




University of
Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Computer Science/ Data Science Applied Data Science/ Data Science	Spring semester, 2022.. Open Open / Restricted access
Writer: Narmin Orujova, Ekaterina Khlyboba	 (Writer's signature)
Faculty supervisor: External supervisor(s):	Krisztian Balog
Thesis title:	Personalizing Human-Robot Dialogue Interactions using Face and Name Recognition
Credits (ECTS):	30
Key words: Personalisation, Face Recognition, Name Recognition, Social Robot, Dialogue Systems, Furhat.	Pages:67..... + enclosure: 4
	15/06/2022 Stavanger, Date/year



Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Personalizing Human-Robot Dialogue Interactions using Face and Name Recognition

Master's Thesis in Computer Science
by

Narmin Orujova and
Ekaterina Khlybova

Internal Supervisors

Krisztian Balog

Ivica Kostic

June 15, 2022

Abstract

Task-oriented dialogue systems are computer systems that aim to provide an interaction indistinguishable from ordinary human conversation with the goal of completing user-defined tasks. They are achieving this by analyzing the intents of users and choosing respective responses. Recent studies show that by personalizing the conversations with these systems one can positively affect their perception and long-term acceptance.

Personalised social robots have been widely applied in different fields to provide assistance. In this thesis we are working on development of a scientific conference assistant. The goal of this assistant is to provide the conference participants with conference information and inform about the activities for their spare time during conference. Moreover, to increase the engagement with the robot our team has worked on personalizing the human-robot interaction by means of face and name recognition.

To achieve this personalisation, first the name recognition ability of available physical robot was improved, next by the consent of the participants their pictures were taken and used for memorization of returning users. As acquiring the consent for personal data storage is not an optimal solution, an alternative method for participants recognition using QR Codes on their badges was developed and compared to pre-trained model in terms of speed. Lastly, the personal details of each participant, as university, country of origin, was acquired prior to conference or during the conversation and used in dialogues.

The developed robot, called DAGFINN was displayed at two conferences happened this year in Stavanger, where the first time installment did not involve personalization feature. Hence, we conclude this thesis by discussing the influence of personalisation on dialogues with the robot and participants satisfaction with developed social robot.

Acknowledgements

We would like to thank our supervisor Krisztian Balog for all the patience and support while working with our team. We would like also to thank him for giving us an opportunity to participate at the conferences mentioned in this thesis. It was an amazing experience!

We would like to thank our second supervisor Ivica Kostric for his assistance to our work. Thank you for sharing your expertise and great ideas!

I would like also to thank my husband, Erling Herstad, for his endless support during my studies and this thesis work! (Ekaterina)

I would like to thank my mother, Aybaniz Orujova, for giving me motivation any time we faced challenges! (Narmin)

Contents

Abstract	iii
Acknowledgements	iv
1 Introduction	1
1.1 Approach and Research Questions	3
1.2 Contributions	4
1.3 Outline	5
2 Related Work	7
2.1 Conversational AI	7
2.1.1 Fundamental concepts	8
2.1.2 Types of Dialogue Systems	10
2.1.3 Conversational Information Access	11
2.1.4 Main Components of Task-oriented Dialogue Systems	11
2.1.5 Rasa	15
2.2 Social robots	18
2.2.1 Furhat	19
2.3 Related concepts	21
2.3.1 Personalisation	21
2.3.2 QR Codes	22
2.3.3 Face Recognition	24
2.3.4 Name Recognition	30
2.3.5 Automatic Speech Recognizers (ASR)	30
3 Approach	35
3.1 Personalisation	36
3.2 Face vs QR-code Recognition	37
3.2.1 QR-codes	37
3.2.2 Business Cards	38
3.2.3 Face Recognition Approach	39
3.3 Name Recognition	46
3.3.1 Names recognition challenges	46

3.3.2	Approach to improve name recognition	47
4	Experimental Evaluation	49
4.1	Datasets	49
4.1.1	Labeled Faces in the Wild dataset	49
4.1.2	Yale Face Database A	49
4.1.3	DAGFiNN dataset	50
4.1.4	NORA dataset	50
4.2	Experimental Setup	51
4.3	Experimental Results	54
4.3.1	Personalization	54
4.3.2	Face vs. QR-code recognition	55
4.3.3	Name Recognition	61
4.4	Discussions	63
5	Conclusions	65
5.1	Conclusion	65
5.2	Future Directions	66
	 Bibliography	 69

Chapter 1

Introduction

In the modern world we are so used to robots in every aspect of human life that it becomes impossible to imagine a day without them. Moreover, if before we were used to the image of robots building cars or steering ships, meaning mechanical parts that are executing tasks requiring physical labor, the robots of the future are humanoids. This means they can listen to you and understand your feelings. These robots are commonly referred to as “social robots.” The term “social” describes robots that can “interact and communicate with humans by following the behavioral norms expected by the people with whom the robot is intended to interact” [1].

Historically research showed that social skill is a fundamental requirement to ensure successful and positive human-robot interaction (HRI) [2, 3]. However, recent research indicates that having only social skills is not enough for robot’s positive acceptance. For positive HRI experience in long-term, conversations need to be personalized based on individual user characteristics [4]. Personalization is explained as the ability of robots to recognize the user, recall shared memories, and adapt conversation to the individual user’s need. Personalization can improve user engagement in conversation and ensure that users will not get bored of repetitive answers. Several studies have been conducted to investigate the effect of personalization, which show that personalized HRI has a positive effect on the user perception [5] and long-term acceptance of the robot [6]. Especially interesting are 2 field experiments [5] and [7], which compare personalized conversation with a baseline social interaction. The former experiment concludes that personalized robots are perceived more intelligent and likable, while the latter shows improved rapport and engagement with the robot. Moreover, this study [8] shows that the use of participant’s first name by the robot has a significant positive effect on participant’s perceptions of robot friendliness, mind, and personality.

These personalized social robots are commonly used as elderly homes assistants, rehabilitation coaches, education assistants, shopping or museum guides, airport assistants, and companion robots. In all the listed cases personalization is achieved by different means. For example, in (Irfan et al., 2020) an Adapted Pepper robot acting as barista at coffee bar at university campus is personalizing conversations by recalling last or most frequent orders [9]. In (Baecchi et al, 2019) [10] a system is proposed to personalize the experience of museum visitors by profiling them and suggesting customized artwork recommendations. Another example of Pepper Robot usage is PHAROS, presented in (Costa et al, 2018) [11], an interactive robot scheduling personalized physical exercises for elderly.

One potential application area for personalized social robots could be academic conferences. An academic conference is an arena for face-to-face knowledge exchange, but usually due to the busy schedule conference participants miss the opportunity to meet people with similar interests. It may also happen that even if they meet and have interesting discussions, their relationship is not preserved after the conference. Lastly, to get information about the conference schedule or speakers one may need to check special resources or look around for busy volunteers. All the mentioned can alternatively be addressed by utilization of a digital conversational assistant, capable of informing about conference events and giving personalized recommendations. Later, the contact details of participants can be made easily sharable, to maintain the network even after the end of the conference.

There are some examples of attempts to design such an assistant in literature. In (Sumi et al, 2001) [12] a mobile and Web application was created for Japanese Society of Artificial Intelligence (JSAI) 2001 Conference, which was eliciting participant's interests based on submitted paper and suggesting people to meet and discuss during the actual conference. Moreover, there have been different applications developed to improve the experience of conference participants, but an alternative approach can be the utilization of a personalized social robot, which may result in a more natural experience for participants.

The available personalization techniques store historical data about the users by interacting with them in the long term. In the case of conference assistants, however, the amount of information available a priori is limited. Conferences usually take place just for a couple of days and lack user's personal details in the conversations. In addition, users approach the robot one or twice throughout the whole duration of conference, hence, limiting the amount of training data required for their successful recognition later.

In this thesis we aim to develop a digital conference assistant named Dagfinn for the 2 conferences, happening this year in Stavanger, Norway. They are European conference on Information Retrieval (ECIR'22) and Norwegian Artificial Intelligence Research

Consortium (NORA'22). The assistant is going to be available as a physical robot (Furhat robot) at the conference venue and used as an assistant for the conference participants. The development of the robot takes place in Team of Teams, each responsible for specific functionality. The goal of our team is to personalize the participants' experience with an assistant. We address the issues related to the development of conference specific assistant by using a dual approach for user recognition, through face landmarks and with QR code. Additional personalization is achieved by recognition of participants' names and construction of personalized dialogues based on collected information, such as name, place they came from, hometown etc. These approaches may ensure successful user recognition in case of model failure and increase satisfaction of participants with assistant. Initially we were planning to estimate, based on facial expression, how satisfied participants are with interactions with robots. However, during the project discussions we realized that this task is not feasible in the context of the scientific conference due to the limited spectra of expressed emotions by the participants.

1.1 Approach and Research Questions

In this study, a social robot Dagfinn was designed and improved in three separate ways. First, we tried to improve its name recognition ability by searching and comparing several methods to enhance the performance of its black box speech-to-text component. Next, as optimal method was defined, we focused on memorization and recognition of returning users by means of two parallel methods, trained model, and QR-code reading. The last part included the memorization of important user's information to structure personalized dialogues for higher user satisfaction. For this we created a dialogue example and tagged each user's response with machine-understandable translation. Later we designed a dialogue-flow to define the actions Dagfinn can take in course of the conversation. We have designed these components from scratch by defining a list of user's intents and corresponding robot's actions.

The problem of human-robot interaction personalization using name and face recognition can be further reduced to the following research questions (RQ):

RQ1a: How can the name recognition using a black box speech-to-text component be improved?

In this thesis a Furhat robot with 2 different speech recognizers was used: Google Cloud and Microsoft Azure. Being high performance recognizers, both were limited in ability to correctly recognize the uttered names. As the complete robot system was considered

a black box to address this research question, we searched for and listed the possible improvement approaches.

RQ1b: How well does out-of-the-box recognition work for recognizing participants' names?

After identifying the potential methods for name recognition improvement, we compare the performance of improved speech recognizers on sample of international names.

RQ2: How do participants recognition using a pre-trained neural model with facial encodings compare to the QR-code-based recognition?

To address this question, we compare 2 different approaches in users' identification, namely face recognition by eliciting face encodings using pre-trained model and reading QR code on participants badges, which include printed QR code with the link to the business card. We identify the advantages and disadvantages of both methods.

RQ3: How does personalization in conversation influence the participants' satisfaction level?

To personalize conversations after the participant's identification his/her name is uttered by the robot several times during conversation. In addition, relevant user's personal information is collected and stored. Part of personal information is acquired through the registration page, rest collected during conversation. This information includes user's name, surname, origin, language, university name, user position (student or researcher), list of publication.

To analyze the performance of the designed system DAGFINN was used by several human-users. After each conversation users were requested to rate the conversation for its overall performance. This feedback was later analyzed to derive the level of satisfaction with personalized conversations.

1.2 Contributions

The main contributions of the thesis are:

1. A literature review on available methods to improve Furhat's Speech Recognizer in its ability to recognize names.
2. Comparison of different methods for improving Furhat's ability to recognize the names.

3. Definition of optimal QR-code size and distance to the robot for accurate QR-code recognition. Comparison of QR-recognition to the models for face recognition using Furhat's low-resolution camera.
4. A study of the influence of personalized conversations with Social Robots on the satisfaction level of users.
5. A new Furhat skill memorizing returning users and asking personalised questions.

1.3 Outline

This thesis is organized as follows: Chapter 2 gives an overview of conversational AI, dialogue systems, social robots and related concepts. The explanation of dialogue systems' type and their examples are provided. In addition, the architecture of dialogue systems is presented. The main components of the DAGFINN's dialogue system are presented in Chapter 3. In Chapter 4, the experimental setup is described. In addition, system evaluation and discussions are also presented here. The thesis is concluded with description of future work in Chapter 5.

Chapter 2

Related Work

In this chapter we are going through the main building blocks of the future developed personalised conference assistant. We start with the definition of conversational agents and fundamental concepts of conversation in Section 2.1. This section also introduces RASA a framework for assistant's development. In Section 2.2 we continue with definition of social robots and description of Furhat. Finally, Section 2.3 concludes by providing background information on personalisation, face and name recognition.

2.1 Conversational AI

The advancement of recent years in AI technologies have increased the capabilities of machines in a way that they are becoming able to mimic human-like interactions. The set of technologies fostering this evolution are referred to as **Conversational AI**. The term Conversational AI is used to denote systems, capable of **natural language understanding (NLU)** and responding accordingly mimicking human conversation. It is an interdisciplinary concept which combines the fields of artificial intelligence, machine learning, human-computer interaction, natural language processing, and information retrieval. Each of these disciplines has its own focus areas and is trying to solve a specific set of challenges for achieving the high performance of conversational agents. A high performing **conversational agent** or **dialogue system** is a computer system that aims to provide an interaction indistinguishable from ordinary human conversation [13]. To achieve this conversational agent should inherit key features of human dialogues.

The following subsections will list these features and briefly overview types and structure of dialogue systems. It will go through their main components, will introduce the RASA framework for development such systems, and will introduce the concept of Social Robots.

2.1.1 Fundamental concepts

Language is the dominant way for humans to express themselves, and conversation or dialogue has historically been the main arena of language. The word “conversation” in its traditional comprehension is defined as “oral exchange of sentiments, observations, opinions, or ideas”¹. In dialogue systems, however, a more pragmatic definition is satisfactory, where conversation is described as a set of attributes intrinsic to human interaction. These attributes are listed as turns, speech acts (dialogue acts), grounding, sub dialogues and dialogue structures, initiative, inference and implicature [13]. In the following paragraph we will go through each of these concepts based on the example conversation given below in the Figure 2.1.

