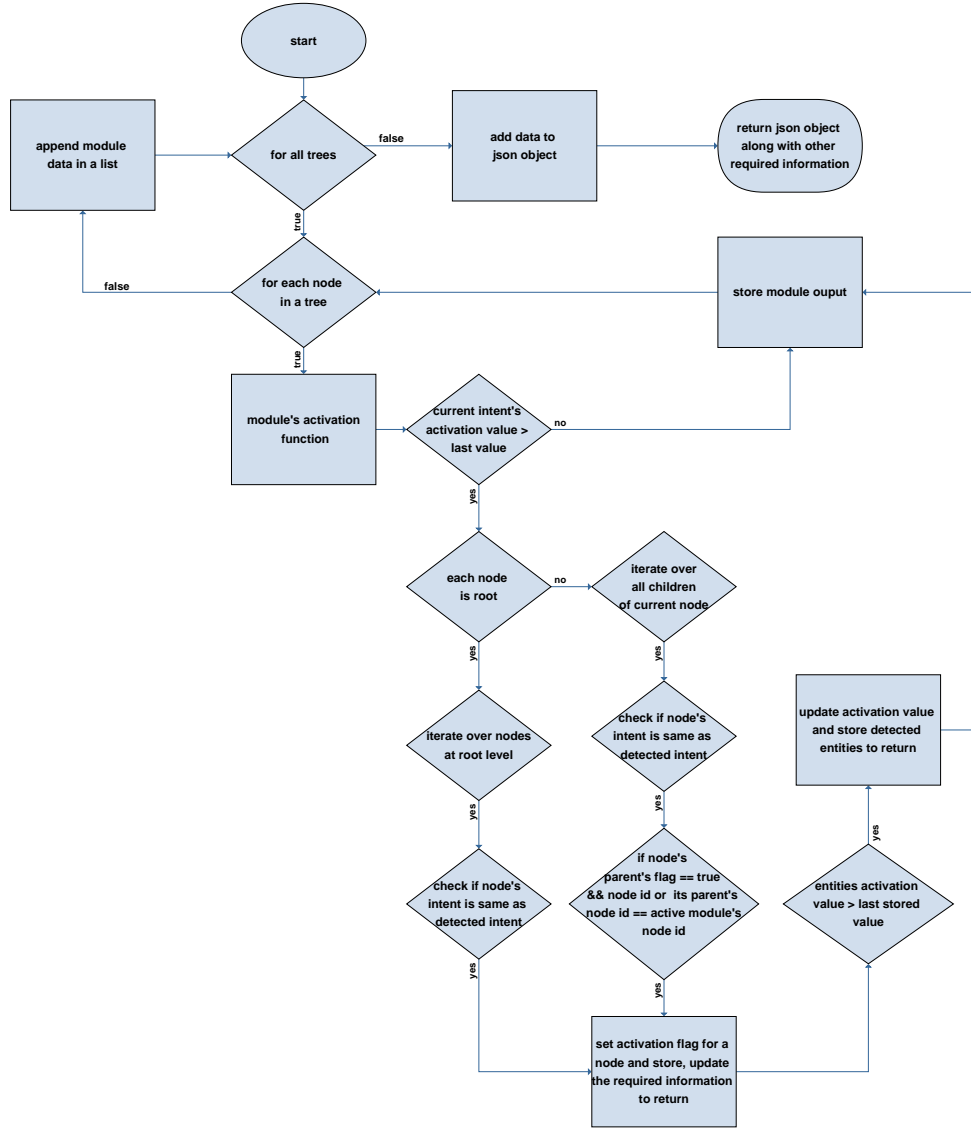


Figure 3.11.: Flow chart to find module with highest activation value.

simple response generator's object, Both of these components have been described next under their respective headings.

Starting with an activation function is graphically represented in Figure 3.12. It initiates from a max activation dialogue manager's function to find a node's module with the highest activation value, all trees are traversed for each corresponding node. Each node contains a particular module object with its NLU and response generator. Using that module, activation function has been called provided user utterance, rasa inter-

3. Frankenbot: System Overview and Capabilities

preter, recent module id for a tree, and session information as parameters to it. And what it does with these parameters received is, it passes them further to make a call for natural language understanding to detect intents and entities. And the encountered intents and entities are then further passed for creating JSON object to return to the client as an API output.

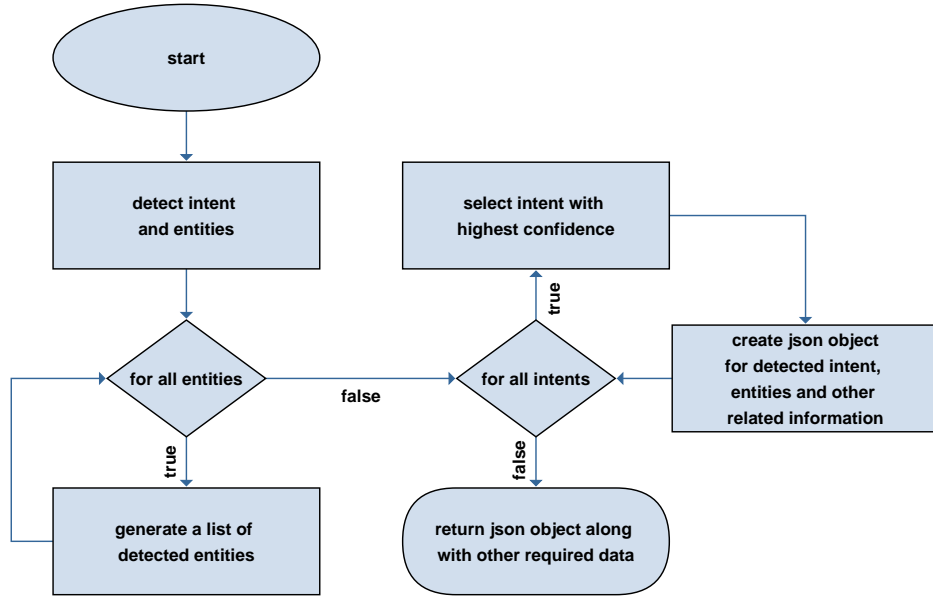


Figure 3.12.: Flow chart for the module's activation function.

Moreover, it also has been used to generate a response with the help of the response generator for an encountered module with the highest activation based upon the identified intent and entities for a particular user.

Intent and Entity Detection It is one of the basic and necessary components for all chatbots. The same applies to Frankenbot. An intent and entity detector is responsible for the identification of intent and entities for a user utterance. So this feature must be handled within a successor class for it. So rasa intent entity detector is implementing this functionality for an abstract parent class.

Each tree node consists of an atom and each atom consists of the intent and entity detector with the intent and entities information stored for each node. Visuals for it have been displayed in Figure 3.13 exhibiting a comprehended process. It is responsible to detect the intent and all the entities for the user utterance by utilizing the parametric RASA interpreter passed to it from the atomic activation function.

Coming towards internal processing, the user utterance is passed to the trained rasa

3. Frankenbot: System Overview and Capabilities

interpreter to detect the intent and entities and matched with the current node's intent. If they have been appeared to be identical then the intent name is stored as a key in a dictionary object and a confidence value received from the result of the RASA interpreter represents its value. Also, the entities are stored as a list of dictionary objects and each object contains key-value pairs for entity name and confidence value, starting and ending index in an utterance. These detected intent and entities are returned from this function to its origin for further handling.

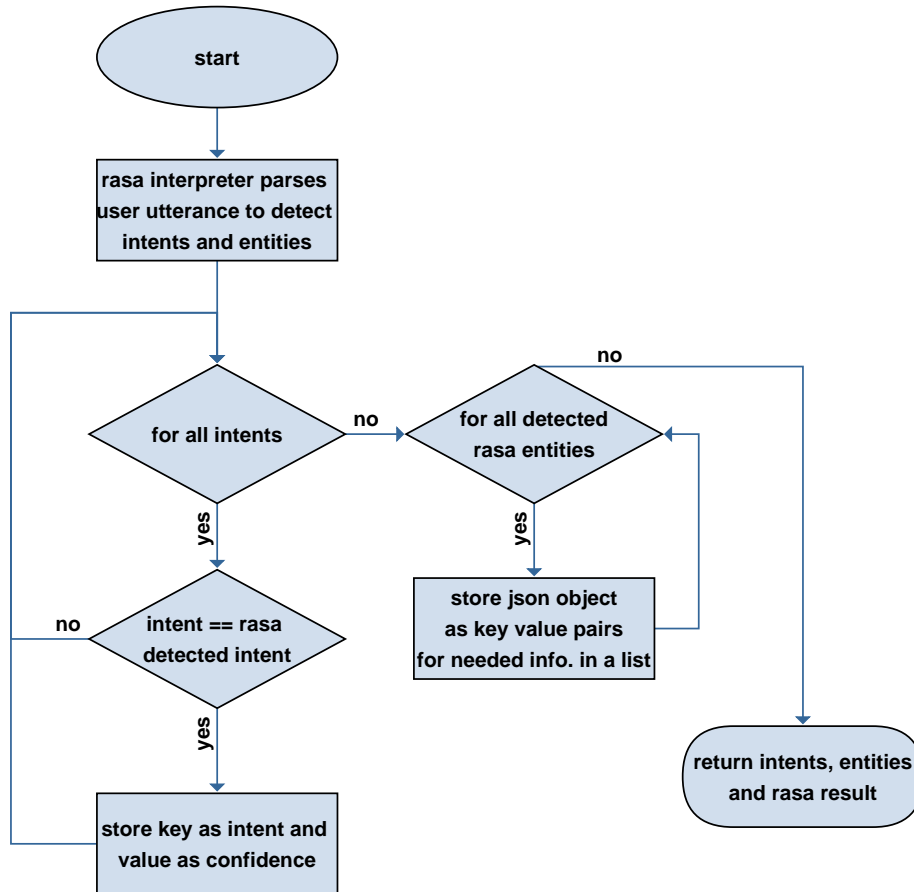


Figure 3.13.: Flow chart for the Rasa Intent and Entity Detector.

Response Generation As the name itself is self-explanatory that what task should it perform. The response generator is liable for producing a response by taking an identified intent into the account. Additionally, it is also responsible for appending chatbot response to the final JSON object designed to be returned as an API output.

3. *Frankenbot: System Overview and Capabilities*

The precedent classes must take in to account both of the functionalities that parent abstract class is responsible for. A simple response generator does it all for Frankenbot. As it needs mode for a response that can be sequential or random and a list of responses for a captured intent for response selection.

Once the pre-processing has been completed, means all the steps have been finished and the dialogue manager has returned the atomic module with the highest activation value and a chatbot needs to generate a response. Now, a bot uses that observed module to send a request to this component by providing it with recognized intents and entities along with session information as an input. So that it can go ahead and perform the task that it is responsible for.

Its graphical representation is shown in Figure 3.14 below. Firstly, it checks for the intent's sequential response counter for a specific user and updates it accordingly for next time usage. Secondly, after the selection of a response, it checks within a response for entities and context variables. If there is any detected entity for a user utterance and a response needs to be exchanged with its value then it should be handled here. Also, if there is any context variable stored previously for a user and bot's reply needs to get modified with its value then it should also be done here. After all, processing has been accomplished, it should create a JSON object with a chatbot response for web API output and send it back to a bot component for performing next required executions.

3. Frankenbot: System Overview and Capabilities

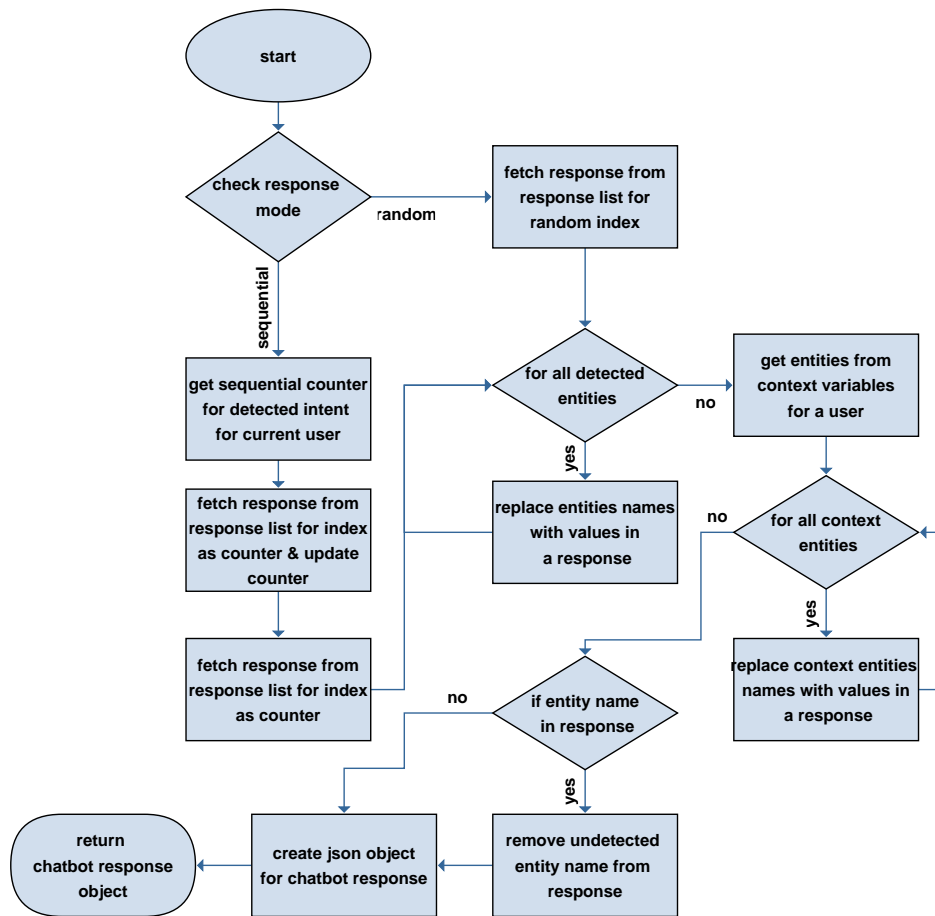


Figure 3.14.: Flow chart for response generation.

JSON API Output After all components are done with their part, bot returns a JSON object to web API which passes it further to a client which is parsed at the client side for displaying needed information for a user. Let's start with an observation of JSON object demonstrated below in Listing 3.4.

3. Frankenbot: System Overview and Capabilities

```
1 {
2   "user_utterance": {
3     "text": "...",
4   },
5   "chatbot_utterance": {
6     "type": "simple_response",
7     "response": "...",
8   },
9   "active_module": {
10    "id": "...",
11    "type": "dialog_tree_module",
12    "activation_value": ...,
13    "module_output": {
14      "recognized_intent": "...",
15      "recognized_entities": [],
16    }
17  },
18  "modules_output": [
19    {
20      "id": "...",
21      "type": "dialog_tree_module",
22      "activation_value": ...,
23      "module_output": {
24        "recognized_intent": "...",
25        "recognized_entities": [],
26        "intent_ranking": [
27          {
28            "name": "...",
29            "confidence": ...
30          },
31          ...
32        ]
33      }
34    },
35    ...
36  ]
37 }
```

Listing 3.4: Frankenbot's API JSON Output.

The JSON object contains `user_utterance` which holds a key `text` for the user's message as a value. Secondly, `chatbot_utterance`'s `response` is a final reply by a chatbot. An `active_module` reflects an `id` of the recognized module, `activation_value` in terms of rasa's confidence value for an identified intent. Whereas, its child `module_output` depicts an encountered intent and detected entities. Finally, `modules_output` involves the dictionary object for all modules in a chatbot through which user utterance has been processed to generate a response after intent detection. So, `id` shows an `id` of a module, `activation_value` is same just like as mentioned before for `active_module`. Furthermore, mod-

3. Frankenbot: System Overview and Capabilities

ule_output is also similar to active_module's module_output but with an additional key for intent_ranking. It is comprised of all the names for the intents and their respective confidence values obtained from rasa's trained natural language interpreter.

Logging It is something that has been used commonly nowadays to track the events occurring in any software or a system.

For the framework, it has been implemented using python's library called "logging" [58]. Before returning the JSON object via web API to the client, this object with all the essential information from start i.e. received user utterance, till end i.e. generated suitable response, has been logged to a log file to keep track of the complete dialogue. Also, the log file contains the information for RASA's NLU training process and what intents and entities have been found in training data. Additionally, it also helps in tracking the error or reason of the chatbot failure, if any such event occurs.

4. Experiments and Evaluations

To evaluate Frankenbot’s modular architecture based on the user experience, the surveys have been conducted.

For evaluation of this research, a link provided in Appendix A to the deployed chatbot has been shared with participants using different social platforms along with an introductory text about the purpose of the study. Random users were selected irrespective of their profession, study background, and any race or gender discrimination. But the age limit constraint was set to be above 18. As participants were considered to be accused of a robbery and have to chat with a detective bot and answer his questions which can be a bit strict and straight forward for youngsters below 18. So the participant who participated with the minimum age was 22. On the contrary, the one with the maximum age was 34. And the average age calculated as 25.65 with a standard deviation of 2.37. Visuals have been shown in Figure 4.1.

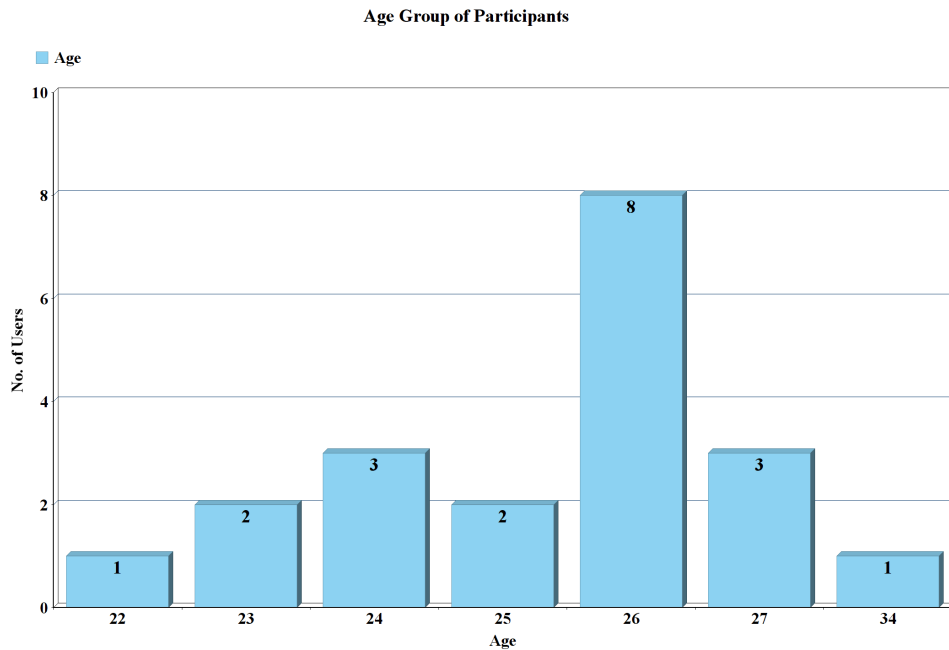


Figure 4.1.: Age stats of participants

4. Experiments and Evaluations

Figure 4.2 depicts that a total of 20 users participated in this study. Most of them appeared to be from a technical background as a maximum of them appeared to be students from different universities. The second majority of participants revealed themselves as IT employees. While remaining revealed themselves as teacher and business qualified.

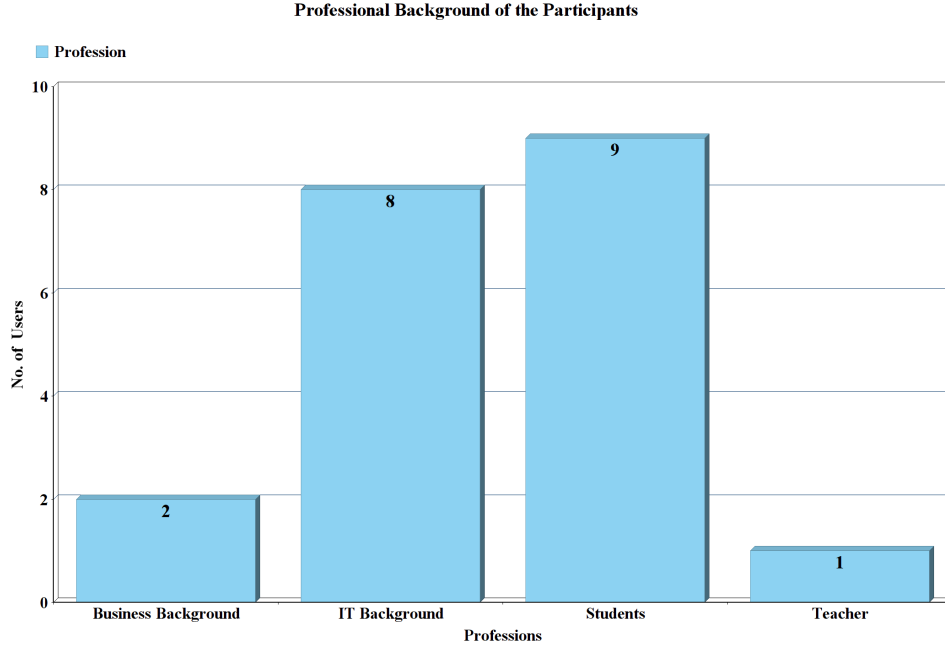


Figure 4.2.: Profession related details of participants

4.1. Conducted Surveys

All participants were asked to complete two surveys. One named as "Frankenbot's Experience Survey" was designed with the help of [59][60]. For the second survey, the standardized tool to personally evaluate the usability and design of an interactive product known as AttrakDiff [7] has been used. The first survey was designed using Google Forms and the purpose of it was to capture the user's interaction experience about the Frankenbot. While the purpose of AttrakDiff's study was to evaluate different aspects such as Frankenbot's utility and usability. Furthermore, it also helped to weigh the chatbot for task-oriented and self-oriented qualities.

The users were provided with the link to the deployed chatbot's interface as a web page and they can easily access it from their places. It provided ease to the user and

4. Experiments and Evaluations

enhanced the comfortability factor for them. Also, they have to read the description and instructions provided for them on the web page. And they have to figure out what to do and how to operate the chatbot on their own without any external help. Which has given more realistic and unbiased essence to the results obtained.

For the surveys, the web page contains the heading "Frankenbot's Request" in which the users were requested to complete the surveys. Once, they are finished having chat for a fun purpose with the chatbot then they could visit the hyperlinks provided for both of the surveys.

The surveys were conducted in the English language. Whereas, AttrakDiff's survey has the option for both English and German. You can find "Frankenbot's Experience Survey" in Appendix H. Furthermore, AttrakDiff's Single Evaluation Study[61] has been used as the second survey.

The questionnaires were filled by the participants on their own devices. While a user was interacting with the chatbot all communication was getting stored into the log file for future records and usage.

4.2. Experimental Setup

This research study has been conducted to gather the results for what the user has experienced while interacting with the chatbot. The whole experiment itself has been divided into four major parts mentioned below:

1. Explanation of the chatbot for the participants to make it's testing successful.
2. Participants were requested for accomplishing a set of tasks with the chatbot.
3. The questionnaire to collect the user's interaction experience about the chatbot named as "Frankenbot's Experience Survey".
4. Quality evaluation of the chatbot via AttrakDiff's Single Evaluation.
5. Short interview of the participants.

All of these are explained underneath in detail.

4.2.1. Explanation of the Chatbot

Users were provided with the description and instructions on the web page about the chatbot. The following sub-sections contain the required information and directions for the users to operate the chatbot. Visual representation has been displayed in Figure 3.9.

4. Experiments and Evaluations

Information

Real information provided to the participants has been provided in Appendix B. They were provided with the introduction about the Frankenbot that it is the detective chatbot designed to interrogate you about the armed robbery that happened a few days back at a spatkauf near Berliner Strasse. It is responsible for investigating, so it is going to ask you some questions to come up with a decision. It will include, what it has investigated so far. Also, you can have a general conversation with it like you can ask it to talk about general stuff with you, about corona virus and its stats, to make you laugh, to do gossips, how does it feel, who is it, what does it eat and other related questions about the chatbot. You can switch between the topics at any moment as it is designed for parallel handling of multiple topics. Kindly, communicate with it until it reaches any decision or says you goodbye before you jump to completion of the surveys.

Lastly, the ending of the information provided a brief note about how to re-initiate the chat. If a user gets lost in between the dialogue and wants to reinstate the detective game then just send a greeting message (hi, hello, etc.) and the chatbot will restart it for him/her.

Instructions

Actual instructions are stated in Appendix C. The user was asked to think as he/she is accused of a robbery and sitting in front of a detective and have to answer his deceptive questions to prove his/her innocence. Otherwise, he/she will be declared as a culprit. It is the responsibility of a detective to make a decision based on the user's answers to his questions.

At the end of the instructions, there was a small notice available for the user to remove his/her doubts about its reality that it is the fictional detective chatbot designed only for testing and fun purpose.

4.2.2. Tasks

The participants were requested to accomplish the following tasks beforehand, to fill the surveys.

- Users were requested to play a small game with the detective chatbot and answer the questions, the way they wanted.
- They were also informed that they could have a general dialogue with the chatbot. They could also talk to it about general stuff, corona virus and its stats, to tell a joke, to do gossips, query about its feelings and emotions and other related questions.
- Additionally, they were also notified that they can switch the topics. This means they can start talking about general stuff if they are talking to the detective bot

4. Experiments and Evaluations

and vice versa by just replying to the last question of the previous topic.

- Lastly, they were requested not to leave the conversation in between until they reach the ending of the detective game.

All users were requested to chat with the chatbot at minimum until it came up with any decision or says goodbye. Other than that, there was no time restriction for them and they could have a dialogue for as long as they desired to.

Frankenbot's Request

Finally, the user was requested as stated in Appendix D that once he/she has finished interaction with the chatbot, please don't forget to complete two of the assessments demonstrated in the Appendix H. They were allowed to start with it after 1 minute of web page activation at a minimum. Meanwhile, a user should get familiar with the chatbot by chatting with it. Frankenbot also stated that it welcomes and highly appreciates the user's prestigious feedback which will help its developer to get it evaluated and enhance its abilities for a better experience.

4.2.3. Interview

In the end, a brief interview was conducted after having their consent. And not all the users permitted it but 12(60%) of the participants showed their availability for a small session. The users have been verbally asked the following questions:

- Whether they faced any problem to make chatbot understand their messages?
- Did they reach the end of detective game i.e. a chatbot declared them culprit or responded them with bye message?
- Whether chatbot guided them well in case of any confusion that how to proceed ahead?
- Did they enjoy the communication?
- Did they try with switching the topics?
- Whether they liked the detective game or not?
- Whether they liked the user interface or not?

4.2.4. Frankenbot's Experience Questionnaire

When the user completed the tasks allocated to him/her then he/she filled out the questionnaire about his/her interaction with the chatbot. The questionnaire is shown in Appendix H.1 has been divided into the following sections:

- Users' overall impression about the interaction with the chatbot.