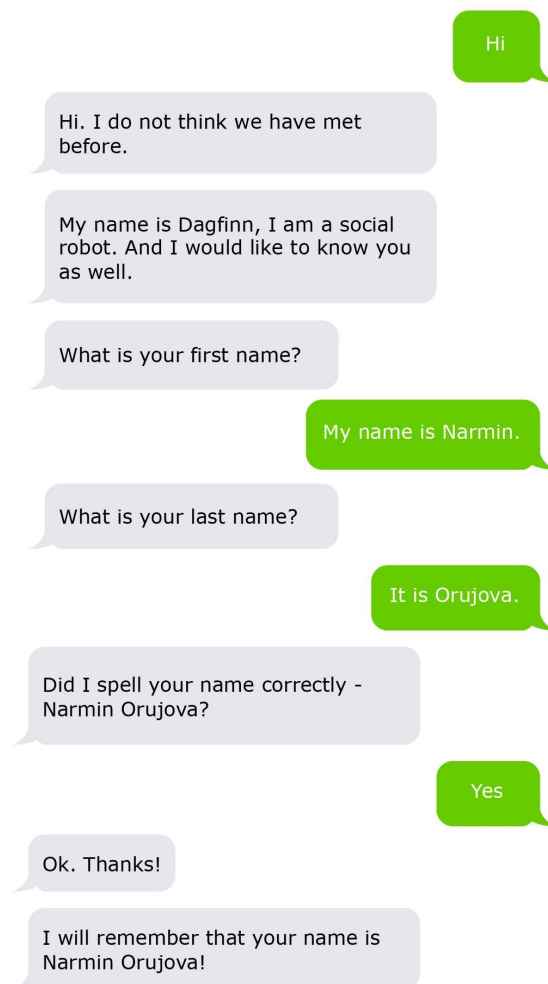


Figure 2.1: Example Conversation

Turn is the time for a speaker to contribute to the dialogue, which combined in a sequence forms the basis of a conversation [13]. The identification of turns is important in detection

¹<https://www.merriam-webster.com/dictionary/conversation>

of **endpoint**, a point where speaker finishes his/her expression and conversational agent can begin its own turn [13]. As seen from the examples, the conversation consists of turns, where the end of one turn gives green light for companion to start its own. Each turn itself represents an **speaking action** or **speech act**, performed by a speaker. These speech acts can be classified into 4 distinct types [13]. The first 2 types are **directives** (advising, requesting), which can be described as an attempt to get the opponent to act, an example utterance “What is your first name?”, and **constatives** (answering, confirming, denying) - statements with answers, example: “My name is Narmin.”. The remaining 2 types are **commissive** (promising, betting), and **acknowledgements** (apologizing, greeting) like “Ok, thanks!”. Another important attribute of human interaction is **grounding**, defined as a point of reciprocal agreement on common ground between dialogue actors [13]. We humans can acknowledge that a message has reached the addressee by responding with “Ok.” or repeating what was said last “I will remember that your name is Narmin Orujova.” These are examples of grounding in the given conversation.

In addition to the listed features, each human conversation has a structure. The example of such structure is the unwritten rule that each question should have an answer or proposal is followed by acceptance or rejection [13]. There are conversations, however, where these adjacent pairs are interrupted by subdialogues with the aim of correction or clarification [13]. An example of clarifying interruption in the given dialogue is the “Did I spell your name correctly?” utterance. Furthermore, there can be scenarios where actors are not explicitly expressing their intention but imply them through the chain of suggestive answers [13]. To infer information from this kind of dialogues one may check the relevance of the expressions used by assuming that speaker attempts to be relevant and does not utter random speech. This attribute is referred to as **implicatures** [13]. An example of the implicature is given in the dialogue below (Figure 2.2).

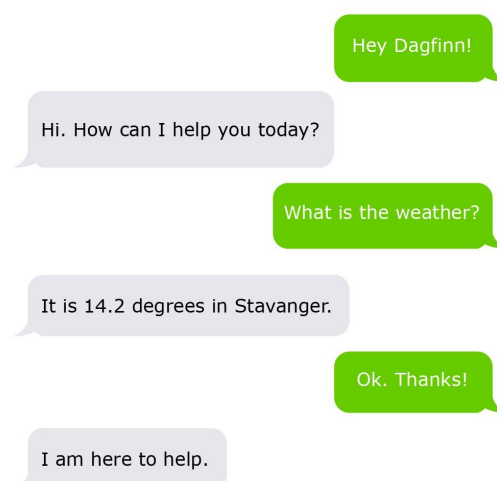


Figure 2.2: Example Conversation

The remaining attribute of human conversation is **initiative**, which is a role assigned to an actor driving the flow of conversation [13]. In some conversations one of the sides takes this initiative, in another this initiative is alternated between participants.

This section has listed the key features of human-human conversation and the ability to reproduce them in human-computer dialogues is one of the success measures for development of conversational agents.

2.1.2 Types of Dialogue Systems

Traditionally dialogue systems have been categorized into **task-oriented** or **goal-driven** and **chatbots** [14]. More recently a third type of agents were added to this categorization, namely **interactive question answering** [14]. Each of these systems has its own structure and their summary is displayed in the graph below (Figure 2.3).

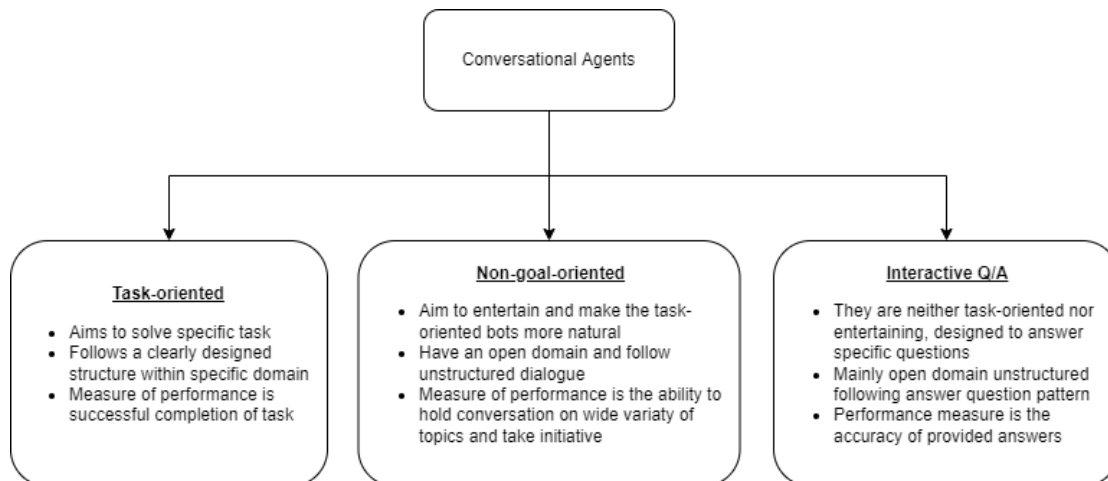


Figure 2.3: Types of Dialogue Systems

Task-oriented agents aim at completion of user-defined tasks [14]. Conversational dialogues in these scenarios have a specified domain and corresponding structure [14]. The performance measure for these agents is the successful and efficient completion of set tasks [14]. Typical examples of these agents are travel planners or appointment schedulers. Non-goal-oriented agents, on the other hand, hold an open domain dialogue and do not follow a structured flow with the aim of entertaining the user [14]. Their objective is the ability to hold conversations on a wide range of topics and take engaging initiative [14]. The third category of agents, interactive question answering bots, are neither task-oriented, nor entertaining, but designed to answer specific questions [14]. The dialogues with them are mostly unstructured open domain, following question answer pattern [14]. The performance measure for these bots is the accuracy of the answers provided [14].

In the next subsections we will briefly go through their descriptions and focus on the architecture of task-oriented by describing its components.

2.1.3 Conversational Information Access

Despite each category having its own goal, real-life agents tend to combine elements of these systems to meet the information needs of users. The subset of these Conversational AI systems that combine the features of all agent types to support different user goals, such as exploratory information search and recommendations are defined under the term of **Conversational Information Access (CIA)** [14]. These systems are also capable of conveying information through several modalities, by combining vocal responses with interactive screens. Moreover, they have the capability of taking initiatives and memorizing user preferences to give personalized answers. CIA systems typically follow a traditional task-oriented dialogue-system architecture. We will cover this architecture and its main components in the next sections.

2.1.4 Main Components of Task-oriented Dialogue Systems

Task-oriented bots use **frame-based** architecture, where a frame is a knowledge structure, containing the **intention** of users, and **slots** representing the key variables present in user input [14]. These dialogue systems' goal then becomes to fill the missing information in the frame and perform an action corresponding to user intention. These architectures are employed in commercial assistants, such as Apple's Siri, Amazon's Alexa, and Google Assistant [14]. To acquire missing information the system asks questions and retrieves the required slots from user's utterances. It keeps asking questions until all the slots are filled with user input. The architecture for modern dialogue systems represents a more sophisticated form of frame-based architecture and is called **dialogue-state** architecture [14]. Its main components and their interconnection is depicted below in the Figure 2.10.

As seen from the Figure 2.4 modern task-oriented dialogue systems are built as a combination of three separate components. First is **Natural Language Understanding (NLU)** which has the goal of identifying user intents and extracting key information. Next is **Dialogue Manager (DM)** used to interpret the structured user input and determine the possible system response. Last is **Natural Language Generation (NLG)** for converting the chosen actions into understandable human language. Here DM itself consists of **Dialogue State Tracker (DST)**, to keep track of states for current dialogue and **Dialogue Police (DP)** to decide the next action to perform in the dialogue. The DP refers to state logged in DST to choose the action either as a

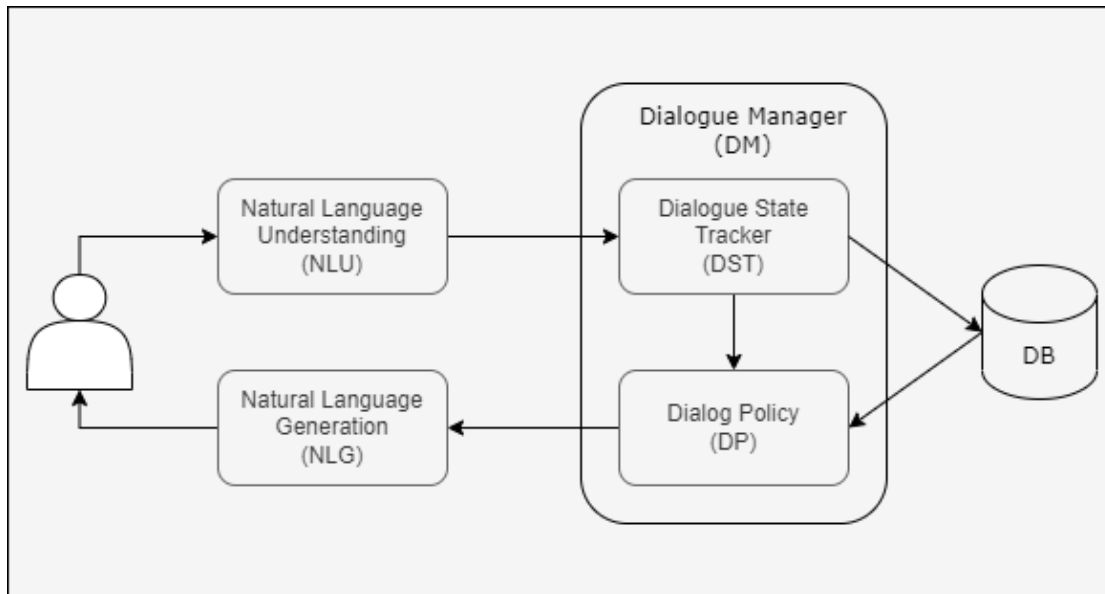


Figure 2.4: Main Components of Dialogue Systems

response to user or some operation on database (DB) [14]. We will go through each of these components in details in the next subsections.

2.1.4.1 Natural Language Understanding (NLU)

The task of NLU component is to map the given utterance into correct semantic slots [15]. These slots are always predefined for different scenarios. The table below shows an example of such mapping. To map a sentence into slots we refer to a commonly used tagging format called **BIO tagging**. BIO tagging, which is short for beginning, inside, outside, is a format for tagging tokens in chunking task for named-entity recognition [16]. The B-prefix here indicates the beginning of chunk and I-prefix shows that tag is inside the chunk. Lastly, an O-tag indicates that token does not belong to any entity [16].

Sentence	My	name	is	Narmin	Orujova.
Slots	O	O	O	B-first name	B-last name
Intent				Name	Surname
Domain				Name Recognition	

Table 2.1: An example of natural language representation

As it seen from the table as well, an utterance can have two representations: utterance-level and word-level. Formal representation informs about user intent and utterance domain, whereas latter extracts entities and fills the slots.

Intent and domain detection is the classification of user sentence as one of the predefined intents/domains using deep learning techniques [15]. One example of successful application of this technique is extraction using Convolutional Neural Network (CNN) a query feature vector for its further classification. To understand better, we can refer to examples. In the following example, all 5 expressions have the same intent - asking for a food recommendation. These intents include 2 entities: the first one is a food type (pizza, burger, coffee, ice-cream, bakery) and second entity is location (Stavanger, Sola, Randaberg, Sandnes, Tananger)

- Where can I buy pizza in Stavanger?
- Is there any burger restaurant in Sola?
- I would like to drink coffee in Randaberg. Can you recommend a coffee shop there?
- Is there any good ice-cream place in Sandnes?
- Can you recommend a good bakery in Tananger?