4. Experiments and Evaluations

- Familiarity of the user with the already existing chatbots.
- Whether the user succeeded to achieve the desired goals.
- How was the communication with the chatbot.
- What was the behavior of the chatbot with the users.
- How well the dialogue was designed.
- Personal impression and user's experience about the chatbot.
- Users' opinions about the chatbot's usability.

All sections have consisted of several questions. And the user has to respond by selecting an option out of the linear scale from 1 to 5 whether he/she strongly agrees, agrees, undecided, disagrees or strongly disagrees with it. Most of the questions needed to be answered using these five options. Other than that overall impression about the chatbot has been measured using the linear scale of 5 options starting from 1 and ending on 5 as excellent, good, fair, poor, and bad.

4.2.5. AttrakDiff Single Evaluation

After completing the Frankenbot's Experience Survey, users have been requested to complete the AttrakDiff's standardized questionnaire to measure how the users perceived the design, quality, and usability of the chatbot.

The AttrakDiff is an institutional questionnaire that has been used to rate the products using the series of several word pairs [62]. Each word pair consists of the options as a linear scale from 1 to 7. Option at position 1 is a word while on number 7 there also exists a word but its acronym [63]. These pairs of words are classified into three following categories: (i) Pragmatic Quality, it includes the word pairs like "cumbersome - straight forward" and "impractical - practical". (ii) Hedonic Quality, the words pairs like "tacky - stylish" and "unimaginative - creative" lie under this category. And (iii) Attractiveness, word pairs such as "pleasant - unpleasant" and "ugly - attractive" are grouped under this attribute [62].

4.3. Quantitative Analysis

This section contains the explanation and quantitative analysis of the results obtained during the research. It has been further divided into two steps: (i) The results for all the sections for the Frankenbot's Experience Survey as mentioned above and also the results for the AttrakDiff Single Evaluation stated before.

4.3.1. Frankenbot's Experience Survey

It analyzes the results gathered about all the sections of the questionnaire.

4. Experiments and Evaluations

Overall Impression

This section includes a question about the user's overall impression about the interaction with the chatbot as pictured in Figure 4.3. It has been measured using the linear scale of 5 options starting from 1 and ending on 5 as excellent, good, fair, poor, and bad. And out of the total 20 responses collected by the conduction of the survey, 2(10%) responded with excellent mark and 9(45%) rated it as good. So, a total of 11 out of 20 which is 55% of the total participants, have passed positive remarks about it. Whereas, other 4(20%) judged it as fair which also lies under the category of acceptable. So after summing up the percentages, 75% of the participants rated their interaction as charming. On the contrary, out of the remaining participants, 3(15%) rated it as poor and only 2(10%) assigned it a bad label. So, it depicts that overall users' impressions about the interaction with the chatbot went well.

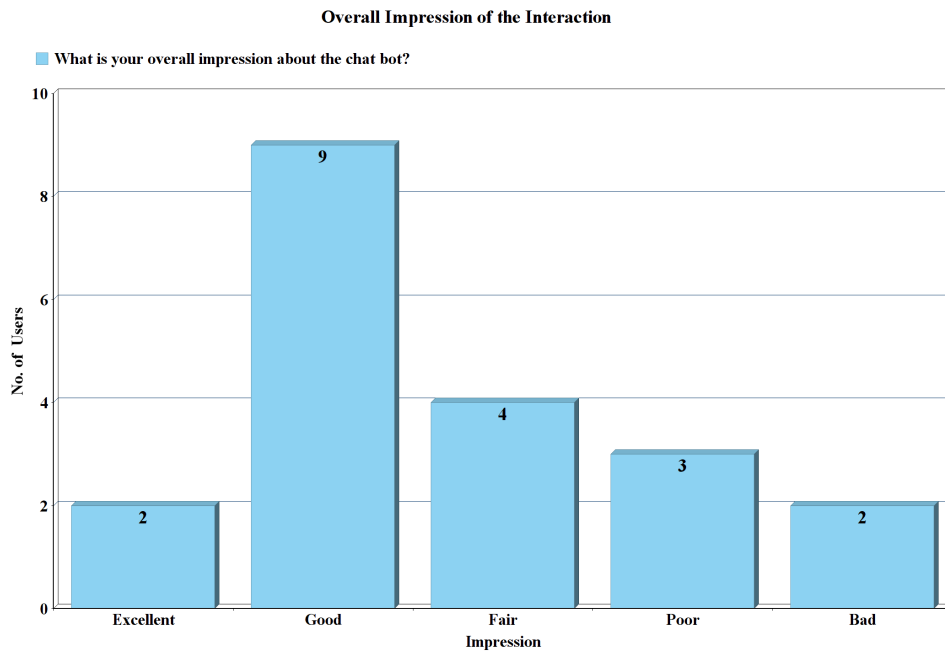


Figure 4.3.: Overall impression of the interaction with the chatbot

Familiarity with Existing Chat Bots

This section added to the questionnaire just to figure out the experience of the participants with already existing chatbots. Whether they were well familiar with the chatbots or are inexperienced with this emerging technology. So that if the majority of them have a good knowledge of any of the existing chatbots, it will provide more strength and

4. Experiments and Evaluations

solidity to the results gathered from such participants.

There were a total of 3 questions available under this section of the survey attached in Appendix H.1 that participants have to answer:

1. I feel that I am well familiar with the chatbots like Google Assistant, Apple's Siri, etc. (Possible options lie between the scale of strongly agree to strongly disagree).
2. I communicate with the chatbots. And the possible options provided were Frequently (daily or several times in a week), Seldomly (rarely in weeks or months), Just a few times and Never.
3. Purpose of my chatbots usage is: (only if you answered a question no. 2 positively). Possible answers could be any out of the following: Personal commands to provide you assistance in performing tasks, For fun, Not feel lonely and No reason.

Fortunately, for the first statement, 16(80%) appeared to be well familiar with the existing chatbots. 2(10%) answered with undecided and the remaining 2(10%) just disagreed with the statement but no one strongly disagreed with it. Graph depicting such results has been shown in Figure 4.4. Coming to the second statement, 5(25%) detected to be frequent users. 8(40%) resulted to be the seldom users. Whereas, 6(30%) communicated just a few times with the chatbots as shown in Figure 4.5. Lastly, almost 60% knew how to operate and give commands to the chatbot. And around 30% appeared to use it for joy and fun purposes. These stats and readings portray that experienced users participated in this study. And the results obtained from the study are trustworthy and reliable.

Achievement of Goals

It highlights whether the user succeeded to achieve the desired goals or not while having an interaction with the chatbot. It contained the following questions in responding to which user checked an option that varied from strongly agree to strongly disagree. Detailed comparison for all the answers collected for the following statements can be found in Figure 4.6.

1. The information provided by the chatbot was clear.
2. The provided information was incomplete.
3. The interaction with the chatbot was efficient.
4. The chatbot is unreliable.

Discussing the results and stats for the very first statement, 13(65%) participants agreed upon the information provided by the chatbot was clear. Whereas, 3(15%) were failed to decide about it. Only 4(20%) just disagreed with it and no one strongly negated it. For a better understanding of it just see Figure 4.6.

4. Experiments and Evaluations

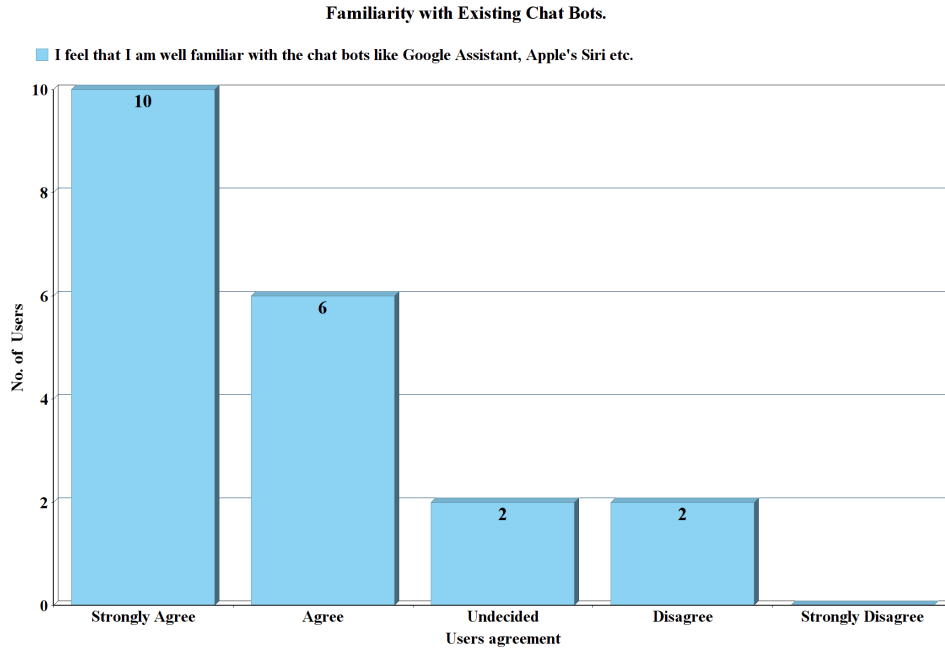


Figure 4.4.: Participants familiarity with the chat bots like Google Assistant, Apple's Siri etc.

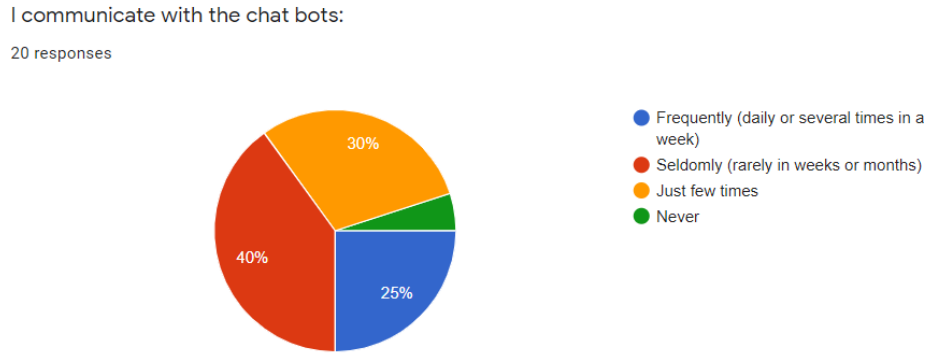


Figure 4.5.: Participants usage of the chat bots like Google Assistant, Apple's Siri etc.

For the second statement, 10(50%) disagreed with it which tells that for them the provided information was complete. Whilst 7(35%) were unable to make any decision about it. This is something can not be ignored. It can happen due to various reasons: (i) due to lack of concentration (ii) misinterpretation of information or instructions (iii) chatbot

4. Experiments and Evaluations

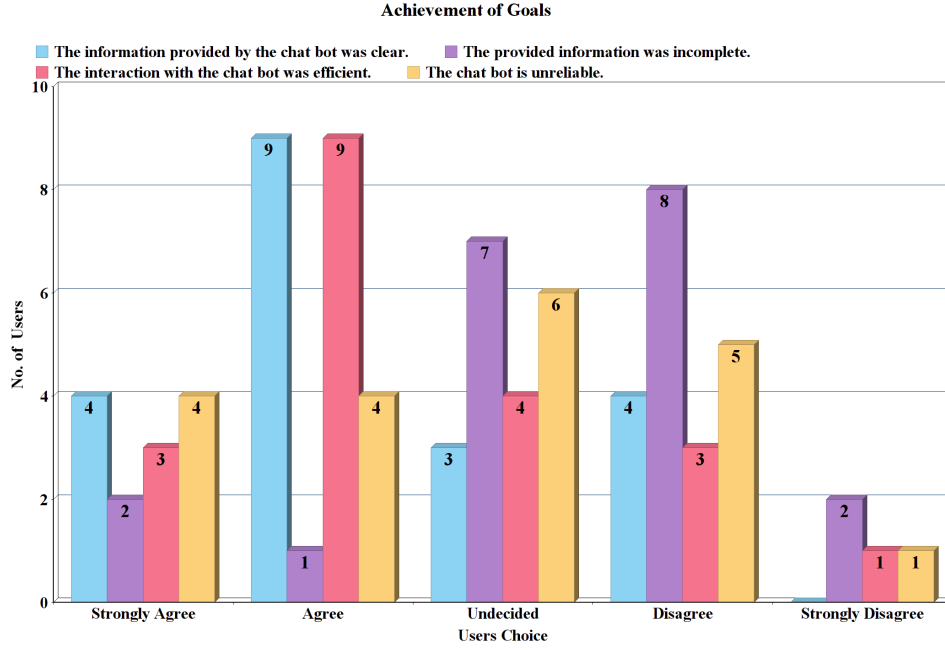


Figure 4.6.: Graphical representation of collected responses for achievement of goals.

responded falsely and a reason for it could be a weakly trained Rasa’s NLU due to limited training data as already mentioned before. But still, 50% answered that the information was complete, so there are high chances that the reason lies somewhere between (i) and (ii). Visuals have been shown in Figure 4.6.

Moving to what has been concluded from the third statement seems to be something positive. As 12(60%) of the participants agreed upon the efficient interaction with the chatbot. Moreover, 4(20%) failed to decide about it and only other remaining 4(20%) disagreed with it. Refer to Figure 4.6 for better understanding by visuals.

Lastly, forth statement stats are a bit disappointing. According to only 6(30%) users, the chatbot was reliable. On the contrary, 8(40%) declared it unreliable, and the remaining 6(30%) failed to take any decision about it. The possible reason for its unreliability could be its inability to respond to the user for all of his/her queries. And it happened due to limited data provided for training and also the demo chatbot was designed for limited use cases. To design a fully loaded chatbot that can reply to the user for any of his/her utterances, a lot of training data and processing is required and within limited time and resources, it was not possible. But, it could be done in the future. See Figure 4.6 for better visualization of the results.

4. Experiments and Evaluations

Communication with the Chat Bot

The purpose of this section was to collect the user's opinion that how was the communication with the chatbot. It consisted of the following three questions:

1. The chatbot understood my messages well.
2. I always knew what to say to the chatbot.
3. The interaction with the chatbot sounded natural.

Graphical representation for the detailed analysis and comparison of the results for these statements has been shown in Figure 4.7.

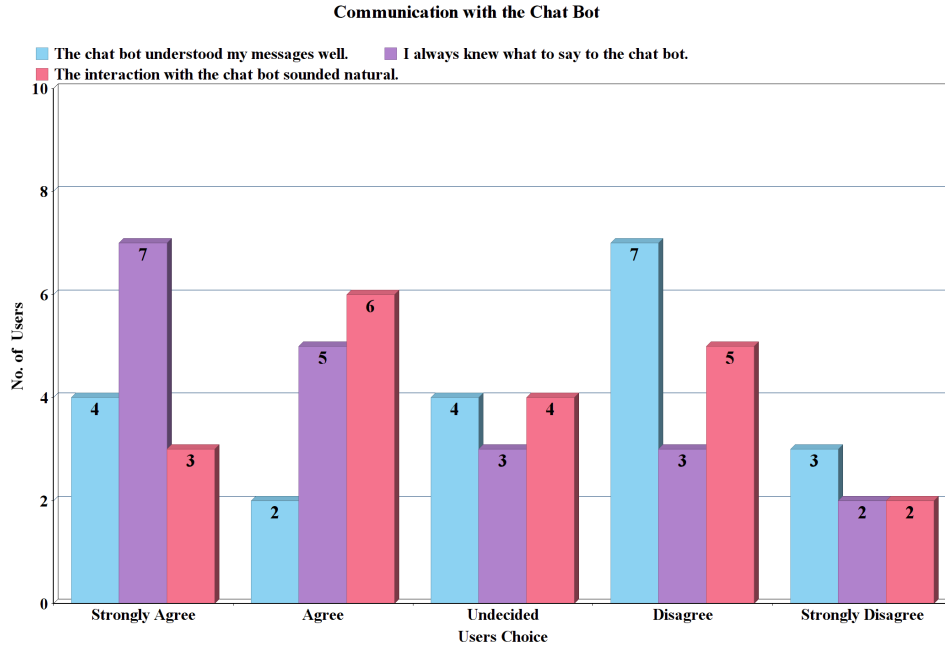


Figure 4.7.: Graphical representation of results collected for the users communication with the chat bot.

As extracted from graphical representation in Figure 4.7 that the total 10(50%) of the participants disagreed with the statement that the chatbot understood their messages well. And 4(20%) out of the remaining 10 were not able to decide about it. Remaining 6(30%) answered positively with it. A majority disagreed with the statement and a reason could be the chatbot responded incorrectly. And it could be due to a problem with natural language understanding. It has been observed while testing that NLU was detecting wrong intents for the inputted utterances. The possible justification for it could be the limited training data used for learning as already mentioned in Section

4. Experiments and Evaluations

3.2.3. And the intents used in training have been displayed in Appendix F.

Coming towards the next statement from Figure 4.7, 12(60%) of the users agreed that they always knew what to reply or ask the chatbot. Other than those, 3(15%) failed to make any decision about it, and the remaining 5(25%) disagreed with it.

Responses for the third statement from Figure 4.7 showed that 9(45%) participants felt communication as natural. Out of the other 11, only 4(20%) failed to decide about it. Along with that remaining 7(35%) showed disagreement with it. And the reason could be the wrong intent detection by NLU. It has been observed through the information collected using a log file. Secondly, manually fed responses could also be a reason for it as every time it has to select from a fixed number of answers.

Behaviour of the Chat Bot

The intention behind adding this section to the questionnaire was to detect the behavior of the chatbot with the user. It consisted of the seven questions answered by the user on the scale of agreement or disagreement as stated below:

1. The chatbot responded too slowly.
2. The chatbot is friendly.
3. The chatbot didn't always meet my expectations.
4. I didn't always know what answer the chatbot is expecting from me.
5. The chatbot made many errors.
6. I was able to recover easily from errors. (only in case of errors).
7. The chatbot behaved cooperatively.

Graphical representation for the detailed analysis and comparison of the results for these statements has been shown in Figure 4.8. Analyzing the results for this section's initial statement, 16(80%) strongly disagreed with it. Whereas, another 1(5%) also negated the statement which makes the number total 17(85%) who portrayed that the responding speed for the chatbot was fast enough as shown in Figure 4.8.

Secondly, 10(50%) rated the chatbot as friendly. While 7(35%) other respondents were unable to come up with any decision about it as displayed in Figure 4.8. It could be due to the reason that different persons have their perception of friendliness. Also, the demo topic included detective bot. So, it was meant to respond with straight forward statements. Which could be felt a bit offensive some times depending upon the user's mood and nature.

4. Experiments and Evaluations

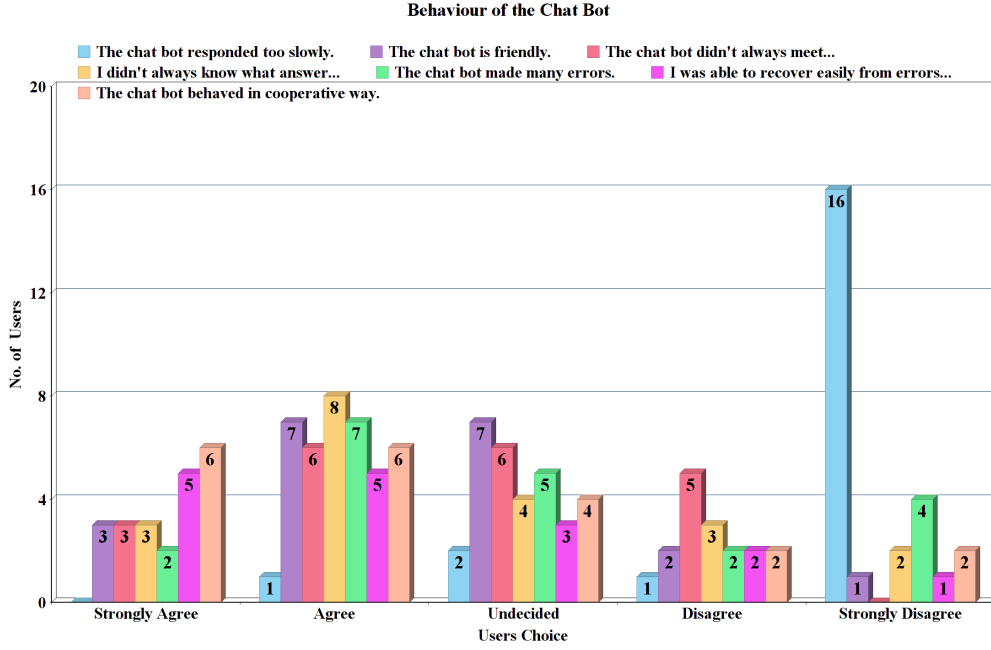


Figure 4.8.: Graphical representation of results collected for the behaviour of the chatbot.

Thirdly, the chatbot didn't meet the expectations for 9(45%) of the participants. Additionally, the other 6(30%) were failed to determine it and only 5(25%) stated that it fulfilled their expectations and can be visualized in Figure 4.8. So it can be concluded from it that the chatbot failed to impress the users by its behavior. The possible reason for it might be the same as the statement in the last section that the chatbot didn't understand the messages well or the users found it harsh or offensive.

Reviewing the results for the fourth statement, 11(55%) of the respondents showed agreement with the statement that they didn't always know what answer was the chatbot expecting from them as displayed in the Figure 4.8. It could be due to the limitation of the training data as the chatbot was just trained for limited intents but the users were provided with free choice to ask anything from the chatbot.

Checking with the participants' opinion about errors made by the chatbot and recovery from them, 9(45%) stated that chatbot made errors. While 6(30%) negated it and the remaining 5(25%) were unable to decide about it. But the positive point about the chatbot was, out of those users who faced the error or unable to decide about it during a chat, 10 of them were able to recover easily from the error. In addition to that, 12(60%) of the participants rated the chatbot as cooperative. On the contrary, just 4(20%) marked it as uncooperative as shown in Figure 4.8.

4. Experiments and Evaluations

Dialogue Assessment

This part of the survey was added to judge the design of the dialogue according to the users' perspective. A total of six questions were asked by the participants for completion of the purpose.

1. I easily lost track of where I am in an interaction with the chatbot.
2. The dialogue was bumpy.
3. I was able to direct the conversation as desired.
4. I felt in control of the interaction with the chatbot.
5. The dialogue quickly led to the desired goal.
6. The dialogue parts were evenly distributed between me and the chatbot.

Graphical representation for the detailed analysis and comparison of the results for these questions has been displayed in Figure 4.9.

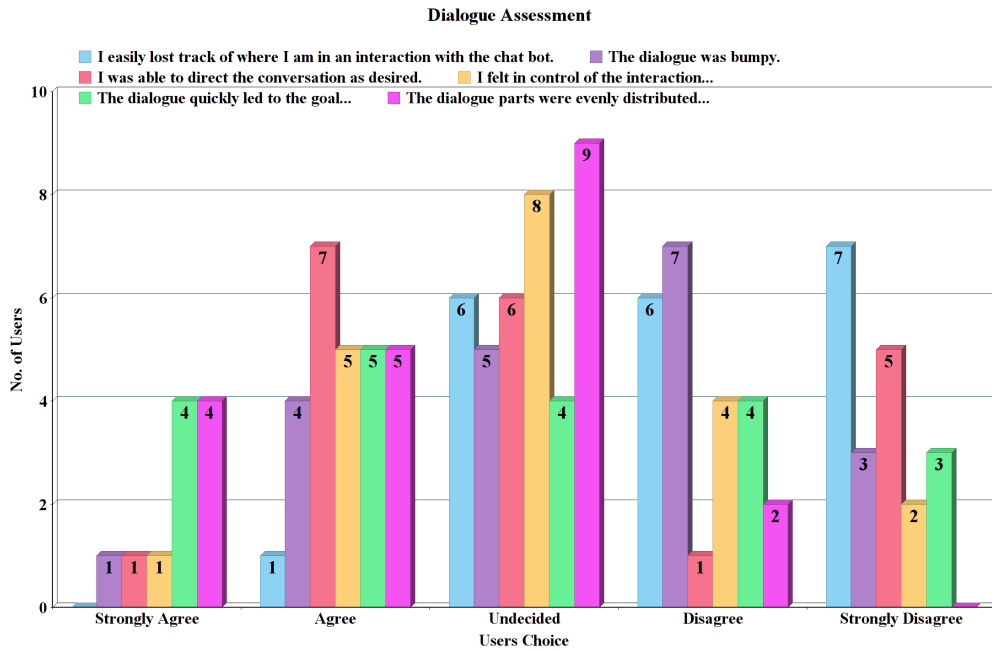


Figure 4.9.: Graphical representation of results collected for the dialogue assessment.

As plotted in Figure 4.9 that 7(35%) strongly disagreed while 6(30%) simply disagreed with the statement that they lost the track during the communication with the chatbot. On the other hand, 6(30%) were unable to make any decision and only 1(5%) just agreed

4. Experiments and Evaluations

with it. So it can be concluded from the result analysis that maintaining a state for each module concerning each user has been appeared useful and effective.

After analyzing the results from Figure 4.9, it is not wrong to say that the dialogue was smooth and communication between the users and the chatbot was comfortable and consistent. 10(50%) of the participants showed disagreement with the statement that the dialogue was bumpy. Contrarily, only half of it that is 5(25%) just find it unstable. This means by using the modular architecture a stable and smooth dialogue can be designed.

Nextly, as presented in Figure 4.9 that 8(40%) of the participants were able to direct the conversation according to their desires. On the other hand, 6(30%) out of the remaining 12 were unable to drive it according to their wish. And the left out 6(30%) answered as undecided. And yet again the ratio of the respondents who were able to direct the conversation as desired appeared to be greater than the ones who were not able to do it.

As shown in Figure 4.9, that the number of participants who agreed and disagreed with the statement that they felt in control of the conversation with the chatbot is the same and that is 6(30%) for both categories. And remaining 8(40%) were not able to decide about it. Now it is something alarming, as in this case, a possible reason could be the topic of the demo chatbot. As I designed the detective chatbot and it was designed in a way to be a bit strict and commanding.

According to Figure 4.9, 9(45%) answered that the dialogue quickly led to the desired goal which refers to the fact that the detective game was designed in such manner that the user should be able to reach the final goal as quickly as possible. But also 7(35%) disagreed with it and the cause behind it could be the wrong intent detection as the user typed in something else but NLU recognized it differently due to which chatbot responded with some unrelated statement.

Referring to Figure 4.9, 9(45%) of the people who took part in the study agreed with a statement that dialogue parts were equally distributed between them and chatbot. Whereas other 9(45%) were leave it undecided and only 2(10%) disagreed with it.

Personal Experience and Impression

This segment has been put to the questionnaire to collect users' experience and personal impressions about the chatbot. It also has been completed using a series of questions.

1. The interaction with the chatbot was pleasant.
2. I felt relaxed.
3. High level of concentration is required while using the chatbot.
4. The interaction was fun.

4. Experiments and Evaluations

5. Overall, I am satisfied with the chatbot.
6. I felt that the chatbot was smart enough to handle the message which was not lying in its scope.
7. I felt that the chatbot guided me well to return to the actual topic when I tried to misguide it.
8. It was easy for me to continue the chat without any reluctance.
9. It was easy for me to understand the response of the chatbot.
10. It took me too long to make the chatbot understand my message by using different terms in sentences.