Slots filling is, on the other hand, a challenge for NLU as it includes a sequence labeling problem [15]. In this scenario the input to NLU is a sentence and the output is a semantic label assigned to each word in that sentence. This problem can be solved by means of deep belief networks (DBNs) or recurrent neural networks (RNNs) [15].

Once the semantic representation of an utterance is generated, it is further fed to the Dialogue Manager.

2.1.4.2 Dialog Management (DM)

The dialogue management process typically consists of two stages – state tracking and policy learning [15].

Dialogue State Tracking

To ensure robustness in dialogues the states of each turn in the dialogue are tracked by DST. In other words, DST estimates the user goal at the end of each turn, summarizes all user constraints and stores this information as state of the frame [15]. To obtain a clearer picture we can refer to the example in the Table 2.2 below:

User:	Will it rain today? <i>ask weather(location=Stavanger)</i>
Dagfinn:	There is always a chance of rain in Stavanger.
User:	Thanks for informing. <i>ask weather(location=Stavanger); thank()</i>
Dagfinn:	I am here to help.

Table 2.2: Example Conversation

After each turn the user intent, corresponding entities are stored and passed to the next turn. The dialogue act after the first turn conveys that out of all dialogue acts *ask weather* intent was chosen with location slot filled as Stavanger. This choice is based on the output of the model, trained with hand labeled dialogue acts. Traditionally, hand-crafted rules were adopted to determine the correct intent, but they were prone to errors. Hence, modern dialogue systems deploy ML and maintain the distribution of true state over multiple turns [15].

Policy Learning

The task of dialogue policy is based on state representation to decide on what action to take next, alternatively what dialogue act to generate [15]. If we assume that at the turn i of a conversation, we want to predict the next action based on current state of dialogue, expressed as $Frame_i$, then the formula for the next action can be expressed as:

$$\bar{A}_i = \operatorname{argmax}_{A_i \in A} P(A_i | Frame_{i-1}, A_{i-1}, U_{i-1}), \quad (2.1)$$

Traditionally, to warm-start a system the rule-based approach is used for selection, then supervised learning is joined [15]. Further the dialogue policy can also be trained with reinforcement learning, which can allow the system to make the policies based on learned conversations [15].

2.1.4.3 Natural Language Generation (NLG)

Once the dialogue act is determined, the respective text needs to be generated to respond to the user. Here NLG comes to the scene, which decides what and how to say. The response is generated in two stages: content planning and sentence realization [15]. We will go through both stages through an example in the Table 2.3.

get weather(location = Stavanger)

1. It is 14,5 degrees in Stavanger.
 2. The weather temperature in Stavanger is 14.5 degrees today.
-

Table 2.3: An Example of NLG Representation

If we assume that dialogue policy has chosen *get weather* as dialogue act with location being Stavanger, meaning the content has been planned, we can construct a corresponding response. To generate responses like 1 and 2, a sentence realizer needs to be trained on a large corpus of dialogues. As it is a challenge to obtain a large dataset of labeled dialogues, the trick called **delexicalization** can be utilized. It is a process of replacing slot values with placeholders representing them [?]. For example, we can replace Stavanger with *location name* in the responses above and hence increase the generality of our training data.

2.1.5 Rasa

To build a conversational agent capable of holding natural human-like conversations in this project we used the Rasa framework and the social robot Furhat. We will cover Furhat with its architecture in a separate section. In the upcoming subsections, however, we will introduce the Rasa Framework, which serves as the software architecture of our system.

2.1.5.1 Underlying Concepts

In this subsection, we will go through the underlying concepts of Rasa Framework and link them to the corresponding components of dialogue system's architecture.

As was mentioned before, Rasa is an open-source framework to create automated chat and voice-based conversational assistants [17]. Its architecture is illustrated in the Figure 2.5 below.

As we observe, the central element of this architecture is Agent, located inside rasa Server and connecting different components of Rasa SDK (Software Development Kit). Its main goal is to make sure that all the components communicate properly. For example, if the next action that Agent needs to handle is a custom action (for instance database query or API calls), it will communicate with Action Server, and get corresponding Response/Event in the response. Hence, the main interfaces the Agent has are with Rasa

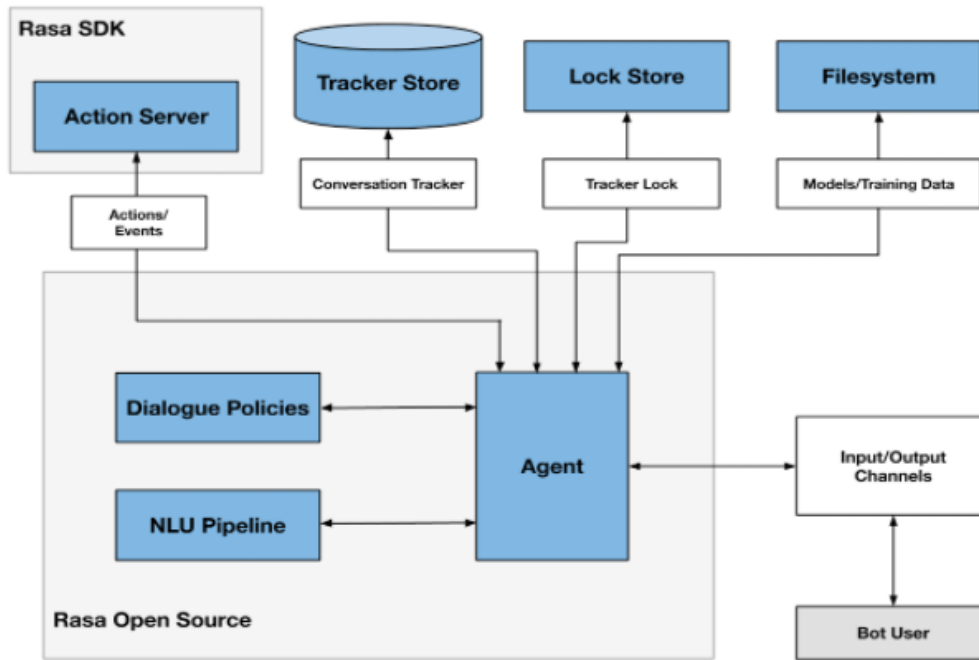


Figure 2.5: Rasa Architecture, Source: [17]

Action Server, Tracker and Lock stores, Input/Output channels, and with model/training data.

Apart from Agent, Rasa Server includes 2 other main components: NLU Pipeline, which interprets the user input and predicts intents/entities and Dialog Policies, that decides what response should be given to the user at the next step [17]. NLU Pipeline is implementation of NLU component of traditional agent architecture, whereas Dialogue Policies correspond to Dialogue Manager.

Furthermore, there are two stores needed to keep track of conversation: Lock Store and Tracker Store. Lock Store ensures, using the ticket lock mechanism, that incoming messages for a certain conversation ID are processed in the proper order and locks conversation while messages are being actively processed [17]. This allows several Rasa servers to run in parallel as replicated services. Tracker Store provides the opportunity to store all assistant's conversations within different store types, such as a variety of databases - PostgreSQL, Oracle, SQLite along with alternative options like Redis, MongoDB, DynamoDB [17]. In addition to available Rasa permits to implement a custom tracker store [17].

Next, the Agent has the capability to connect to different messaging channels to communicate with users on different platform [17]s. Rasa comes with support for a variety of messaging providers: such as Facebook messenger, WhatsApp, Telegram, Twilio, Slack

or a custom channel [17]. One can configure the respective channel in `credentials.yml` file [17].

To implement correct action Agent requires information stored in pre-trained model [17]. Hence, it needs access to the models and training data. To ease this process these files are stored under certain file structure with pre-defined naming. The model can be stored at any place after it is trained, however, the path to it should be specified. The trained models can be loaded in 3 separate ways: from local disk, one's own HTTP server or from cloud storage [17]. By default, they are stored on a local disk and only the latest one is loaded at the start of the server. The training data on the other hand should be stored in `nlu.yml`, `nlg.yml`, and `domain.yml` [17].

To understand how to build a Rasa model and what to store in these files, we first will give definitions of Intents, Entities, Stories, Rules, Forms, Slots, and Responses.

Intents and entities

The set of example sentences for each intent form a training data for intent classification and defined in the `nlu.yml` file [17].

Stories

A story represents a potential conversation flow between a user and an assistant, and have a well-defined structure that needs to be followed, such as each user input expressed as an intent is followed by an assistant's action, which can be an utterance or custom action [17]. If actions start with "utter" that means assistants will say some predefined text back to the user, such text replies are predefined in `domain.yml` or `nlg.yml` file, otherwise it can be some custom actions, for example querying database. In this case actions are defined in the `actions.py` file [17].

Rules

Rules are used to handle short conversations which should always follow a specific predefined path [17]. The conversation pattern will be followed once Rasa recognizes specific intent, which is included in the rule file. Examples of rules are to say back "Hi" anytime intent greet is detected or say "Bye" anytime intent goodbye is detected.

Forms and slots

Slots can be considered as the agent's memory, which is used to store key information provided by the user [17]. For example, departure country or the age of the user. They are also used to store information, which is gathered from database queries.

Forms are used to collect pieces of information from a user, or slots in other words, which will be further used in decision making [17]. For example, to provide travel COVID-19 information or make a flight ticket reservation, book a restaurant, call an API. Forms are defined in the *domain.yml* file and contain a list of several required slots, which must be filled by the user when the form is activated [17].

The form can be activated in the stories or rules files, by calling the form name under action.

Responses and Events

Responses are the replies that an assistant sends back to the user [17]. A response is usually a text message, which is defined in *domain.yml* file under the responses key.

Rasa Action Server can return apart from responses an event as the output of a custom action. These events are the way of action to influence the conversation [17]. An example can be an event of slot setting.

2.2 Social robots

At the height of the 4th industrial revolution among the new concepts that vigorously moved from fiction to reality are social robots. This poses a natural question of what social robots are and which goals do they serve. If we look back in history, the first mentions of robots appear already in the 15th century in the face of battlefield robotic knight designed by Leonardo da Vinci [18]. The usage of this concept in books and movies has been ever increasing since the 19th century. It is hard to imagine someone who has not dreamed about owning C-3PO or R2D2, the stars of Star Wars, or Ava from Ex Machina. The literature portrays robots with unrealistic capabilities of high ‘intelligence’ and with the invention of conversational AI systems, such as Alexa or Siri, we are moving in that direction. There are various branches of development within robotics, such as avatars and chatbots, who aim to be social, but lack the physical form, or manufacturing robots which are good at performing tasks, but lack the social component. Social robots, on the other hand, are physical machines with capabilities to interact with humans by mimicking social behaviors of listening, speaking, and expression of emotions. They are developed to be used in social contexts and are truly diverse in their forms and usage domains. Examples of such social robots are Sophia [19], a human-like robot of Hanson Robotics or Moxie [20], a robot built for children’s development enhancement.

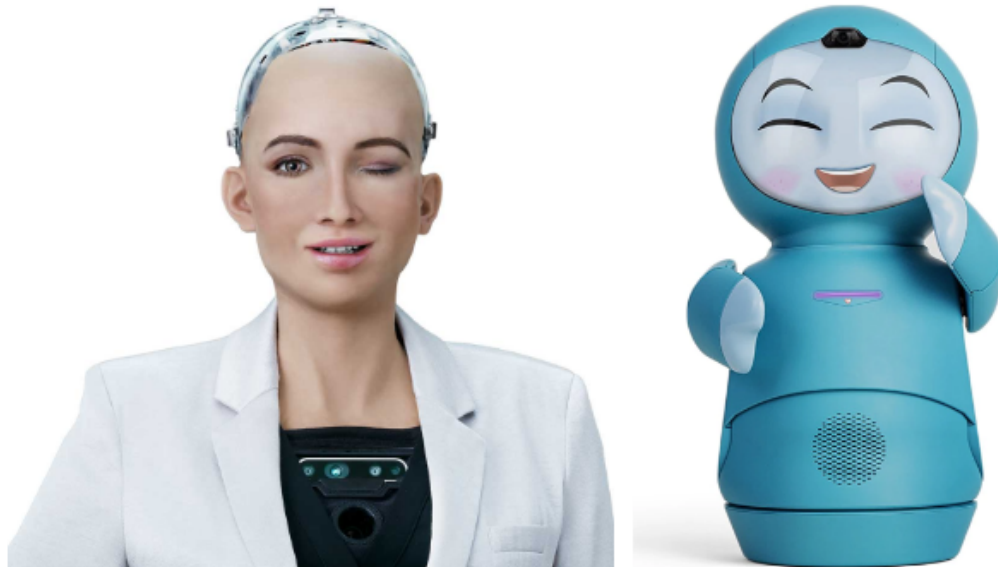


Figure 2.6: Left: Robot Sophia, Source: ², Right: Robot Moxie, Source: ³

2.2.1 Furhat

The social robot that we have used in this work, the Furhat, was developed at KTH, Stockholm, Sweden. It derives its name from the *"fur hat"* that it used to wear to hide the nakedness of its sculp [?] and is unique in its ability of face back-projection [21]. This makes it able to impersonate different characters and be capable of expressing a wide spectrum of emotions. Adding to this its ability to mimic nodding, raising eyebrows, and lip synchronization in 40+ languages [21], Furhat is mastering at simulating human-like conversations.

The robot is equipped with audio, visual sensors, and standardized I/O ports enabling its extensions and inclusion into other systems [21]. The operating system of the robot is called FurhatOS and it provides a runtime environment for enablement of Skills or in other words various applications of the robot. The FurhatOS hooks the developed skills to dialogue flow, providing it with function for natural language understanding, user attention model, and other key functions such as gesture and motion expressions. The operating system includes the set of modular subsystems, such as motion functions, facial animation, vision, audio processing, I/O, and cloud service integration, to run the robot. These subsystems are written in C++ and Java programming languages and intercommunicate by messaging bus.