Graphical representation for the detailed analysis and comparison of the remarks for these assertions has been shown in Figure 4.10.

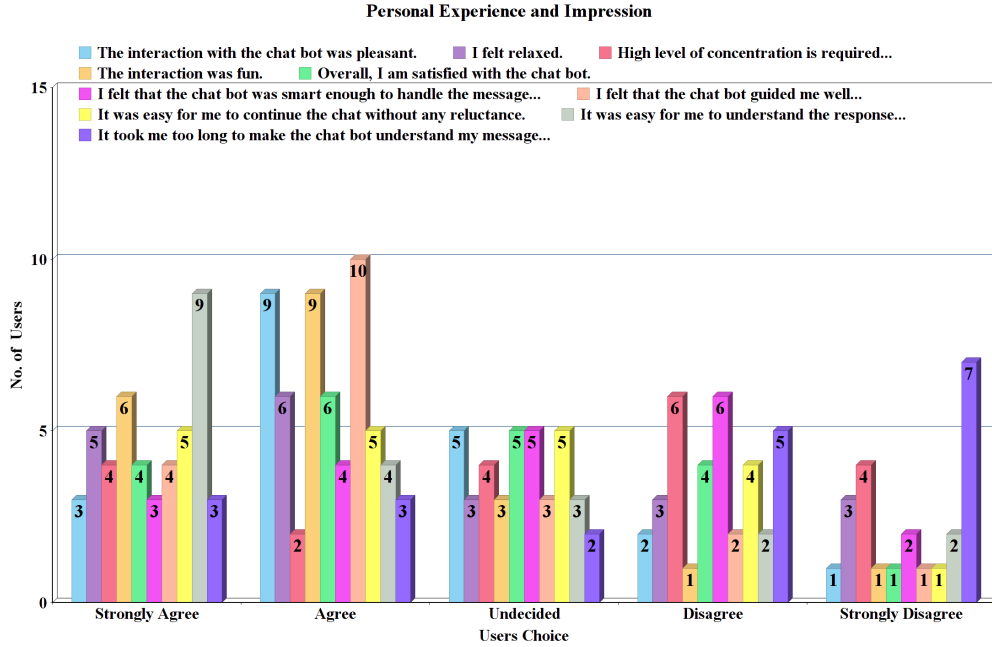


Figure 4.10.: Graphical representation of results collected for the users personal experience and impression.

As presented in Figure 4.10, the majority of the participants i.e. 12(60%) felt that the interaction with the chatbot was pleasant. It could be due to the reason that the user doesn't have to restart the chat whenever he/she wants to move to some different topic.

4. Experiments and Evaluations

The user was able to jump between different topics at any moment and can handle multiple topics at the same time without losing the state for the last topic.

According to Figure 4.10, 11(55%) respondents responded that they felt relaxed while having a conversation with the chatbot. It was something important to make sure that chatbot is not making someone feel tensed or bad about anything. Furthermore, 10(50%) of the users also showed disagreement with the statement that a high level of concentration was required while using the chatbot. So, from this result, it can be concluded that the chatbot and the dialogue design was simple and also user friendly. Additionally, 15(75%) of the users found the interaction as fun. It was also the main purpose of the Frankenbot to entertain the users instead of making them feel bored. So, this purpose also gets accomplished.

After analyzing the result from Figure 4.10 about the user satisfaction for the chatbot, it can be deduced that the majority of the users 10(50%) felt satisfied with it. While 5(25%) out of the other 10 left it undecided and the remaining 5(25%) were not satisfied with it.

Nextly, as figured out from Figure 4.10, majority 8(40%) of the participants disagreed that the chatbot was able to handle the messages well which were out of its scope. Whereas, 7(35%) of the respondents agreed with it. By surveillance of the log file based on detected intents for user utterances, it has been observed that the wrongly recognized intent could be a reason for it. Otherwise, the framework has been tested in such a way that, if on each step an utterance is provided from training data and intent has been identified correctly then its performance was up to the mark. Secondly, it has been designed to handle such a scenario well as already explained in Chapter 3 of this document. In addition to it, 14(70%) of the respondents showed agreement to the statement that chatbot guided them well to return to the actual topic when they tried to misguide it. It makes the stance clear about the chatbot abilities that it contains the skills to well manage the messages which do not lie under its scope.

Moving to the next statement, 10(50%) of the users agreed that they were able to continue the chat without any stoppage. It means the chatbot was working well for them as they wanted. The chatbot nor the user were reluctant to each other. It can also be concluded from this result that the dialogue was well structured and designed to perform smooth conversation without any objection or hesitation. Another important factor that can't be neglected is that the user should be able to understand the chatbot's response. So, 13(65%) of the participants were able to do so. Which also marked this property as accomplished for the chatbot.

Lastly, 12(60%) out of the total 20 participants negated the statement that it took them too long to make the chatbot understand their message by using different terms in sentences. It reflects that the information provided to the users was pretty much

4. Experiments and Evaluations

clear and also the guidance by the chatbot was enough for the user to enter the correct answer.

Usability

This part of the survey has been added to gather the users' opinions about the chatbot's usefulness and whether it is easy to use or not. So, it has been judged based on what users have answered the following questions:

1. The system is difficult to use.
2. It is easy to learn to use the chatbot.
3. The chatbot is too inflexible.
4. I would like to use the chatbot again in the future.
5. The chatbot operation was worthwhile.

Graphical representation for the detailed analysis and comparison of the responses for these assertions has been shown in Figure 4.11.

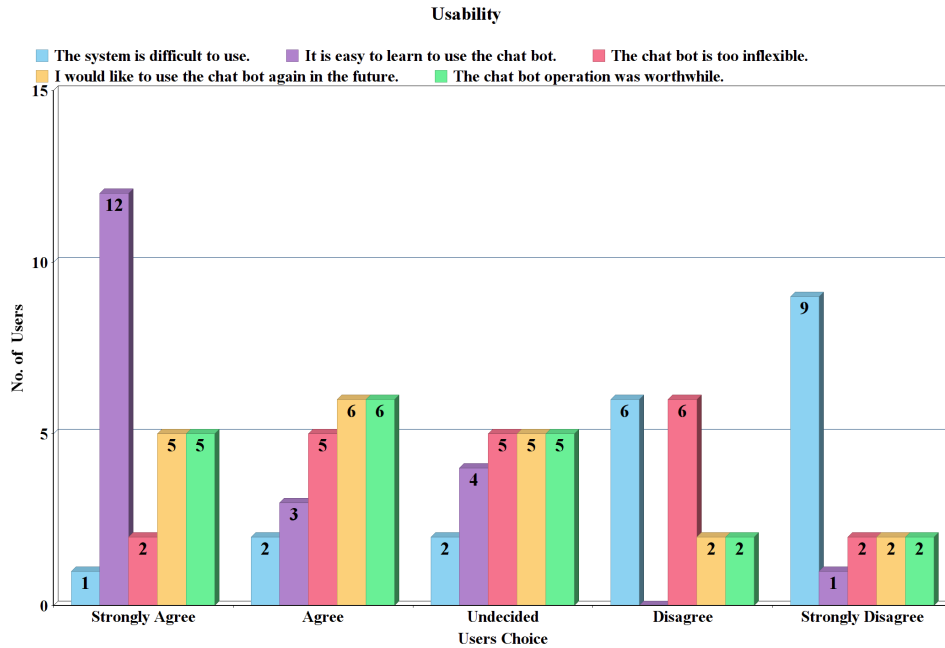


Figure 4.11.: Graphical representation of results collected for the chatbot's usability.

According to the result deduced from Figure 4.11, 15(75%) of the users contradicted with the statement that the chatbot was difficult to use. Which means the majority

4. Experiments and Evaluations

of the participants found it easy to use. Additionally, again the same amount of the respondents i.e. 15(75%) discovered that it was easier to understand the chatbot. It could be due to several reasons like the information provided was enough, the chatbot's ability to guide the user, structured dialogue, and a good understanding of the chatbot's response by the user.

Furthermore, 8(40%) of the respondents disagreed with the statement that chatbot was too inflexible. Whereas, 7(35%) showed agreement on this statement. This mixed opinion of the users could be just because of the reason that the chatbot was designed in a way that it should guide the user to return to the actual topic instead of continuing the chat on any user's desired topic. As the chatbot has been trained using limited data, time, and resources. Once, it will undergo good training then one can easily make it work for all user utterances regardless of the specific topic.

In the end, the users were asked whether they want to use the chatbot again in the future. And 11(55%) replied positively. Contrarily, only 4(20%) responded negatively. Moreover, they were also asked to rate the chatbot's functioning whether it's worth the time and effort spent. And majority i.e. 11(55%) of the participants agreed with it. On the other hand, only 4(20%) of respondents disagreed with it.

4.3.2. Evaluation via AttrakDiff

AttrakDiff's single evaluation method¹ has been used to judge the chatbot based on the following qualities:

- Hedonic quality (HQ), includes 14-word pairs which refer to the joy of use, emphasized stimulation, identification, and evocation generated by the system.
- Pragmatic quality (PQ), involves 7-word pairs that reflect the system's usefulness, efficiency, and how easy is it to use.
- Attractiveness (ATT), it is also comprised of 7 items that are being used to judge the system's pleasantness and how catchy is it according to the user's point of view.

Word Pairs

Coming to the results gathered using Attakdiff's questionnaire and also what word pairs have been used in a single evaluation method along with the mean of ratings by participants and standard deviation have been displayed in the Figure 4.12.

The AttrakDiff questionnaire contains information for the pragmatic and hedonic quality of an interactive system. The language of the study was English but the participants also had an option for German.

¹<http://www.attrakdiff.de/#tab-einsatz>

4. Experiments and Evaluations

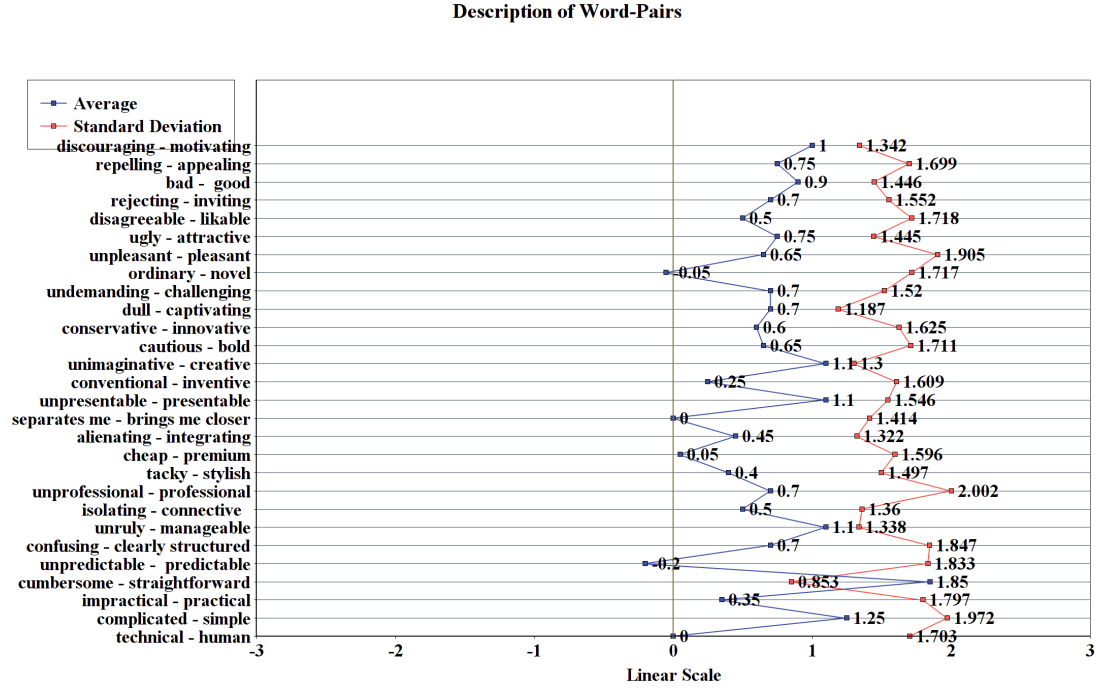


Figure 4.12.: Description of the word pairs along with the mean values and standard deviation [7].

As shown in Figure 4.12, the linear scale has been appointed the values from -3 to +3. The values -3 to -1 has been assigned to the negative aspect of the word pair e.g. "unpleasant". While 0 lies in the middle of the negative and positive aspects and referred to as neutral. Whereas, the range from +1 to +3 has been referred to the positive aspect of the word pair e.g. "pleasant".

The division of the word pairs according to their groups and classification with seven items each can also be inferred from Figure 4.12. The word pairs starting from "technical - human" till "unruly -manageable" lies under the pragmatic (PQ) category. Furthermore, word pairs from "isolating - connective" to "unpresentable - presentable" falls under the hedonic attribute group named Identification (HQ-I). Additionally, word pairs ranged from "conventional - inventive" and till "ordinary - novel" has been put under the shadow of hedonic stimulation (HQ-S). Lastly, Attractiveness (ATT) includes the set of the word pairs starting from "unpleasant - pleasant" and ending on "discouraging - motivating".

4. Experiments and Evaluations

Overall, the pragmatic quality for the chatbot rated by the participants is positive. It has been deduced from the results displayed in Figure 4.12. Respondents rated it neither technical nor human. The initial goal is achieved that at least the users didn't find it task-oriented or following some order. It is the first step towards more humanly. On the other hand, they are unable to discover their humanly characteristics. The possible reason for it could be a lack of ability to generate natural responses based on real communication. As it has been stuffed manually with a fixed number of responses. Moreover, the inadequate amount of data containing the limited number of intents (shown in Appendix F) has been used for training purposes. It could also be a reason for such feedback. Secondly, users rated it "Simple" with a positive mark greater than 1. The greatest peak at the positive side for the pragmatic section goes near to 2 for the term "Straightforward". Which means it was easy to understand and was uncomplicated. Moreover, users also found the chatbot practical, clearly structured, and manageable. The only negative point almost near to 0 has been encountered i.e. the chatbot is unpredictable. The possible reason for it could be the responses of the demo detective bot. But it has been designed like this for fun purposes. So by the results, it has been inferred that the users found the chatbot useful and assisting to achieve the desired goal.