To create a new skill for Furhat one should use Furhat SDK (Software Development Kit), which includes all development tools, APIs, as well as tutorial and documentation. The development can be done using Kotlin programming language and hooked to API

provided by FurhatOS. Moreover, virtual simulated Furhat environment can be used to develop, execute, and debug skills on development workstation before deployment to actual physical robot to perform final tests.

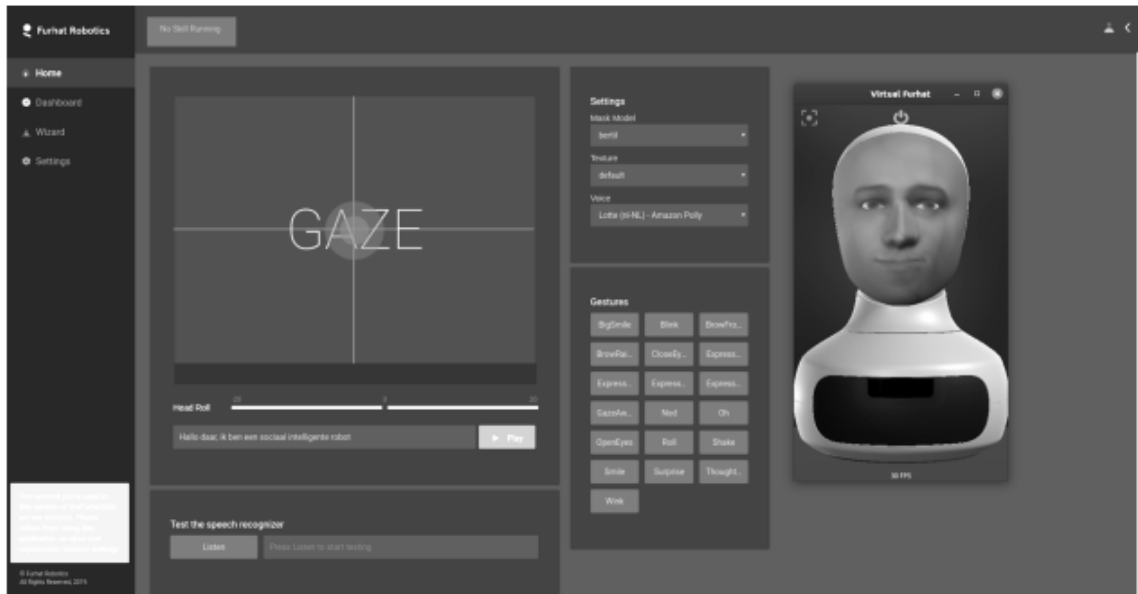


Figure 2.7: The virtual Furhat environment

The Furhat SDK uses flow, intents, and entities concepts as key abstractions. Flow is a state chart for building complex interaction and code re-usage [21]. A State is a building block in state diagram and flow is always in one state, from which it can transition to another. States also define triggers, containing the actions to perform [21]. This setup allows to classify each utterance as intent with entities, which are used to fill the frames and perform corresponding actions.

One of the main Furhat capabilities utilized in this work is its visual perception. By utilizing its onboard camera Furhat is capable of:

- 120°diagonal field of view
- Face Detection
- Face Recognition
- Facial Gesture Recognition
- User Tracking
- Body Position Detection

Apart from this, Furhat can detect the user's attention and can keep track of 5+ concurrent users [21].

Another component used is Speech recognizer, which takes input from 2 x 100Hz 10kHz digital omnidirectional microphones, set 180mm apart on the robot's shoulders. In addition, the automatic echo cancelling, noise suppressing omnidirectional USB microphone comes in bundle. It can pick-up far-field voice up to 5m in 360° [21].



Figure 2.8: Furhat: Inside out

All the mentioned above features make Furhat a powerful tool for exploration of the research questions defined in this work.

2.3 Related concepts

In this section we will give the background information for concepts used in this thesis, namely: Personalisation, Face Recognition, and Name Recognition.

2.3.1 Personalisation

As it was stated earlier there are several means to construct a personalised dialogues with users. This may include face recognition, name recognition, or alternatively small talks or humorist responses.

As several studies indicate, small talk helps to maintain user engagement and to build trust between the user and the robot [22]. In addition, the use of humor helps computers interact with humans, in the same way as humans interact with each other [23]. Another research suggests that humor plays a significant role in successful HRI interaction and

positively influences users' perceptions of anthropomorphism [24]. Anthropomorphism is derived from the Greek words "Anthropos," meaning human, and "morphe," meaning form. Humans tend to anthropomorphize objects with human-like features, emotions, cognition, or intention. Personification, or perceived anthropomorphism, was found to be correlated with user satisfaction with Alexa [25]. The interaction between the user and the agent in each of these contexts can benefit from using humor for purposes such as lightening the mood and increasing the engagement level while also enhancing the user's perceptions that the technology is human-like.

2.3.2 QR Codes

As in the 1960s Japan has entered to the period of high economic growth and supermarkets started to spring up in neighborhoods, the need to automatically key the price on goods has emerged. As a result, 1D barcodes were first introduced and became spread extremely fast. However, the fact that they could hold only up to 20 alphanumeric characters of information was a huge obstacle. To overcome this a Japanese Denso Corporation in 1994 has introduced an alternative 2D QR code, that allowed the information to be read across and up/down [26]. The design for the QR code was modeled on colored pieces of board game "Go" and the main challenge was to make the reading process as fast as possible [27]. The solution was found by adding positional information to the QR code, which allowed the orientation to be determined irrespective of the scanning angle, that could be any from 360°. Hence, the resulting codes with capability to be read more than 10 times faster were named QR, which stands for Quick Response. Although the patent on QR Codes belongs to Denso Corporation, there is no cost for its usage, and it is publicly available. Since 2000 it has been approved by the ISO as one of the standards [26].

Modern QR codes may seem very usual to our eyes, but they have distinctive design features which make them extremely popular. One of them is the squared shape, proposed by the creators. After an exhaustive survey they concluded that square was the pattern less likely to appear [26]. Apart from having specific shape, QR Codes consist of 7 key elements, each having a specific role. Each of these elements creates puzzle-like pixel pattern and holds information for direction, error tolerance, and timing. We will go through each of them briefly below.

First is positioning marking, corresponding to the first image in the top left corner. It indicates the QR Code print direction and has a unique ratio of black to white to be distinguishable from similar looking marks around. Next in the sequence is alignment marking to orient the image if it is too large. Then comes the timing pattern, that helps

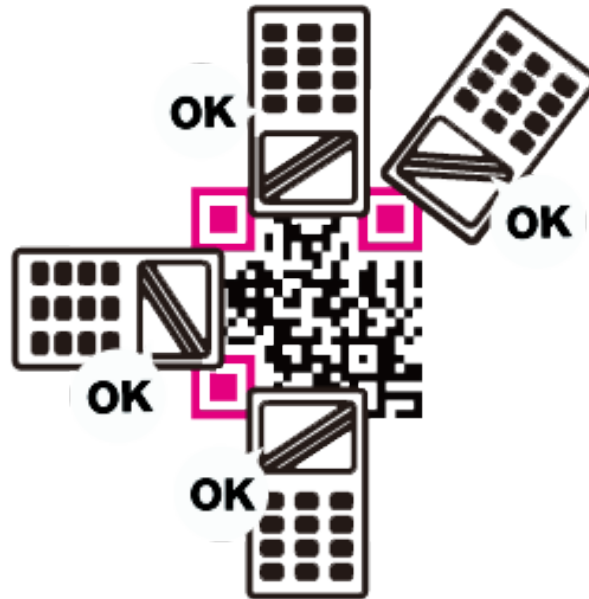


Figure 2.9: QR Codes, Source: [26]

the scanner to determine the size of the matrix in QR Code. The next crucial part is the quiet zone, which helps to segregate the QR Code from the surroundings. In total there are over 40 various versions of QR Code. The markers in the low left corner show which version is used. The adjacent image shows the format information, conveying error tolerance information to make the scan easier. The last image displays the most important data and error correction module. It stores the actual data and surrounding blank space to allow the data to be readable with up to 30% of code damaged [28].



Figure 2.10: QR Code Components, Source: [28]

One of the reasons for robustness of QR Codes in the physical world is in their ability to function even if part of it is obscured or removed. The algorithm allowing this high

functionality is called the Reed-Solomon Error Correction algorithm [29]. Concisely, it expresses the original data as polynomial and adds to the image in the background while code is generated. There are 4 correction levels, each of which adds more backup data depending on the expected amount of damage in the intended environment. These levels vary from L starting with 7% damage tolerance through M – 15% and Q – 25% to H with up to 30% damage tolerance. A fundamental property of QR Codes is that more data is programmed by addition of row and columns. Hence, the higher the error correction level the more backup data is included and QR Code becomes denser. This creates a trade-off between the size of the QR Code and its error tolerance level [30].

With the spread of QR Codes globally, new types were introduced to meet the user's needs. One example is micro QR Codes, the smallest codes. As seen from the Picture 2.11, opposed to QR Codes micro codes have only 1 position detection pattern at top left corner and the 2 module-wide quiet zone margin instead of 4 module-wide. This allows to print Micro QR Codes in ridiculously small areas, but also limits the amount of data stored, max 35 numerals or 21 alphanumeric [31].



Figure 2.11: Left: Micro QR Code, Right: iQR Code, Source: [31]

2.3.3 Face Recognition

Face Recognition is the process of recognizing human face using technology. Human recognition ability is so complicated that it becomes a challenge to develop an automated system with similar performance. This process involves image processing, pattern recognition, and even psychology. In fact, the work on this technology originated in the field of psychology [32].

The history of face recognition traces back to the 1960s. The father of this technology is computer scientist and mathematician Woodrow Wilson Bledsoe who developed a system of measurements which allowed to classify the images with faces. Soon Law Enforcements Agencies became interested in the technology and took its development to the next level.

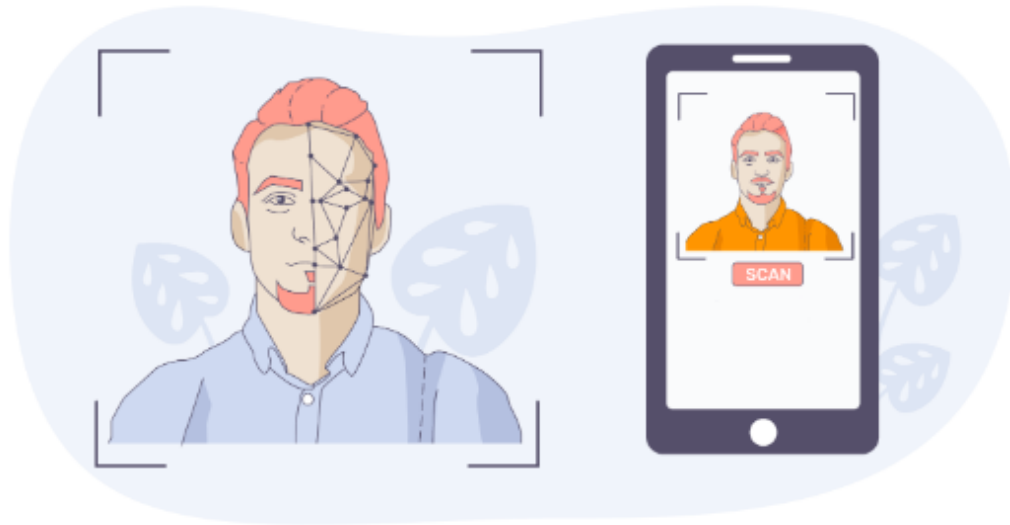


Figure 2.12: Face Recognition, Source: ⁴

Even though these technologies were used in practice, it was not until the year 2010 when computers became powerful enough to make these technologies a more standard feature [32]. Nowadays face recognition has become ubiquitous. We use it in our smartphones and other personal devices.

Throughout history different approaches were taken to improve the state-of-the-art technologies both performance wise and in time complexity. Some of these approaches suggested the usage of subjective features, others template matching. In the late 90s the dominant approach became the mapping of high-dimensional features to eigenvectors [32]. In this section we are going to discuss the different approaches available and will describe the method we have used in our work.

2.3.3.1 Face Recognition Workflow

The overall structure of Face Recognition System is given below in the Figure 2.13, and we will go through each component one by one. The input for this system is an image or video, the output is the identity of the person recognized in the given image or video. The input of this system passes through 3 steps on its way to the output. They are face detection, feature extraction, and face classification.

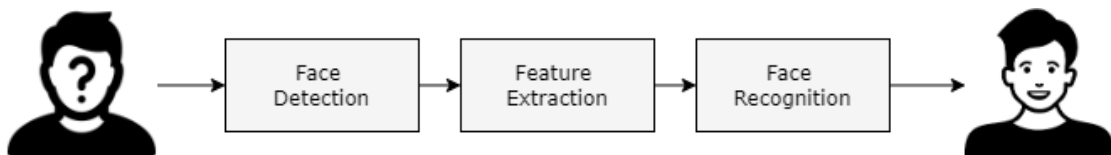


Figure 2.13: Face Recognition Workflow

Face detection involves the identification of a certain image region as a potential face and extraction of the face. Next, the facial features are obtained from the extracted face during the second step. Lastly, the obtained features are used to recognize a face by matching them with the database values during the face classification step. The variation of these 3 steps and their combination gives birth to different approaches for face recognition [32]. In the following paragraphs we will briefly go through each of these steps.

Face Detection

Contemporary technologies sometimes bypass the face detection step while recognizing a face. This happens due to the normalization of face images before inputting them to the recognition algorithm. An example of this can be the database with pictures of criminals, where each image contains only their faces [32]. However, usually the input image contains multiple items and people, and here Face Detection step comes in handy. These input images not only contain several objects, but also vary in several attributes, that create challenges for detecting a face [32]. These challenges are:

- **Variation in Pose:** Ideally one would like to take a picture of a person sitting right in front of and looking directly at the camera. This is not the case in most real-life scenarios. The images are subject to uncontrolled conditions such as moving person or wrong camera angle. The resulting “bad” images degrade the performance of recognition algorithms.
- **Occlusion of Features:** Apart from pose, the face can be hard to identify due to the presence of glasses, beards, or coverage of part of the face by another object.
- **Facial Gestures:** Another issue with the correct identification of facial features is facial expressions, which cause the noise.
- **Imaging Conditions:** Lastly, lightning, or other ambient conditions can affect the face appearance or lower the quality of image.

The face detection itself can also further be divided into 2 sub steps: face detection and location. These steps can either be combined or executed sequentially. Overall, the process is as follows, the input image is preprocessed and analyzed to get the facial features, which are later weighted and compared to labeled data to classify if there is a face or not. In other words, face detection can be viewed as a binary class classification problem, whereas face recognition is a multi-class classification problem. This brings us to conclusion that face detection is simplified case of face recognition problem, hence both use similar approaches [32]. Before listing these approaches, we need to mention that there are 3 distinct scenarios to identify a face [32]. They are:

- Controlled Environment, when the images are taken under controlled ambient conditions and faces can easily be identified.
- Identification by skin color. Studies show that even people have different skin colors, it is the intensity of colors that causes this variation. Therefore, chrominance can be used to identify a face [32].
- In some scenarios the movement of human bodies or some part of the body, such as eyes, can be used as a feature to detect a face.

Regardless of these scenarios we can list 4 different methods to detect a face in the image, such as:

- Knowledge-based method, which uses the general knowledge about human faces to create a set of rules, based on which a face is detected [32]. An example of such a rule is that each person has 2 eyes, or that cheeks area is lighter than eye areas. This method has a disadvantage however, as it is difficult to choose the optimal set of rules to refer to.
- Feature-invariant method on the other hand uses the features of human faces that are invariant to face position and angle. Even though this method can achieve a high success rate, it can work with only simple images, misdetecting in the presence of face occlusions [32].
- Template matching method tries to match a face using a template, which is the standard for all the phases, such as the elements of face, the relations between them, or even the color of skin. This method also has disadvantages when it comes to occlusions or variations in the pose [32].
- Appearance-based method also uses templates to identify faces, but these templates are derived from available training data. For this it uses machine learning or statistical methods. Some techniques use the probability of feature vector belonging to certain class, others use discriminant functions to define the boundaries between these 2 classes of face and non-face pictures. The algorithms used in this method include Neural Networks, Support Vector Machines, Naïve Bayes Classifiers, or Hidden Markov Model [32].

Feature Extraction

We human beings can identify faces from our childhood and are doing that without difficulty even if a person wears glasses or has grown a beard. This task that seems

trivial to us is hard to replicate in computers. To do so we need to extract the correct set of features from the images in comparatively less time and space. The choice of these features highly affects the performance of the classifier. One of the reasons for this is the well-known problem of “curse of the dimensionality,” according to which the high number of features can result in overfitted model, low amount in underfitted model [32]. To overcome this one need to choose training at least 10 times available features, and sometimes we are limited in data. The solution here comes with feature extraction and feature selection. The definition for the former is the usage of techniques such as Principal Component Analysis (PCA) to map the original feature space to lower dimension, whereas the latter technique chooses the subset of available features for further analysis. The feature extraction step here utilizes either one of these techniques or combines them to get the optimal set of features from input images to further optimize the performance of classification [32].

Face Classification

Once the optimal set of features is selected, it needs to be fed to the classifier, which at the end will map these features to the record of the person’s identity, stored in the database. There are several types of classifiers, and their choice is crucial for successful face recognition. Despite this variation in its core a classifier tries to match its input to the set of known data and what makes them vary is the metrics they use for matching. Based on this we derive the following key concepts:

- Similarity, which is an approach of matching based on similarity metric. On example of such metric is the Euclidean distance. The object with similar features is assumed to belong to the same class and each new instance is classified by its similarity to the meaning of the respective class. An example of such a classifier is k-NN (k Nearest Neighbours) [32].
- Probability. The methods using probabilistic approach define the correspondence to the class by calculating the probability of belonging to that class by using Bayes Decision Rule and Bayes error. An example of Bayesian approach is Maximum A Posteriori (MAP) rule, defined:

$$p(Z|w_i)P(w_i) = \max\{p(Z|w_j)P(w_j)\}, Z \in w_i \quad (2.2)$$

where w is the class name and Z is an input image. Here posterior probability of belonging to the class is calculated by multiplication of prior probability with class

specific density, defined as:

$$p(Z|w_i) = \frac{\exp -\frac{1}{2}(Z - M_i)^t \Sigma_i^{-1}(Z - M_i)}{(2\pi)^{\frac{m}{2}} |\Sigma_i|^{-\frac{1}{2}}}, \quad (2.3)$$

where m and Σ are the class mean and covariance matrices, respectively. Examples of this type of classifier are Logistic Regression or Parzen Classifier [32].

- **Decision Boundary.** The approaches using Decision Boundary divide the feature space into regions by defining the boundaries using parametric and non-parametric methods. These boundaries can be linear and non-linear depending and the choice of algorithm varies based on that. Examples of classifiers using decision boundaries are LDA (Fisher's Linear Discriminant), Multiplayer Perceptron, SVM or Decision Trees. In the case of higher dimensional feature space kernel functions can be used [32].

Each classifier is good for a specific set of tasks and depending on the problem definition the appropriate one or combination of several can be used for face detection. Among the reasons for combining the classifiers can be the datasets collected at different condition and containing distinctive features. Alternatively, it can be the case when the output of combination of classifiers is more robust and accurate compared to the output of single one.

2.3.3.2 Face Recognition Approaches

All the mentioned 3 steps can be unified in one algorithm and the combination of steps used in it can vary. In this section we will go through the main types of algorithms used for Face Recognition.

Geometric or Template based

Face Recognition algorithms can be either template or geometry based. Template based algorithms use templates, constructed from statistical tools, such as SVM, PCA, LDA, or Kernel Functions, to match the set of features. Geometry-based algorithms, on the other hand, define facial features and their relations for matching. The former approach is more used in newly developed technologies. Some algorithms can have the combination of both [32].

Wholistic or Piecemeal

Faces can be recognized by extracting only the key components of the face or the wholistic view can be obtained by considering the relationships between features and face contour

as well. The former approach is called piecemeal and is used less nowadays. The latter on the other hand is wholistic approach [32].

Model or Appearance based

Another division of recognition algorithms is model, or appearance based. Appearance based models usually use raw images with facial properties represented by high-dimensional vectors. This vector is further matched to feature space using statistical tools and each sample image is compared to training data. Model based algorithms, however, model human faces and find the optimal parameters that can be used later to recognize the face.

Appearance based methods can be linear and non-linear depending on the feature space, with latter using more complicated methods for dimensionality reduction. Model based algorithms can also be divided into 2D and 3D. Model based algorithms can classify the face even if the pose of a person is different. These types of models are called morphable. With 3D models one can capture the face in 3 dimensions [32].

Template, Statistical or Neural Networks based

The last categorization is the same as in the case of face detection algorithms. Faces can be recognized by finding the minimum distance of images pixels, textures to the labeled data. Alternatively, the feature vector can be extracted and classified using discriminant function. Lastly, neural networks can be used for modeling the faces [32].

2.3.4 Name Recognition

This subsection briefly introduces Automatic Speech Recognizers and their accuracy metrics.

2.3.5 Automatic Speech Recognizers (ASR).

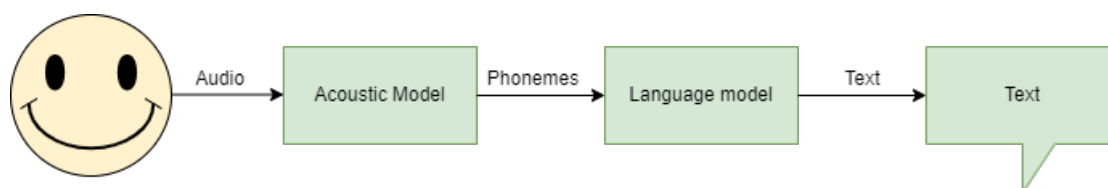


Figure 2.14: Traditional ASR System (simplified)

Automatic Speech Recognition (ASR) technology converts spoken words into text and is an important component powering many conversational AI solutions, enabling most natural human-machine interaction. Successful implementation of ASR systems in the

recent years in mobile applications and virtual speech assistants (e.g., Amazon's Alexa, Apple's Siri, Google Assistant and Microsoft's Cortana) led to increased interest in ASR from both industry and research community. Recent surge in the deep learning techniques powered by big data and increased computational ability resulted in continuous improvement of ASR systems. The modern generation of speech recognizers is based on deep learning which continuously learns to extract relevant features from audio data and uses them for text prediction. Traditional Automatic Speech Recognizers (ASR) System takes an audio file from a user into Acoustic model. The acoustic model looks at the audio wave forms or a spectrogram and converts these wave forms into sounds which it thinks are there, also known as phonemes. The language model then looks at the groupings and series of phonemes and tries to understand from those sounds which words or phrases are being said. Most language models also take context and other factors into account when determining the text which the phonemes represent. Some models produce just a single output whereas others have n best alternatives that could be determined from that grouping of phonemes. A Speech-to-Text API synchronous recognition request is the simplest method for performing recognition on speech audio data. Speech-to-Text can process up to 1 minute of speech audio data sent in a blocking synchronous request. After Speech-to-Text processes and recognizes all of the audio, it returns a response.

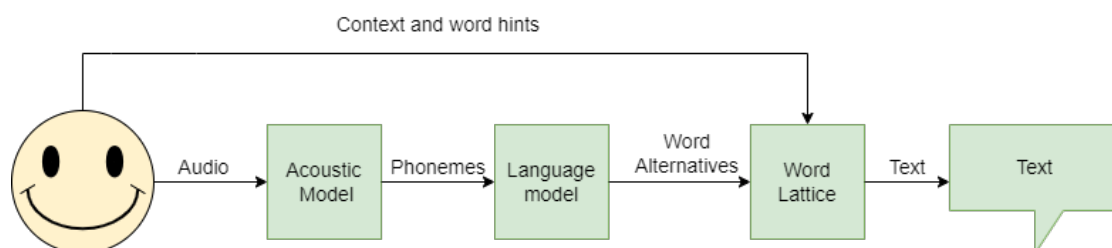


Figure 2.15: Google Speech-to-Text API (simplified)

Simplified version of Google Speech-Text API has 2 major differences: instead of producing just a single highest confidence hypothesis or a series of n best alternatives, it produces entire lattice of potential word alternatives. The other difference is the speech adaptation API, which allows user to send context and word hints about what the audio might contain. This information is then used to operate on the word lattice and determine based on these hints as well as the confidences from the language model, what the highest confidence hypothesis text is as well as n best alternatives if that's interesting. This differs from other types of model customization where you might have a custom language model where entire language model is changed to account for specific proper nouns or various sequences which might be expected.

2.3.5.1 Accuracy metrics for Speech Resognizers

Speech accuracy is vital when it comes to speech to text recognition since this text is further used in NLU pipeline to determine someone's intents and extract entities and the text is displayed directly to user. Having a higher quality transcript that's more accurate is going to result in better user experience. There are whole number of factors which can affect speech accuracy, from background noise to audio quality to various accents or strange verbalizations and all of this can have a big impact. Modern ASR try to make model as robust as possible out of the box on a wide variety of types of speech and general vocabulary. But all speech recognition systems are very sensitive to input data. That means, when developing application using ASR, you think about accuracy you really need to be thinking what the accuracy of this system on my specific data in my recorded environment is, the type of speakers I have, the type of speech it is. To do that, you must measure it, and to measure it we must have a common language for how we talk about what the quality of speech-to-text recognition

$$WER = \frac{(S + D + I)}{N}, \quad (2.4)$$

- S is the number of substitutions,
- D is the number of deletions,
- I is the number of insertions,
- C is the number of correct words,
- I is the number of insertions,
- N is the number of words in the reference ($N = S+D+C$)

2.3.5.2 Customization of ASR.

Despite being high accurate on general vocabulary, Speech-to-text performs poorly on rare vocabulary recognition and especially Named Entity Recognition (NER) tasks. There are 3 ways of improving speech accuracy through model customization.

1. Customize the model to your domain by providing contextual information An example if we know what people will talk about like booking restaurant with names of meals: sushi, pizza, burger etc.

2. Tweak weights to address specific word/phrase issues Commonly this can occur with proper nouns, people names, individual products names, but words which occur very rarely in all speech
3. Use context to bias towards specific types of information or specific situations for example, for AVR systems (Automate Voice Response Systems), where users will tell account number or phone number, we can tell the ASR system to only look for a digit sequence or a alphanumeric sequence. Both Azure Recognizer and Google Cloud Speech support all 3 types of these customizations using speech adaptation tools. This tools, if used correctly, can be extremely powerful and can shift quality very significantly.

Chapter 3

Approach

In this section we are going to present our approach to develop a personalised conference assistant. In section 2.1 we will overview the software architecture of our system. Next, we will describe the approach taken to personalise the dialogues with our assistant in Section 2.2. We will look separately into personalisation through participant’s recognition more closely in Section 2.3 and their name recognition in Section 2.4.