Moving to the hedonic assessment, firstly hedonic identification ability (HQ-I) of the system has been tested using different word pairs that how the chatbot is delivering important personal values. It can also be inferred from the results portrayed in Figure 4.12 that as a whole HQ-I appeared to be positive. The participants of this study graded the chatbot as connective, professional, stylish, integrating, and highly presentable. As the word pair "cheap - premium" is answered as neutral and it could be because of the shortness, unreality, and immaturity of dialogue due to the limitations of training data and responses. The positivity deduced from it is at least a chatbot is not ranked low in quality. Contrarily, neither it is rated as prime featured may be due to the definite dialogue topics or absence of any real-time achievement. Furthermore, ranking for "separates me - brings me closer" highlights that neither the majority felt isolated nor get apart with it nor they get attracted to it. It raised a need for improvement to make it more delivering and catchy for the users. It can be inferred from the overall analysis that the chatbot accomplished its task to deliver the values that users were expecting from it and fulfilled the users' expectations. But still there exist some areas that must be subjected to improvements.

Moreover, hedonic stimulation (HQ-S) has been used to measure the chatbot's challenging ability according to the users' perspective. Overall results for it have shown a positive trend. The users found it inventive, highly creative, bold, innovative, captivating, and challenging. But the arc for the term "ordinary - novel" has been graded as neutral. As users can see many other much improvised and state of the art chatbots like Google's Assistant, Apple's Siri, and Amazon's Alexa. So, they just considered it as a normal chatbot. It could be the reason that they neither rated it as ordinary nor

4. Experiments and Evaluations

novel. But instead of that by taking overall result into an account, the users found it challenging, interesting, and fascinating.

Finally, attractiveness has been measured for the chatbot by taking the results from PQ and HQ into consideration and by using separate related word pairs for it. By the results gathered from the participants' responses, it can be stated undoubtedly that all the users found it pleasant and attractive as the values for all the terms are positive. Which also makes it likable, inviting, good, appealing, and motivating.

Average Values

According to Figure 4.13, if the average values are considered to grade the quality of the chatbot then it is not wrong to say that attractiveness(ATT) has the highest value 0.75 and standard deviation(SD) of 0.15. While pragmatic quality(PQ) has been placed at second position with the mean value of 0.72 and SD of 0.67. On the other hand for hedonic quality's race, the challenging aspect of the chatbot is leading as hedonic stimulation(HQ-S) has the mean value 0.56 with SD as 0.34. Moreover, the chatbot succeeded in delivering the important values to the users at the lowest with an average of 0.46 and SD as 0.35. As it is not wrong to say that the initial chatbot has been rated well in all aspects. But the average values lie just under the category of lower positive. None of the sections collectively crossed the mark of +1 whereas the maximum limit that can be reached is +3. So, as the initial start, these results can be considered as good. But it will not be lame to say that the chatbot must go through some more training, structuring and designing to get improved in its next version to reach the desired level for its users.

Portfolio Discussion

The dimensions for the portfolio of hedonic and pragmatic qualities have been displayed in Figure 4.14. The vertical axis represents HQ(bottom = low value) while horizontal axis depicts PQ(left = low value). In the portfolio presentation, the outer bigger light blue rectangle is the confidence rectangle. It shows how divergent are the users in evaluating the product. The bigger is the confidence rectangle, the more diverged and less confident are the gathered results. The small dark blue spot is representing the actual rating of the system. The actual value for the PQ is 0.72 and HQ is 0.51. It can be visualized that the chatbot falls into the upper neutral area for both of the dimensions (PQ and HQ). Whereas, the confidence rectangle calculated by the users' agreement shows that the confidence interval for PQ is dispersed between neutral and task-oriented character-regions. Whereas, for HQ it falls below the self-oriented region. It illustrates that there is a room for optimization in both dimensions but more likely for HQ as compared to PQ. If the hedonic quality could be raised by optimizing the natural language understanding for the chatbot using enriched and sufficient training data. Then a user can communicate with it in a much better way and the chatbot can also be able to deliver its best and users could find it more joyful and delivering. Although, pragmatically it is touching

4. Experiments and Evaluations

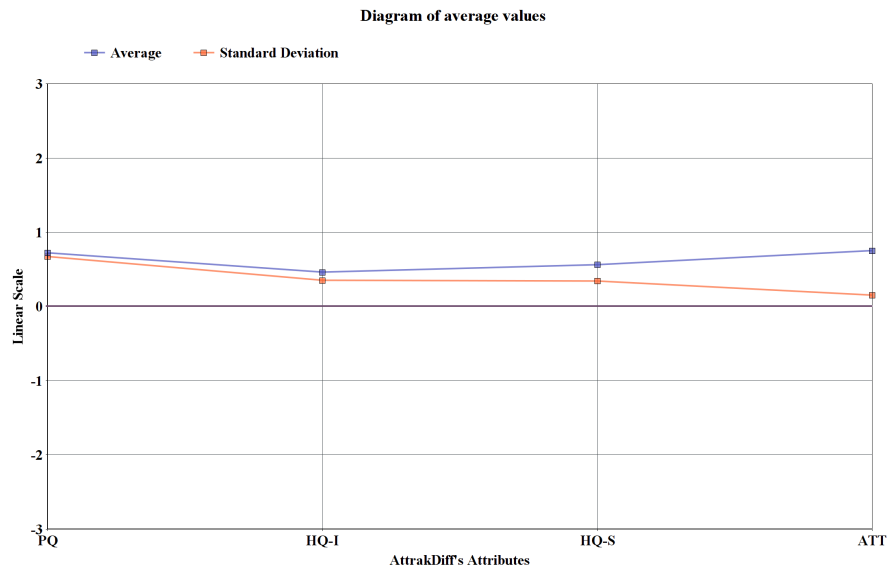


Figure 4.13.: Average values for PQ, HQ-I, HQ-S and ATT [7].

the task-oriented area for now but still needs to be elevated by improving the provided assistance for the users to achieve their desired goals. By undergoing such enhancements it can reach the region of desired characteristics.

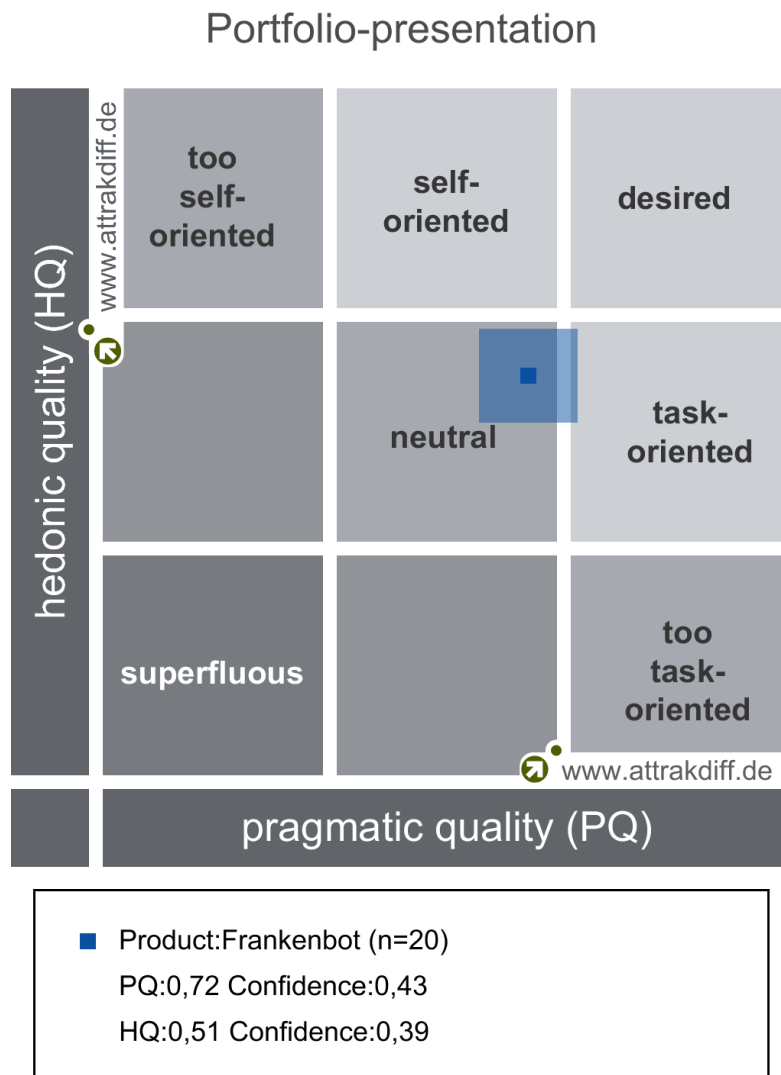


Figure 4.14.: Portfolio of the results for HQ and PQ of the chatbot [7].

4.4. Qualitative Analysis

This section illustrates the incidents and behaviors that have been observed during the accomplishment of this research study. Some of the following characteristics have been noticed and reported in a verbal interview conducted by consulting 12 actors out of a total of 20 participants.

4. Experiments and Evaluations

4.4.1. Natural Language Understanding(NLU)

It has been examined while performing self-testing on it that when the chatbot was inputted with the longer utterances, it was not able to identify the correct intent. Also whenever it has been asked for something that it is not trained for, the NLU was detecting the wrong intent. It is a real problem that could be a cause of confusion for the users and could also have made them lose their interest.

Out of total 12, 8 of the interviewed participants started the interaction with a greeting message and it worked well as the chatbot was designed so. But the remaining 4 tried to start the communication with any other statement and they reported that the chatbot was smart enough to guide them that how to start the conversation. All of them revealed that once the dialogue has been initiated they tried to enter longer utterances but overtime shortened their sentences due to irrelevant responses. And with shorter and simple utterances the chatbot produced relevant answers. The possible reason behind it is the NLU training using limited data.

10 out of a total of 12 interrogated users changed terms in a sentence to make the chatbot understand the correct semantics of their statements. But in the end, all of them were able to continue with the chat.

4.4.2. Chatbot Guidance and Intelligence

Also, 8 users reported that they were unable to move to the next question while playing a detective game but chatbot guided them well to make them understand what sort of input was it expecting from them at that specific moment. So, it can be deduced from it that the chatbot's ability to guide the user was good enough about the stuff for what it has not been trained or not expecting at some specific occasion. It was also able to deliver and to make the users understand its responses well. Furthermore, 7 of the users notified that they tried to deviate the chatbot from the topic but the detective didn't get diverted and forced them using convincing responses to put them back on track.

4.4.3. Interaction

A total of 12 participants were asked for feedback about their interaction with the chatbot. And 11 of them replied that it was fun and they enjoyed the communication with the chatbot. They also answered that the jokes it was cracking were joyful and made them laugh. Other than that they also liked the sarcasm and strictness in the responses of the detective bot as it was designed for making users feel that they are talking to some real detective. Only 1 out of 12 mentioned that the chatbot's answers were rude to some extent but when he was provided the reason for it then he realized and understood it. Conclusively, all the users who have been asked about the dialogue and the interaction with the chatbot loved it.

4. Experiments and Evaluations

4.4.4. Switching Topics

As it was mentioned in the description of the chatbot to the users that the chatbot can talk about parallel topics at the same time. The users have to switch the topic and come back to the previous topic by answering the last question of that topic at any moment in the future. When the users have been asked about it in verbal conversation, all(12) of them mentioned that it worked fine and they liked and appreciated this feature.

4.4.5. Detective Game

The same 12 participants have also been questioned to gather feedback about detective games whether they liked it or not. To be precise, 6 out of them liked it and the other 4 complaints about the short length of the game and the scenario were not directed well. In addition to it, the remaining 2 found it as a source of recreation for them. Cumulatively, all of them liked the idea of the detective game as a demo. Also, the users have been requested to end the game before moving to the completion of the surveys. And all of them replied positively on asking whether they reached an end of the game before filling out the questionnaires.

4.4.6. User Interface

9 out of the total 12 participants liked the interface. But the remaining 3 of them disliked it due to the dark theme as they recommended the light-colored scheme for it. Otherwise, all of the users liked the design and style of an interface.

5. Discussion and Conclusion

This is the final chapter for this master's thesis which summarizes it. Additionally, it also discusses the limitations of the framework stated in Chapter 3 and evaluation methodologies along with the gathered results illustrated in Chapter 4. Lastly, possible future work has been manifested.

5.1. Summary

The motivation of this master's thesis was to implement a framework to enhance the development of conversational interfaces using a novel modular architecture discussed in Chapter 3 of the document. After its implementation, it has to be evaluated based on the users' experience and quality. This has been accomplished by crafting and implementing the "Frankenbot", a modular architectural virtual conversational agent that communicates with the users and acts as a detective to investigate a robbery. This conversational agent then has been judged based on the opinions collected by the users who played with it.

Chapter 2 explains all the foundations and related work. Starting from the history and overview of the chatbots to their tasks and components. Furthermore, dialogue systems along with the existing state of the art frameworks have also been mentioned. In addition to that, Dialogue Management Systems(DMS) have been discussed in detail along with their challenges and evaluation methods.