3.0.0.1 Software Architecture

Rasa is an open-source machine learning framework that we used to build our conversational assistant. It was used for intent/domain classification and custom action execution, whereas the interface with users is established through the Furhat robot and external screen, developed with Sanic Web Framework, and ran on our local PCs. The detailed architecture is depicted in the diagram below (Figure 3.1).

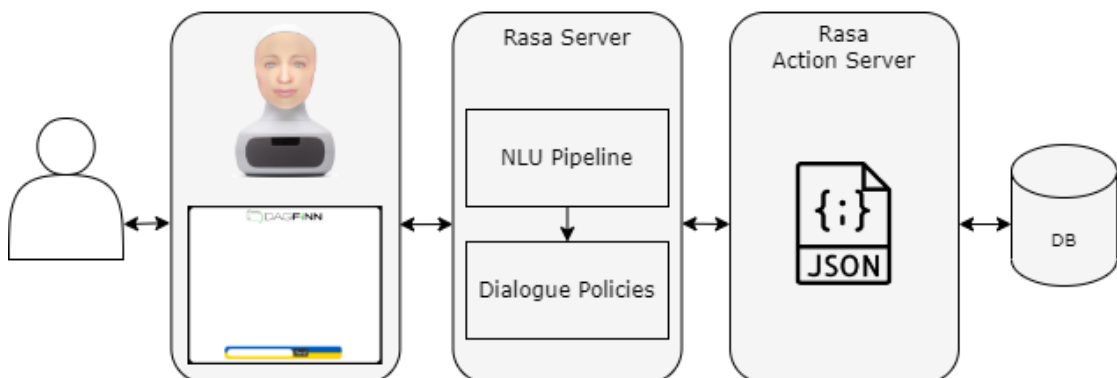


Figure 3.1: Software Architecture

As seen from the diagram the architecture consists of 2 servers: Rasa Server and Rasa Actions Server. Once the user enters the field of view the output of Furhat’s

Speech Recognizer/Camera is sent through custom Socket.IO¹ channel to Rasa Server. Rasa Server is used for 2 primary components of dialogue systems NLU and Dialogue Management. In the process whenever the assistant predicts a custom action, Rasa Server automatically sends a POST request to Rasa Action Server. This request is a json payload with the name of predicted action, the content of the tracker, ID of conversation, and domain content. Once the custom action is executed, if required, the action server connects to the respective database to populate or fetch the information. The connection to these databases is automatically set at the start of the Rasa server to save time. The output of the action server is also a json payload with corresponding events and responses, that are later returned to users and added to the tracker. Since the user interface is implemented through Furhat robot and interactive screen, the response generation in this case is separated from dialogue manager and outsourced to the Furhat's NLG through the same custom channel.

In this thesis we have focus on Furhat robot itself and were receiving frames through Socket.IO channel.

3.1 Personalisation

Based on available literature review and our own experience from ECIR-2022 conference, where Dagfinn was been introduced for the first time as a conference assistant, people are more willing to interact with a robot that is able to establish a good small talk communication. Therefore, in this work knowledge of the Dagfinn has been enriched by 20 small talks questions, where 10 of them were personal and another 10 general. The scope of the question was collected during ECIR-2022 as a suggestion from the conference participants. Moreover, our observations showed that humorist responses tend to have higher success, hence we tried to include this flavor in the DAGFINN's utterances. As a result, we came up with the following personalized conversation design:

1. User identification using visual recognition. As a first step, to initialize a personalized dialogue, we tried to recognize returning user or remember the first-time user. In this work, we have tested 2 different approaches in users' identification, namely face recognition and QR code reading from participants' badges, where QR code included the link to the respective participant's business card.
2. After user has been identified, the users name utters by the robot several times during conversation.

¹Socket.IO is a python library, built on top of WebSocket protocol, for bidirectional, low-latency communication between server and client.

3. Collecting and storing relevant user's personal information, such as name, surname, origin, language, university name, user position (student or researcher), list of publication. Querying database to recall some of the relevant user characteristics during conversation.
4. Small talks. 20 small talks examples, which are printed and available for users. Participants are very engaging in dialogue when asking personal questions about DAGFINN, such as if he can sing/dance or if he can go out for a date with them.
5. Humor. DAGFINN can make a joke and asks if the joke was funny and if user wants one more. This is the most successful topic to ask from robot from our experience during conference and experimental set up.

3.2 Face vs QR-code Recognition

To recognize a participant we have used 2 separate methods, QR code reading and Face Recognition. In this section we will describe each approach separately.

3.2.1 QR-codes

When we talk about the size of QR Code we refer to the size of QR Code image, as the minimum printing size is determined by the size of the little black squares called modules, that make up the image [33]. As mentioned before, more data introduces more rows and columns and consequently smaller modules size. If this size falls below the resolution limit of scanning camera, it will not be able to read it. An additional factor influencing the readability of QR Code is the scanning distance. The further QR is from camera, the smaller the modules appear in the camera viewpoint [33]. At some point it becomes so small that the camera is not able to read it.

To sum up, the main rule for QR codes is that the lower the error correction level the less dense the QR Code is and hence smaller. On the other hand, the higher the error correction level more prone it is to damage. As a result, the aim in this project was to find an equilibrium point and experiment with the minimum width of the image with printed QR code for varying scanning distances, controlling the minimum module size to be seen when scanned by the camera.



Figure 3.2: Trade-off between QR Code Size and Scanning Distance, Source: [33]

3.2.2 Business Cards

As our task was to scan the QR codes printed on the badges of the participants for participant's recognition, to make use of these printed codes, we have encoded the link to each participant's own digital business card in it.



Figure 3.3: Participant's Badge

To create these cards, we have used python web framework *Flask*. The choice was made on this microframework as unlike Django or any other framework, Flask is *pythonic*. Its learning curve is not huge, which allows an easy start for beginner developers, as we were. It is modern, up-to-date, and its functionality can be extended to build complex applications. To design this card, we have written 3 separate files. The structure of the card was defined in HTML (Hypertext Markup Language) file, the design was given in CSS (Cascading Style Sheets) file, and the app was created using flask inside Python file. The information of participants was fetched from the populated SQLite database available beforehand using *sqlite3* library.

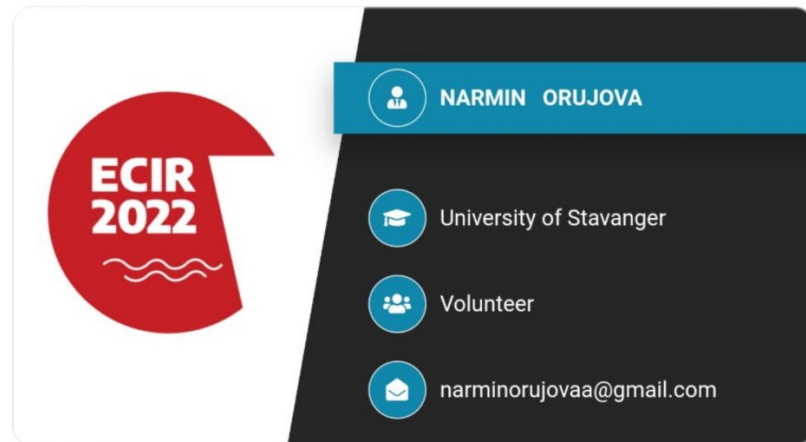


Figure 3.4: Business Card

3.2.3 Face Recognition Approach

Face Recognition is expensive in its nature, some services cost 86.40 per day, others 30000 per camera per year [34]. Hence, it is widely used mainly by the government or the big companies. Recently, however, more free frameworks and libraries have started to emerge, allowing us students to practice them. The open-source library that was used in this work is called *face recognition* and available on GitHub [34]. In this section we will go through the face recognition pipeline built in this project and will look closely at each component of the pipeline.

Face Recognition

To recognize a face, one needs to run a series of subsequent actions, by building a pipeline and passing the output of each pipeline to subsequent steps. These mentioned steps are:

1. Find all the faces in the image.
2. Analyze the face in the image and consider that even if there was bad lightning or different angle, it is still the same face in the image.
3. Get all the features of the face that can help create a facial signature for a person.
4. Compare the sample image to the database of known to infer the identity of person.

Each of these steps involve different machine learning techniques, that will be described in subsequent paragraphs. *Face Detection*

The first step in this pipeline uses two different ML methods.

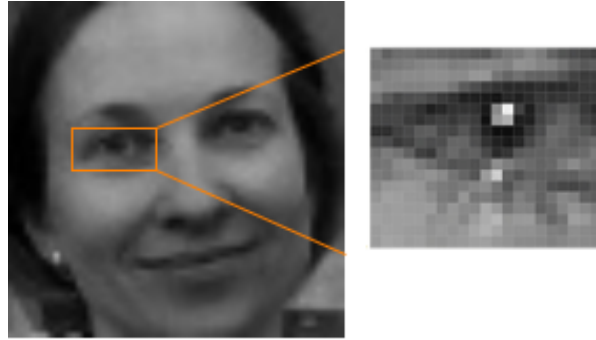


Figure 3.5: Example Feature

The goal here is to understand how darker the considered pixel is compared to its neighbors and mark the direction of change to darker region with an arrow. For example, in the image below (Figure 3.6), the pixel in upper right corner is darker, hence arrow also points in that direction.



Figure 3.6: Gradient of 1 pixel

By repeating this step for every pixel in the image we will get an image with arrows, replacing all the pixels. These arrows are also referred to as gradients, that show the change from lighter to darker regions throughout the image [35].

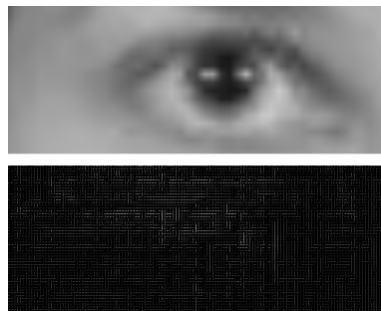


Figure 3.7: Gradient Feature

The reason for implementing this step is to simplify the process. As light and dark pixels have different magnitudes, using them in analysis may bring to wrong interpretation. We omit this consequence by considering only the change in light intensity or direction of brightness, we normalize the picture. This eases the problem the algorithm tries to solve. The next step is to replace the arrow of each individual pixel by the dominant direction in that region of the image. If we do not do so, then the resulting image will contain lots of arrows and the pattern we are looking for will be lost in the forest of trees.

Hence at this stage the image is divided into 16×16 pixel boxes and the arrow in each direction is counted. At the end, the direction with the most arrows is chosen dominant and assigned the respective box. In this way at the end, we get a simpler representation of the original image, conveying only information about face structure.



Figure 3.8: HOG Representation of Original Image

Lastly, to find the face in the resulting HOG image, one needs to compare it to the known HOG images created from the training data and find similar regions.

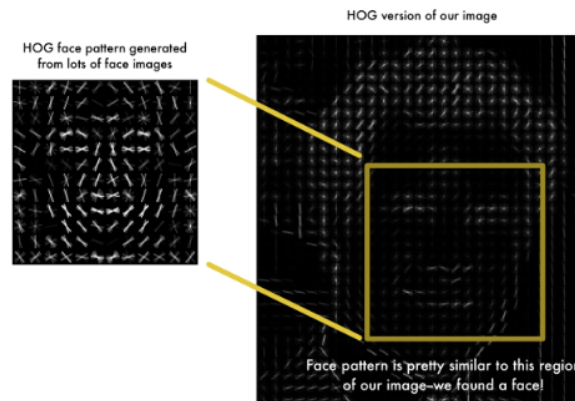


Figure 3.9: Mapping of HOG Representation to Training Data

As a result, a face is found on the picture. We have used this approach with Furhat's camera.

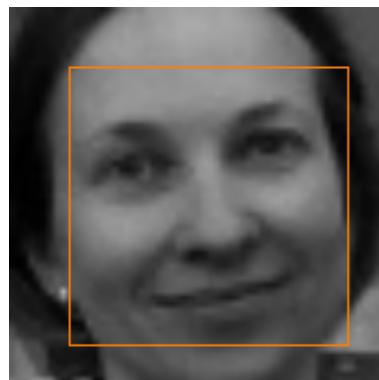


Figure 3.10: Identified Face

Another method used for face detection is Convolutional Neural Network (CNN). For this we have used pre-trained Max-Margin (MMOD) CNN model and compared its performance to the HOG algorithm both in terms of accuracy and complexity [36].

Feature Extraction

The next problem to address is that one can turn while looking at the camera, and then the computer will have difficulty understanding that it is still the same person. To make the computer's job easier we can make sure that some facial features, such as eyes, mouth are in a fixed position. For this we twist the original image.



Figure 3.11: Rotated Face

This is done through the algorithm called face landmark estimation. Its basic idea is to extract 68 landmarks or points that every face has, as shown in the picture below. Then train the model to find these landmarks in every face.



Figure 3.12: Face Landmarks, Source: [32]

After these landmarks are identified one can use affine transformations, the ones that preserve parallel lines, meaning scaling, rotation, and shear to transform the image to format where all main face components are at the right position, regardless the pose. In this way we simplify the task of classifier in face recognition.

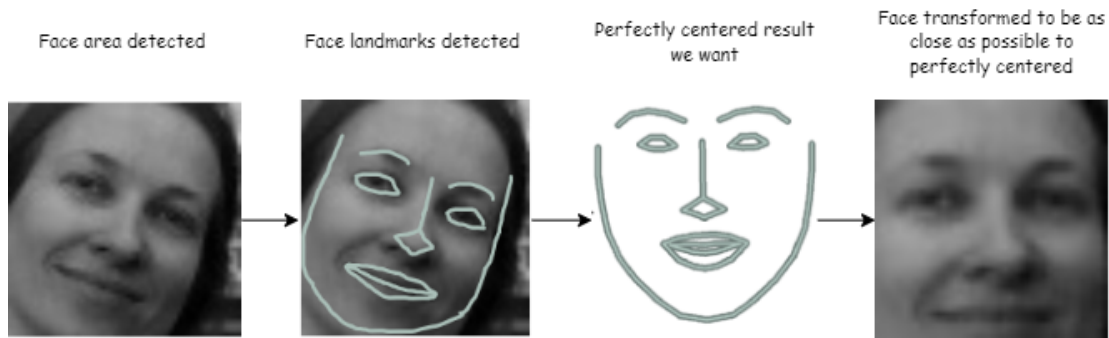


Figure 3.13: Face Transformation

Face Recognition

Now we have reached the most fruitful and interesting part of the pipeline, where the identity of a person is identified. This part is the trickiest as one needs to find the way to match the unknown face to the database of known faces, which can be of order of trillions in the matter of milliseconds. The solution to this can be to extract the specific measurements for a face such as ear size, the length of the nose etc. Then the same metrics can be calculated for unknown faces and the picture with smallest distance between measurements can be considered belonging to same person. This brings us to question which measurement we should calculate for optimal comparison. As was mentioned previously, computers do not look for the same features as humans when identifying a person. Hence, a workable solution for is to use Deep Neural Networks, particularly Convolutional Neural Networks to generate for each face the 128 measurements. The training procedure follows the following steps:

1. The model reads the image of the known person
2. Then it read another image of the same person
3. Then it read another image of the same person

After loading the pictures, the algorithm generates the measurements for each face and checks if the measurement created for images 1 and 2 are close to each other, whereas the measurements for images 2 and 3 are apart.

By repeating these steps millions of times neural networks learn how to correctly generate the 128 measurements, representing one's face and called face encodings. This process of reducing the initial image to 128 dimensions helps to boost the performance of the recognizer. However, to get the model with the ability to infer these encodings with accuracy one needs 24 hours of continuous training, which is very time consuming. On the other hand, once the network is trained it can create these encodings for any image,

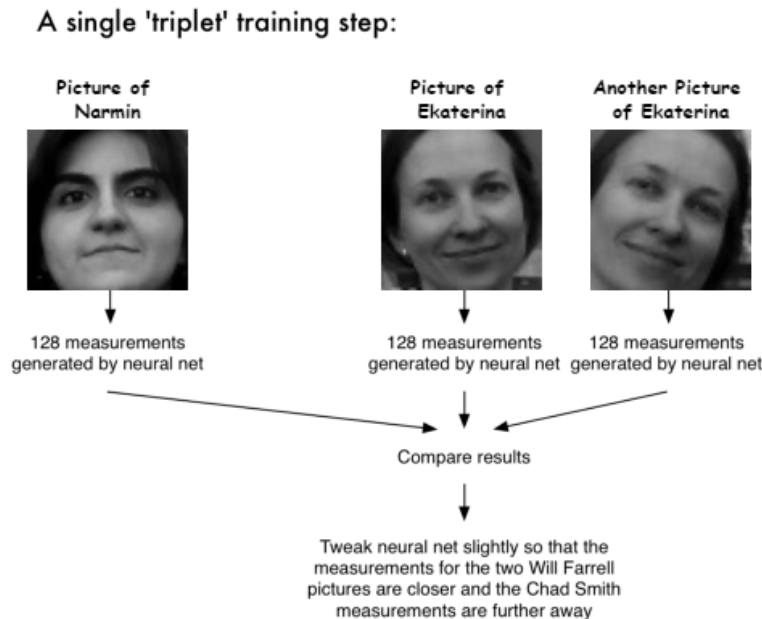


Figure 3.14: Model Training

even with an unknown face. Hence, we have used the pre-trained model for recognition, which luckily for us was available for free. So, what we did in this project, is that we ran the collected pictures through the pre-trained model and got a list with 128 numbers - encodings for each face, where these numbers were not interpretable. However, the same face had these numbers close to each other across multiple images.

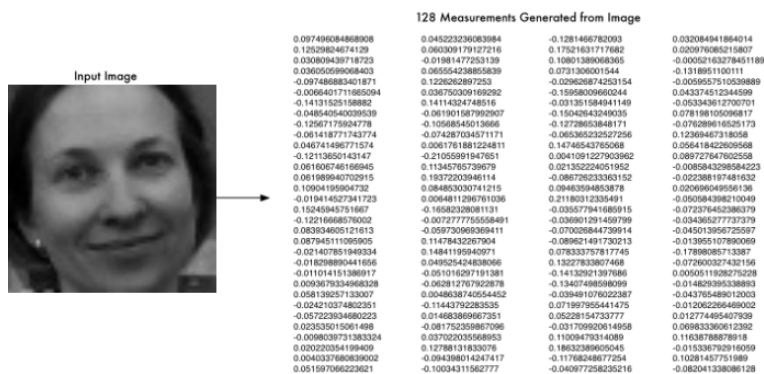


Figure 3.15: Face Encodings

The pre-trained model used in this project is a Residual Network (ResNet) with 29 convolutional layers. Its architecture with few adjustments was taken from the ResNet-34 network developed by researchers at Microsoft [37]. The classifiers used for recognition were k-NN and SVM. k-NN is supervised ML algorithm that calculates the Euclidean distance between sample point to and training data for classification. After calculating the distance, the algorithm chooses k nearest neighbors, where is the pre-defined value and estimates the probability of belonging to a particular class based on the number of representatives of that class in the neighborhood [38].

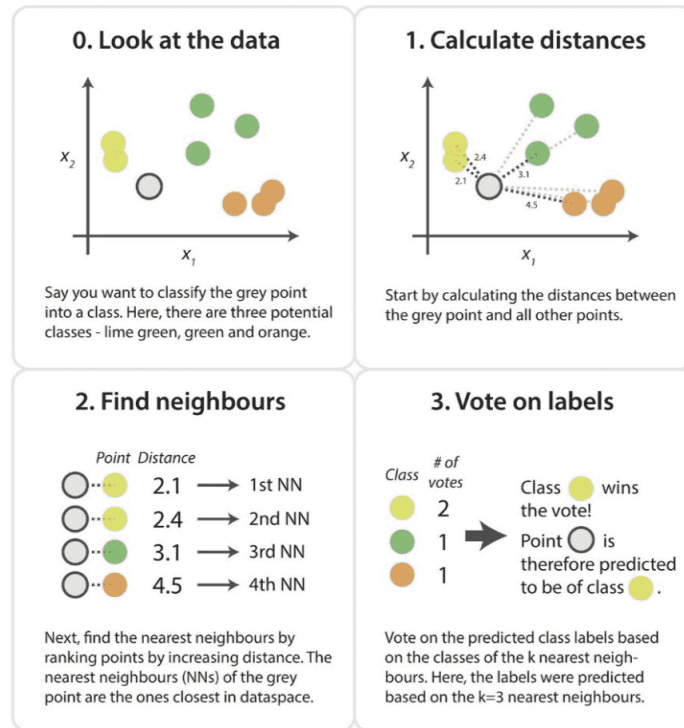


Figure 3.16: k-NN Algorithm, Source: [38]

Another classifier used is SVM, which stands for Support Vector Machines. This algorithm tries to classify points in N -dimensional space by finding an optimal hyperplane. Hyperplane is a decision boundary used to separate 2 classes. Data points on different sides of hyperplane belong to different classes. There can be many hyperplanes to separate classes, however, SVM tries to choose the one that maximizes the margin (distance) between plane and support vectors. Support vectors are the points closer to the hyperplane and influencing its position and orientation. [39]

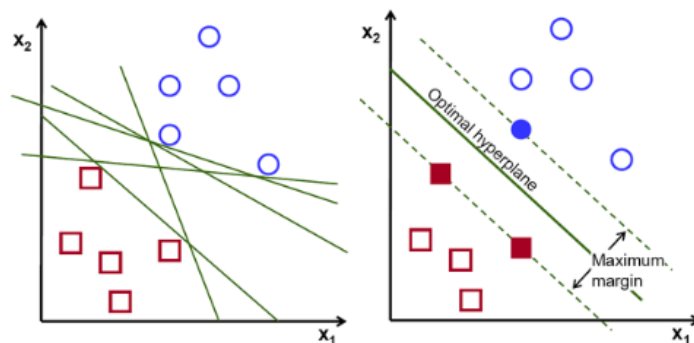


Figure 3.17: SVM Algorithm, Source: [39]

In this work we have compared the performance of both classifiers and choose the optimal one for face recognition in terms of accuracy and time complexity.

3.3 Name Recognition

3.3.0.1 Google and Azure Speech-to-Text API

There are several Speech-to-Text API recognition systems available on the market today. Currently there are two supported recognizers supported on Furhat Robot: Google Cloud Speech-to-Text and Microsoft Azure Speech-to-Text. In this thesis, we aim to provide empirical evidence on the performance of these two ASR providers. Here is a differences overview of Google Cloud and Microsoft Azure recognizers:

Feature	Google	Microsoft
Credentials provided by default	Yes	No
Logging	Yes	No
Available in China	No	Yes
Multiple language recognition	Yes	Yes, but not in real-time and maximum 2
Longest phrase	60 seconds	20 seconds
Multiple recognition hypotheses	Yes	Yes
Pricing	\$0.006/15 seconds*	\$1 per audio hour (\$0.0028/second)
Languages	~ 120	~ 37

* When listening, Google charges for a minimum of 15 seconds of audio. This is per listening call. I.e. if your robot listens for 7 seconds, then for 10 seconds, and then for 17 seconds, you would be charged $15+15+17=47$ seconds.

Table 3.1: Google vs. Azure Speech Recognizers

3.3.1 Names recognition challenges

Automatic speech recognition of people's names is a challenging task due to the long-tail distribution of the names and a large variety of different spellings or pronunciations. In this thesis, personalized human-robot interaction begins with querying a person's name, which is then stored in the database with all personalized information collected during the conversation. Therefore, recognizing a person's name is one of the key factors for person identification.

Challenges:

1. Long-tail distribution of the names. There are thousands of person names, including very rare names which are underrepresented or even missing in the training data set.
2. No commonly accepted spelling. The spelling of the same name might have several different variations, depending on the name origin, for example, name Aleksander:
 - Aleksander (Czech, Polish)

- Alexandre (French, Hungarian)
 - Alexandros (Greek)
 - Alsander (Irish)
 - Alessandro (Italian)
3. Different pronunciations. Names can be pronounced in a very different way, depending on a person's origin and dialect. For example, in Norway name, Martin can be pronounced as "Martin" or "Maatin" depending on the dialect.
 4. Not much context information. When the person is asked, "What is your name?" The name utterances can be as short as "Amelia" or with a little context "It is Amelia", which is not enough context that a language model can leverage. Name recognition can be improved with longer answers "My name is Amelia", but this is not a frequent way to introduce yourself.

3.3.2 Approach to improve name recognition

In our project we improved names recognition using 2 steps, first, we customized our speech-to-text default model by providing our specific domain-related words and phrases, and second we used phonetic similarities between domain-defined names and the prediction from Speech Recognizer. We compared 4 different algorithms that computes the phonetic key of a string: Soundex, NYSISS, Metaphone, and Double Metaphone.

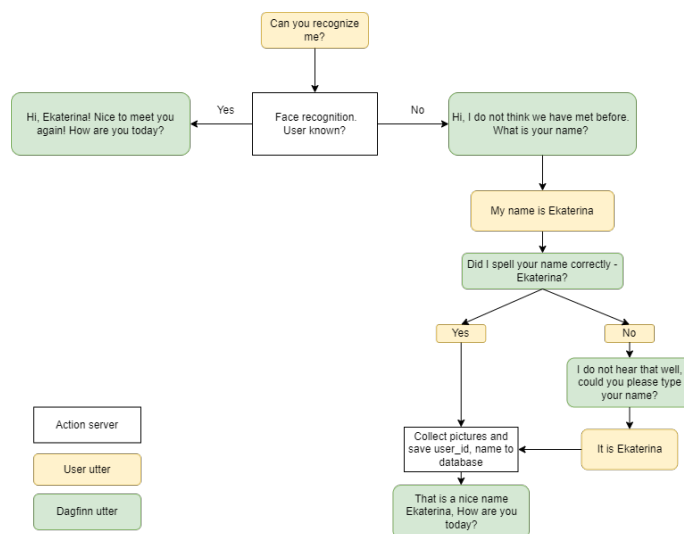


Figure 3.18: Dagfinn trying to recognize name

Chapter 4

Experimental Evaluation

In this chapter, we present the experimental results. We begin with datasets description in Section 4.1. Experimental setup and model parameters are described in Section 4.2. In Section 4.3, we present experimental results of our face recognition model and the name recognition improvement. Section 4.4 contains discussions on the topic.

4.1 Datasets

4.1.1 Labeled Faces in the Wild dataset

The Labeled Faces in the Wild dataset (LFW) [40] was created in 2007 to study the problem of unconstrained facial recognition, that is the variation in posture, expression, skin color, lighting, facial clothing, hair color and style, image quality, and other parameters. The dataset contains 13233 images of 5749 different identities. The size of the images is (250x250). This dataset is one of the most commonly used dataset to test face recognition models. We include this dataset as a reference for our model with a known accuracy of 99.38.

4.1.2 Yale Face Database A

Yale Face Database A [41] contains 15 individuals, with 11 images each. There are changes in light conditions, facial expressions and occlusions. Each image is 320x243 pixels. One set of images is presented below.