In Chapter 3, the design, and implementation of the modular chatbot(Frankenbot) have been explained in detail. The Frankenbot is made up of a client(user interface) and a backend(webserver) implemented in Python. And a client communicates with a server using Web API implemented using python's library named as Flask. For intent and entities detection, the framework for natural language understanding(NLU) named as RASA has been used after feeding and training it using a data source in JSON format. The training data provided to it was just for the demo detective game along with few other topics like humor, bot's profile, and gossips, etc. The web server receives the user utterance from the client using web API and does further processing accordingly which involves intent and entities detection using already trained Rasa's NLU Model. Once the intent has been detected the chatbot completes the remaining essential tasks granted to itself by its modular behavior and finally produces a suitable response for the user. This response is sent back to the user and can be viewed on an interface that he/she has already used to send a request.

5. Discussion and Conclusion

Chapter 4 contains the research study that was developed and conducted to gather the results about the user's experience and the system's quality of the Frankenbot.

The practical results determined in Chapter 4 exposes that the users' overall impression about the chatbot designed using modular architecture is good. Additionally, pragmatic and hedonic qualities along with the attractiveness of the system have been rated fair by the users but still there exists room for improvement. Moreover, the qualitative analysis highlights the users' experience and problems with the framework. It also provides feedback about the framework's new added feature, client, and the detective game.

5.2. Discussion

This section formally discusses the limitations and positive aspects of the implemented framework and approaches used for its evaluation.

5.2.1. Framework

There exist several states of the art frameworks for the chatbots but to the best of my knowledge, none of them is following the modular architecture introduced during this research study. It has several advantages such as enhanced usability, simpler tree structure, and connection between tree nodes, minimal redundancy as a module has to be declared only once. Other than that, it can act as a unified framework for different technologies. As a module can be more than a dialogue tree. As a theory, other systems (question answering, neural systems, etc.) can also get fit in this framework as long as they implement an activation function. And there exist different methods to implement an activation function over the modules.

1. Each module calculates its activation independently based on the added modules in the chatbot.
 - Pro: Easy and liberal implementation of modules.
 - Con: Comparing the confidence values of different modules can lead to errors.
2. All modules use the same intent recognition module. Therefore the system can calculate a joint intent recognition overall modules.
 - Pro: Better results are expected.
 - Con: Module-specific NLU is not possible. Combinations of different NLU strategies is limited.
3. NLU Chain: First an option 1 NLU is applied. If there is no match then the NLU of option 2 will be used.

5. Discussion and Conclusion

Another advantage of using the framework following the modular architecture is that each module manages its state. Therefore, it is straightforward to switch between modules and come back to the last state. These were some important technical facts that need to be addressed here.

Along with the positive aspects of it, several limitations have been encountered during its implementation. For intent and entity recognition, the RASA NLU component has been used. It appeared to be a challenging task as currently it has been fed with limited related data only as shown in Appendix F. But it needs a large data set for training purposes to perform well. Secondly, dialogue designing needs well-structured JSON data to be processed further by a dialogue manager. Moreover, the chatbot has been designed using a dialogue tree structure for its simplicity. It means a module should be designed in the form of a dialogue tree. Due to which it undergoes few constraints that a node can only be a child of only one different node. In other words, one node can only have one parent node assigned to it. Due to this limitation, one module can not be appointed as a child to two different modules within the same or different chatbots. Furthermore, large memory storage will be required to save the current state of each user for all the activated modules within complex dialogues. Depending upon the complexity of dialogue it may take longer to detect an intent as it has to process all the available modules separately. The selected intent would be the one with the highest confidence value. Lastly, only intent specific responses are currently available for the users.

5.2.2. Evaluation

The evaluation has been accomplished with the help of following methodologies: (i) Frankenbot's Experience Survey (Appendix H.1), (ii) Frankenbot's Evaluation via AttrakDiff (Appendix H.2) and (iii) Interview (Section 4.2.3). Several limitations have been faced during the evaluation process. Firstly, only 20 participants were allowed to fill the AttrakDiff's survey by default and this limit has been set by tool designers for using AttrakDiff's Single Evaluation Technique. For the other survey as it has been designed using Google Sheet and there was no such constraint on the number of participants. But to maintain the same number of participants, both of the surveys have been shared with the same 20 participants. While only 12 out of them have appeared for an interview. Secondly, due to the current pandemic, it was not possible to have an in-person meeting with the participants. So, for this reason, they have been contacted using social media platforms. It was also the cause of hindrance in the process. Additionally, the chatbot has been evaluated containing only two modules designed for demo Frankenbot i.e. detective and general as shown in Appendix F. And it has been judged for the user experience, communication, usability, design, qualities (Hedonic and Pragmatic), and framework's modular state handling. But it should also be evaluated and compared with any existing state of the art dialogue framework for determining its true potential.

Heading towards the collected evaluation results and stats, the users and participants of

5. Discussion and Conclusion

the surveys liked the chatbot as a majority of the respondents rated its overall impression as good. Also, AttrakDiff’s study has shown that this approach is practical and initial rating about the system’s quality collected according to the users’ perspective is also worthwhile. Users also appreciated the feature of parallel topic handling and coming back and forth to different topics at the same time without re-initiating the chatbot. There can be multiple modules within the same chatbot and each module maintains a separate state for each user which makes this feature successful.

Contrarily, the experiments and evaluation also exposed the deficiencies in this initially designed and implemented framework. These problems can be categorized as technical problems as they occurred due to the misbehavior of the system and didn’t meet the users’ expectations. Secondly, some conceptual problems have also been noticed.

A common technical problem occurred was the useless or irrelevant response generated by the framework to the user utterance. And that happened just because of wrongly detected intent by the NLU. And the NLU caused this problem just because of limited resources and data available for training as shown in Appendix F. Another problem encountered was again associated with NLU as if user types any meaningless word then the NLU still provides some result in the form of the detected intent. Lastly, one conceptual problem was noticed by the users that the chatbot was not responding with a meaningful answer if the user utterance is a long sentence with some additional information that is not required at that moment in a conversation. But sometimes the chatbot responded correctly as well. This problem raised due to the misleading caused by incorrectly recognized intent by the NLU. But with the continuous usage, users shortened their inputs by providing only necessary terms required by the chatbot and the NLU worked as expected. This indicates that the system needs to be improved for natural and humanly conversation.

As it is just a simple demo chatbot designed under various limitations of time and resources but still able to capture positiveness out of the users. So, if the framework will undergo proper training and design then it might be a revolutionary transformation for the chatbots.

5.3. Future Work

The results have demonstrated that the modular architecture implemented in the Frankenbot got successful and rated comprehensively good by the users. But it still needs some optimization to enhance the user experience and pragmatic and hedonic qualities of the system to reach a desirable mark for it. The solutions for deficiencies listed in Section 5.2 are consigned here.

To boost the user experience and the abilities of the Frankenbot, it is an essential need to improve the training for natural language understanding. And it could be done using

5. Discussion and Conclusion

the efficient and populous data set. This adjustment could be helpful for the advancement of natural language understanding for any type of user utterances and statements imposed on the chatbot. Secondly, it can also be enhanced by applying live training using the data collected during the dialogue between the chatbot and the user. But it can lead to miscellaneous results. So, to make it work a developer should be responsible to check whether the detected intent for the user input was correct and then add it to the training data. Additionally, the Rasa framework for NLU has been utilized for this purpose in the current version of the chatbot. But if there will be any other better and more advanced framework developed in the future then it can also be replaced for a better understanding of the semantics for the user utterances.

Activation for the modules could be calculated using two of the following approaches independently: (i) each module calculates its activation independently based on the added modules in the chatbot. (ii) All modules use the same intent recognition module. Therefore the system can calculate a joint intent recognition overall modules. But for better results for intent recognition the hybrid method "NLU Chain" could also be implemented as stated in Section 5.2.

The dialogue graphs could also be implemented instead of dialogue trees. As the graph data structure supports the multiple parents' feature which is not allowed in a tree. But as a reminder, the main purpose of the modular architecture was also to keep the framework simple to minimize the transitions between the nodes as discussed in section 3.4.2. By using graphical structure it could become more complex.

The performance can also be raised by adding a feature of response generation using identified entities along with the recognized intent for the user utterance. By adding the support for this characteristic the chatbot could be able to produce more precise and relevant responses for the users' statements.

To measure the real potential and capabilities of the framework implemented in this master's thesis, it would be great if it gets compared with existing advanced frameworks like IBM's Watson Assistant, etc. It could be done by designing the same dialogue using Frankenbot's framework and existing dialogue frameworks. Afterward, the dialogue design and structure should be compared to check the simplicity and module usability for both of them. Also, the same user utterance should be inputted to both of them to observe differences between the results. Secondly, evaluation from the users' perspective is always a great idea. One could also just design the same dialogue using a modular dialogue manager and any other state of the art dialogue manager. Once it has been done, the evaluation strategies already mentioned in this study can also be used for the assessment of both of the dialogue frameworks.

Bibliography

- [1] IBM Watson Assistant. <https://www.ibm.com/cloud/watson-assistant/docs-resources/>. Retrieved: 2020-01-24.
- [2] Getting Started with Rasa. <https://rasa.com/docs/getting-started/>. Retrieved: 2020-01-24.
- [3] Plato Research Dialogue System. <https://github.com/uber-research/plato-research-dialogue-system>. Retrieved: 2020-01-24.
- [4] Florian Peters. Design and implementation of a chatbot in the context of customer support. Master's thesis, University of Liège, 2018.
- [5] J. Harms, P. Kucherbaev, A. Bozzon, and G. Houben. Approaches for dialog management in conversational agents. *IEEE Internet Computing*, 23(2):13–22, 2019.
- [6] Martin Schrepp, Theo Held, and Bettina Laugwitz. The influence of hedonic quality on the attractiveness of user interfaces of business management software. *Interacting with Computers*, 18:1055–1069, 09 2006.
- [7] AttrakDiff. <http://www.attrakdiff.de/index-en.html>. Retrieved: 2020-05-16.
- [8] Dan Jurafsky and James H. Martin. Dialogue systems and chatbots. In *Speech and Language Processing*, chapter 26, pages 487–515. 2019.
- [9] Jack Cahn. Chatbot: Architecture, design, & development. *University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science*, 2017.
- [10] Rashid Khan and Anik Das. *Build Better Chatbots*. Apress, 2018.
- [11] Petter Brandtzaeg and Asbjørn Følstad. Chatbots: changing user needs and motivations. *Interactions*, 25:38–43, 08 2018.
- [12] A. M. Rahman, A. A. Mamun, and A. Islam. Programming challenges of chatbot: Current and future prospective. In *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, pages 75–78, 2017.
- [13] Bayan Abu Shawar and Eric Atwell. Chatbots: Are they really useful? *LDV Forum*, 22:29–49, 2007.

Bibliography

- [14] Wlodek Zadrozny, M. Budzikowska, J. Chai, N. Kambhatla, S. Levesque, and N. Nicolov. Natural language dialogue for personalized interaction. *Commun. ACM*, 43(8):116–120, August 2000.
- [15] Vinícius F Galvão, Cristiano Maciel, and Ana Cristina Bicharra Garcia. Creating chatbots to talk with humans: Hci evaluations and perspectives. pages 1–11, 2019.
- [16] Types of bots: In Overview Botnerds. <http://botnerds.com/types-of-bots/>. Retrieved: 2020-01-24.
- [17] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A survey on dialogue systems: Recent advances and new frontiers. *SIGKDD Explor. Newsl.*, 19(2):25–35, November 2017.
- [18] Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue & Discourse*, 9(1):1–49, 2018.
- [19] Elahe Paikari and André Van Der Hoek. A framework for understanding chatbots and their future. In *2018 IEEE/ACM 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 13–16. IEEE, 2018.
- [20] G Churcher, ES Atwell, and C Souter. *Dialogue management systems: a survey and overview*. University of Leeds, School of Computing Research Report 1997.06. 1997., 1997. Churcher, G, Atwell, ES and Souter, C (c) 1997, University of Leeds. Reproduced with permission from the copyright holders.
- [21] Michael F. McTear. Spoken dialogue technology: Enabling the conversational user interface. *ACM Comput. Surv.*, 34(1):90–169, March 2002.
- [22] David Griol, Zoraida Callejas, and Ramón López-Cózar. Statistical dialog management methodologies for real applications. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL ’10, page 269–272, USA, 2010. Association for Computational Linguistics.
- [23] ITU. Subjective quality evaluation of telephone services based on spoken dialogue systems. *Recommendation P.851 (11/03)*, 2003.
- [24] M McTear, Z Callejas, and D Griol. The conversational interface: Talking to smart devices: Springer international publishing. *Doi: <https://doi.org/10.1007/978-3-319-32967-3>*, 2016.
- [25] Gokhan Tur and Renato De Mori. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons, 2011.
- [26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Bibliography

- [27] Lian Meng and Minlie Huang. Dialogue intent classification with long short-term memory networks. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 42–50. Springer, 2017.
- [28] Steve Young. Semantic processing using the hidden vector state model. *Computer Speech Language*, 19:85–106, 01 2005.
- [29] Asa Ben-Hur and Jason Weston. A user’s guide to support vector machines. *Methods in molecular biology (Clifton, N.J.)*, 609:223–39, 01 2010.
- [30] Truyen Tran. On conditional random fields: applications, feature selection, parameter estimation and hierarchical modelling. 01 2008.
- [31] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018.
- [32] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [33] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.
- [34] S. Dipanjan and RedHat. Implementing Deep Learning Methods and Feature Engineering for Text Data: The Skip-gram Model. <https://www.kdnuggets.com/2018/04/implementing-deep-learning-methods-feature-engineering-text-data-skip-gram.html>. Accessed: 2020-01-27.
- [35] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.
- [36] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [37] Aniruddha Tammewar, Monik Pamecha, Chirag Jain, Apurva Nagvenkar, and Krupal Modi. Production ready chatbots: Generate if not retrieve. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [38] Kyusong Lee, Hongsuck Seo, Junhwi Choi, Sangjun Koo, and Gary Geunbae Lee. Conversational knowledge teaching agent that uses a knowledge base. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 139–143, 2015.

Bibliography

- [39] Zhou Yu, Ziyu Xu, Alan W Black, and Alexander Rudnicky. Strategy and policy learning for non-task-oriented conversational systems. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 404–412, Los Angeles, September 2016. Association for Computational Linguistics.
- [40] Alicia Abella, Michael K. Brown, and Bruce Buntschuh. Development principles for dialog-based interfaces. In Elisabeth Maier, Marion Mast, and Susann LuperFoy, editors, *Dialogue Processing in Spoken Language Systems*, pages 141–155, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.
- [41] Hans Dybkjær, Laila Dybkjær, and Niels Ole Bernsen. Design, formalization and evaluation of spoke language dialogue. *Andernach et al.(eds.)*, 1995.
- [42] Bayan Abu Shawar and Eric Atwell. Different measurement metrics to evaluate a chatbot system. In *Proceedings of the workshop on bridging the gap: Academic and industrial research in dialog technologies*, pages 89–96, 2007.
- [43] How do you feel? Understanding emotions to craft satisfying experiences. <https://www.keepitusable.com/blog/tag/hedonic-qualities/>. Retrieved: 2020-06-12.
- [44] Frankenstein’s Monster. https://en.wikipedia.org/wiki/Frankenstein%27s_monster. Retrieved: 2020-04-11.
- [45] Frankenstein (1931 film). [https://en.wikipedia.org/wiki/Frankenstein_\(1931_film\)](https://en.wikipedia.org/wiki/Frankenstein_(1931_film)). Retrieved: 2020-04-11.
- [46] Request/Response Pairs. <https://techdocs.broadcom.com/content/broadcom/techdocs/us/en/ca-enterprise-software/devops/devtest-solutions/10-3/using/using-ca-service-virtualization/ca-service-virtualization-concepts/request-response-pairs.html>. Retrieved: 2020-06-07.
- [47] Microsoft: Define conversation steps with waterfalls. <https://docs.microsoft.com/en-us/azure/bot-service/nodejs/bot-builder-nodejs-dialog-waterfall?view=azure-bot-service-3.0>. Retrieved: 2020-06-07.
- [48] DFKI: Slot-filling. <http://www.dfki.de/etai/SpecialIssues/Dia99/araki/araki/node3.html>. Retrieved: 2020-06-07.
- [49] Wikipedia: Dialogue tree. https://en.wikipedia.org/wiki/Dialogue_tree. Retrieved: 2020-06-07.
- [50] Usability first: Question and Answer Dialogue. <https://www.usabilityfirst.com/glossary/question-and-answer-dialogue/index.html>. Retrieved: 2020-06-07.
- [51] What is a Knowledge Graph?, howpublished = <https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/>, note = Retrieved: 2020-06-07.

Bibliography

- [52] Flask API. <https://flask.palletsprojects.com/en/1.1.x/api/>. Retrieved: 2020-04-17.
- [53] RASA Training Data Format. <https://rasa.com/docs/rasa/nlu/training-data-format/>. Retrieved: 2020-04-19.
- [54] Choosing a Pipeline. <https://rasa.com/docs/rasa/nlu/choosing-a-pipeline/>. Retrieved: 2020-04-18.
- [55] Bootsnip Simple Chat. <https://bootsnipp.com/snippets/y8e4W>. Retrieved: 2020-04-17.
- [56] MDN Web Docs: Window.localStorage. <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>. Retrieved: 2020-04-17.
- [57] Any Tree: Node Classes. <https://anytree.readthedocs.io/en/latest/api/anytree.node.html>. Retrieved: 2020-04-19.
- [58] Logging HOWTO. <https://docs.python.org/3/howto/logging.html>. Retrieved: 2020-05-01.
- [59] Sebastian Möller, Paula Smeele, Heleen Boland, and Jan Krebber. Evaluating spoken dialogue systems according to de-facto standards: A case study. *Computer Speech Language*, 21(1):26 – 53, 2007.
- [60] ITU-T Recommendations. <http://handle.itu.int/11.1002/1000/7043>. Retrieved: 2020-05-16.
- [61] AttrakDiff’s Single Evaluation. <http://www.attrakdiff.de/#tab-einsatz>. Retrieved: 2020-05-16.
- [62] Thilo Michael, Stefan Hillmann, and Benjamin Weiss. Alex: An artificial conversational agent for students at the tu berlin. In Jürgen Trouvain, Ingmar Steiner, and Bernd Möbius, editors, *Studentexte zur Sprachkommunikation: Elektronische Sprachsignalverarbeitung 2017*, pages 238–245. TUDpress, Dresden, 2017.
- [63] Marc Hassenzahl, Michael Burmester, and Franz Koller. *Attrakdiff: A questionnaire to measure perceived hedonic and pragmatic quality*, pages 187–196. Vieweg+Teubner Verlag, Wiesbaden, 2003.

Appendices

A. Hyperlink to Deployed Frankenbot

URL to deployed Frankenbot: <https://www.dfki.de/frankenbot/shahid/index.html>

B. Information for Participants

I am the detective chat bot designed to interrogate you about the armed robbery happened few days back at a spatkauf near Berliner Strasse. I am responsible for its investigation, so I am going to ask you some questions to come up with a decision. It will include, what I have investigated so far. Also you can have general conversation with me like you can ask me to talk about general stuff with you, about corona virus and its stats, to make you laugh, to do gossips, how do I feel, who am I, what do I eat and other related questions about me. You can switch between the topics at any moment as I am designed for parallel handling of the multiple topics. Kindly, chat at least with me until I reach any decision or say you goodbye before you jump to completion of the surveys. Thanks!

Note: If you ever lost where were you and want to reinstate the detective game then just send me a greeting message (hi, hello etc.) and I will restart it for you.

C. Instructions for Participants

Just think like you are accused of a robbery and you are sitting in front of a detective and you have to answer his brutal and tricky questions in order to prove your innocence otherwise, you will be declared a culprit. It is a responsibility of a detective to make a decision based on your answers to his questions.

Note: It is the fictional detective chat bot designed only for testing and fun purpose. Have fun while staying safe at home :)

D. Request for Participants

Once, you are finished having fun with me. Please don't forget to complete following assessments provided as the links below. They will be activated after 1 minute, meanwhile you should get familiar with me by having a small chat with me. I welcome and

highly appreciate your prestigious feedback which will help my developer to evaluate me and enhance my abilities for your better experience. Thanks in advance :)

E. Frankenbot's Dialogue Structure (frankenbot.json)

URL to Frankenbot Dialogue Structure: https://bitbucket.org/nordpolemil/modular-chatbot/src/5ea57ec/frankenbot-framework/bot_def/detective/frankenbot.json?fileviewer=file-view-default

F. RASA NLU Training Data Stats

F.1. Module 1(detective_tree.json)

- intent examples: 115 (6 distinct intents)
- Found intents: '#negation', '#purchasing', '#bye', '#affirm', '#robbery_time.info', '#greet'
- entity examples: 2 (1 distinct entities)
- found entities: '@place'
- *URL to Training Data for Detective Module:* https://bitbucket.org/nordpolemil/modular-chatbot/src/5ea57ec/frankenbot-framework/bot_def/detective/rasa/detective_tree.json?fileviewer=file-view-default

F.2. Module 2(general_tree.json)

- intent examples: 205 (9 distinct intents)
- Found intents: '#humor', '#botFood', '#gossip', '#botProfile', '#psychology', '#emotion', '#coronaStats', '#general_talk', '#corona'
- entity examples: 41 (1 distinct entities)
- found entities: '@feeling'
- *URL to Training Data for General Module:* https://bitbucket.org/nordpolemil/modular-chatbot/src/5ea57ec/frankenbot-framework/bot_def/detective/rasa/general_tree.json?fileviewer=file-view-default

G. RASA Configuration File(config_spacy.yaml)

```
1 language: "en_core_web_sm"
2 pipeline:
3   - name: "SpacyNLP"
4     - name: "WhitespaceTokenizer"
5     - name: "SpacyTokenizer"
6     - name: "SpacyFeaturizer"
7     - name: "SklearnIntentClassifier"
8     - name: "CRFEntityExtractor"
9     - name: "EntitySynonymMapper"
```

Listing 1: RASA Pipeline for Frankenbot.

URL to a file: https://bitbucket.org/nordpolemil/modular-chatbot/src/5ea57ec/frankenbot-framework/bot_def/detective/rasa/config_spacy.yaml?fileviewer=file-view-default

H. Surveys

H.1. Frankenbot's Experience Survey

Frankenbot's Experience Survey

This questionnaire is designed to collect the user experience for Frankenbot. The results collected by conduction of this survey will be evaluated and discussed in final report for my masters thesis.

* Required

1. Email address *

Overall Impression of the Interaction

2. What is your overall impression about the chat bot? *

Mark only one oval.

	1	2	3	4	5	
Excellent	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Bad

Familiarity with Existing Chat Bots.

Purpose of this section is to get the knowledge that how familiar are you with chat bots.

3. I feel that I am well familiar with the chat bots like Google Assistant, Apple's Siri etc.

*

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

4. I communicate with the chat bots: *

Mark only one oval.

- ☐ Frequently (daily or several times in a week)
- ☐ Seldomly (rarely in weeks or months)
- ☐ Just few times
- ☐ Never

5. Purpose of my chat bots usage is: (only if you answered last question positively)

Mark only one oval.

- ☐ Personal commands to provide you assistance in performing tasks
- ☐ For fun
- ☐ Not feel lonely
- ☐ No reason
- ☐ Other: _____

Achievement of Goals

6. The information provided by the chat bot was clear. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

7. The provided information was incomplete. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

8. The interaction with the chat bot was efficient. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

9. The chat bot is unreliable. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

Communication with the Chat Bot

10. The chat bot understood my messages well. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

11. I always knew what to say to the chat bot. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

12. The interaction with the chat bot sounded natural. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

Behaviour of the Chat Bot

13. The chat bot responded too slowly. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

14. The chat bot is friendly. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

15. The chat bot didn't always meet my expectations. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

16. I didn't always know what answer is the chat bot expecting from me. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

17. The chat bot made many errors. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

18. I was able to recover easily from errors. (only in case of errors)

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

19. The chat bot behaved in cooperative way. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

Dialogue

20. I easily lost track of where I am in an interaction with the chat bot. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

21. The dialogue was bumpy. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

22. I was able to direct the conversation as desired. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

23. I felt in control of the interaction with the chat bot. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

24. The dialogue quickly led to the desired goal. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

25. The dialogue parts were evenly distributed between me and the chat bot. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

Personal Impression

This section includes the questions to evaluate your experience about Frankenbot.

26. The interaction with the chat bot was pleasant. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

27. I felt relaxed. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

28. High level of concentration is required while using the chat bot. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

29. The interaction was fun. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

30. Overall, I am satisfied with the chat bot. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

31. I felt that the chat bot was smart enough to handle the message which was not lying in its scope. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

32. I felt that the chat bot guided me well to return to the actual topic when I tried to misguide it. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

33. It was easy for me to continue the chat without any reluctance. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

34. It was easy for me to understand the response of the chat bot. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

35. It took me too long to make the chat bot understand my message by using different terms in sentences. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

Usability

36. The system is difficult to use. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

37. It is easy to learn to use the chat bot. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

38. The chat bot is too inflexible. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

39. I would like to use the chat bot again in the future. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

40. The chat bot operation was worthwhile. *

Mark only one oval.

	1	2	3	4	5	
Strongly agree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly disagree

User
Information

It will be just used to make comparisons between different age groups and their technical knowledge.


41. Age *

42. Profession *

This content is neither created nor endorsed by Google.

Google Forms

H.2. Frankenbot's Evaluation via AttrakDiff


Deutsch | English

Assessment of **Frankenbot**

With the help of the word pairs please enter what you consider the most appropriate description for **Frankenbot**.

Please click one item in every line.

human *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	technical
isolating *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	connective
pleasant *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unpleasant
inventive *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	conventional
simple *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	complicated
professional *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unprofessional
ugly *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	attractive
practical *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	impractical
likeable *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	disagreeable
cumbersome *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	straightforward

* required field

Back
Continue

Figure .1.: AttrakDiff's Questionnaire Page 1 [7]

Assessment of **Frankenbot**

With the help of the word pairs please enter what you consider the most appropriate description for **Frankenbot**.


Please click one item in every line.

stylish *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	tacky
predictable *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	unpredictable
cheap *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	premium
alienating *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	integrating
brings me closer to people *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	separates me from people
unpresentable *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	presentable
rejecting *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	inviting
unimaginative *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	creative
good *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	bad

* required field

Back
Continue

Figure .2.: AttrakDiff's Questionnaire Page 2 [7]


[Deutsch](#) | [English](#)

Assessment of **Frankenbot**


With the help of the word pairs please enter what you consider the most appropriate description for **Frankenbot**.

Please click one item in every line.

confusing*	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	clearly structured
repelling*	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	appealing
bold*	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	cautious
innovative*	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	conservative
dull*	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	captivating
undemanding*	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	challenging
motivating*	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	discouraging
novel*	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	ordinary
unruly*	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	manageable

* required field

Figure .3.: AttrakDiff's Questionnaire Page 3 [7]


[Deutsch](#) | [English](#)

Assessment of **Frankenbot**

In the following section we would ask you to give information about yourself and your own experience with the product.

Age

Gender

Education completion

Profession

How long have you been using **Frankenbot**?

Product experience*

* required field

Figure .4.: AttrakDiff's Questionnaire Page 4 [7]