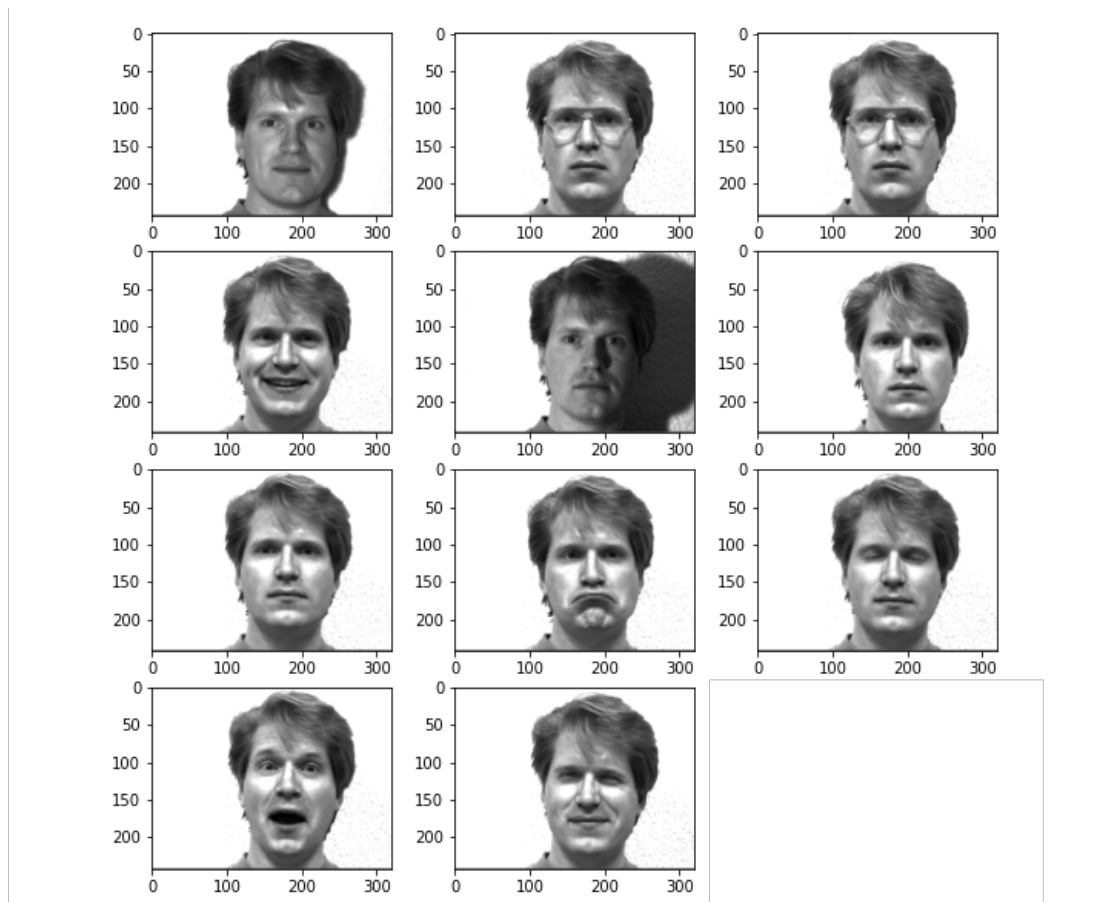


Figure 4.1: Yale Face Database A - example.

4.1.3 DAGFiNN dataset

DAGFiNN dataset is a custom dataset contained a set of 13 identities, with 10 pictures each. Dataset was collected from the DAGFiNN project group members and from experiment participants at the UiS. There was no specific background. Pictures were collected in good lightning conditions next to the big window and were not blurry. Pictures were taken every 4 frames from a real-time video streaming from Furhat camera, 10 most different (in terms of variations of facial expressions) images were selected. Most people had similar facial expressions. Size of images was varying from 49x49 to 129x129 pixels, depending on the distance (from 30cm to 120cm) between an individual and Furhat. Example of one individual is shown below on the Figure 4.2.

4.1.4 NORA dataset

NORA dataset was collected during the NORA conference and consists of 6 conference participants who volunteered in providing their pictures for our experiment. During the conference, Furhat was placed in a very dark hallway with weak light conditions,

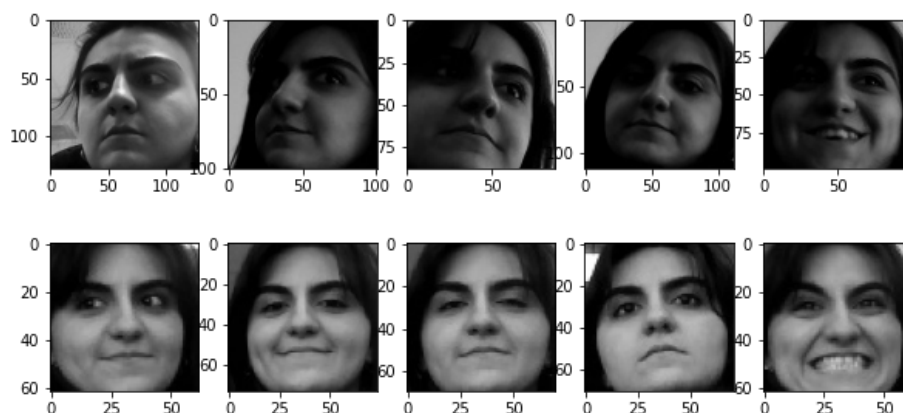


Figure 4.2: Example of DAGFiNN dataset

which extremely affected image quality and it was very interesting to evaluate model performance in this specific conditions. Many pictures were blurry, and the size of the images was varying from 40x40 to 109x109. Therefore 6 participants were asked to take extra images - 30 images each to provide bigger dataset for evaluation. This dataset was used to check how robust our model for bad image quality, weak illumination and small image size. One example of collected images for conference participant can be seen on Figure 4.3.



Figure 4.3: Example of NORA dataset.

4.2 Experimental Setup

The evaluation of DAGFiNN was performed in the setting displayed in the Figure 4.1. It consists of the robot forming the main interface with participants and screen used to

visualise the dialogue acts and alternatively take the input from the system users. The chat screen is displayed on our local PC with the following specifications:

- Processor Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz 2.90 GHz
- Installed RAM 16.0 GB (15.9 GB usable)
- System type 64-bit operating system, x64-based processor
- Edition Windows 10 Pro
- Version 21H2



Figure 4.4: DAGFiNN at Nora Conference

To facilitate the smooth flow of the experiment, all the participants were given a list of questions DAGFiNN is familiar with as well as the figure with possible discussion topics, depicted in the picture below.

Prior any conversations participants were informed about collection of personal information, such as images and names, in this thesis. The consent form provided to and signed by participants was constructed based on the EU General Data Protection Regulation



Figure 4.5: DAGFiNN's Discussion Topics

entered into law in December 2018. As soon as consent was acquired, participants were asked to initialise the conversation.

If the participant uttered a sentence with intent "recognize a user", RASA's form, in this case name form, is activated to get the name as the input. Once the name is correctly identified, DAGFiNN starts to take the picture of participant, every 4th frame from real-time video obtained from SocketIO custom channel, and stores it locally on computer. Later, it assigns an id to participant and writes the name corresponding to id in database. This scenario is valid if DAGFiNN and participant have met before. Once the training data is acquired, the ResNet model is used to obtain the facial embedding of respective participant. Hence, when a known user approaches DAGFiNN, it is able to match the face to known embeddings and utter the name of participant, it recognized.

Alternatively, if user denies to provide personal data, it can start conversation by asking DAGFiNN to read his/her QR Code. As soon as "recognize by qr code intent" is detected, DAGFiNN asks participant to show it closer and reads the link to one's business card, which contains the id of the participant. Using this id, DAGFiNN queries the database by ID key and fetches all the respective information about the approacher. This information is further used to personalize conversations. Another attempt of personalisation is build

based on memorizing entities in the course of actual conversation and recalling them when necessary. Example is, country of origin, once clarified, DAGFiNN uses it to personalise weather information to reflect one's country of origin information as well.

After each participant finalizes its conversation, he/she is asked to rate the experience by rating DAGFiNN on 5-point Likert scale.

4.3 Experimental Results

In this section we will first present results obtained for personalization in the Section 4.3.1, qr code versus face recognition in the Section 4.3.2, and name recognition in the Section 4.3.3.

4.3.1 Personalization

The resulting scenario to personalize conversations conversation after face recognition is shown in figures 4.6 and 4.7. Once the familiar faces are recognized, DAGFiNN responds to "recognise a user" intent by uttering the matched names. It can identify and refer to several participants in one frame.

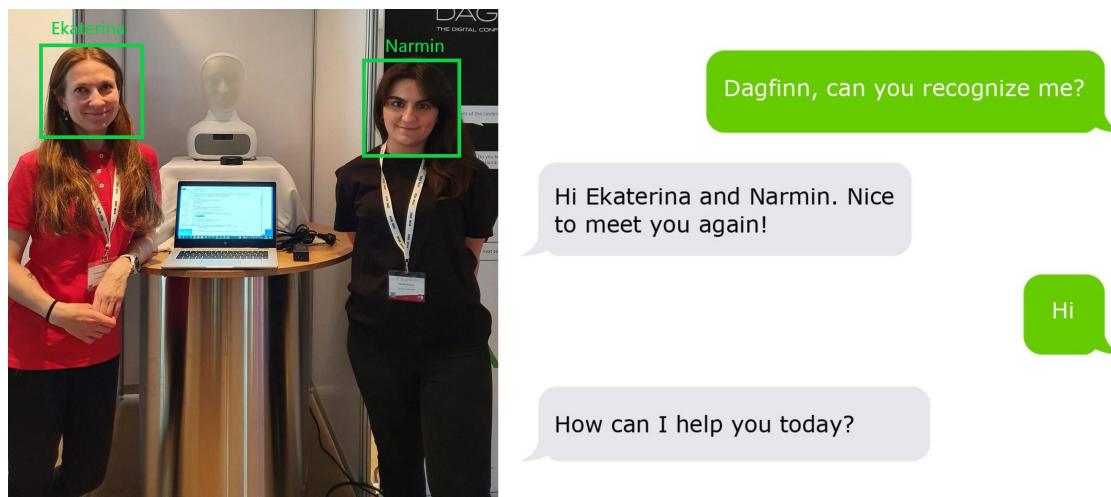


Figure 4.6: The scenario with known faces

In case if DAGFiNN meets participant for the first time, it asks for his/her name. If several participants are unknown DAGFiNN will ask for the name of each of them. This is also valid for the scenario where the frame contains known and unknown participant pair. In this case, DAGFiNN welcomes the known user and asks for the name of unknown.

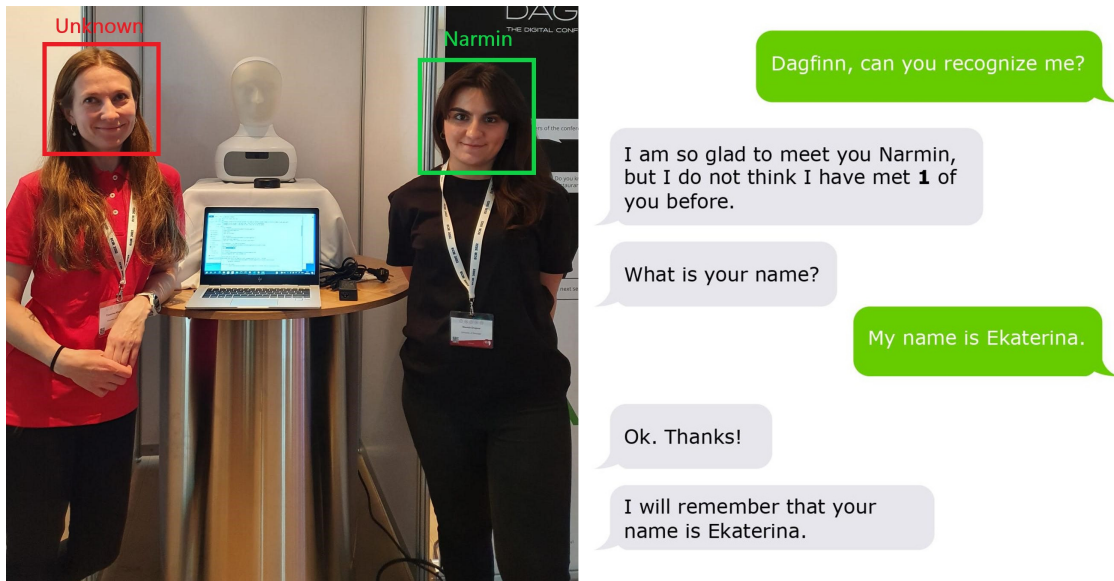


Figure 4.7: Scenario with unknown face

4.3.2 Face vs. QR-code recognition

As it was mentioned in the background section, Face Recognition consists of 3 steps, which are face detection, feature extraction, and face recognition. In this subsection we will first briefly recap these steps and methods used. Then we will elaborate on the parameter selection for kNN classifier and threshold metrics for distance calculation. Afterwards, through subsections 4.3.2.1-4.3.2.3 we will summarize the performance of models on each of the datasets described above.

Techniques	Feature extraction	Execution time (s)	Accuracy Lab test result
Viola-Jones Algorithm (OpenCV)	HAAR Features	0.011	0.94
HOG + Linear SVM face detector (Dlib)	Hog Features	0.016	0.99
Max-Margin (MMOD) CNN face detector (Dlib)	CNN Features	1.75	0.99

Figure 4.8: Face detection methods comparison.

For real-time face detection projects both accuracy and execution time play significant role. After the literature review, 3 algorithms for face detection were chosen: Viola-Jones Algorithm, Histogram of Oriented Gradients and Convolutional Neural Networks. Performance evaluation at the lab settings are displayed in the Figure 4.8. Execution time was measured as average from 100 processed frames from Furhat camera of a single person in different poses and lightning conditions. The least accurate with algorithm,

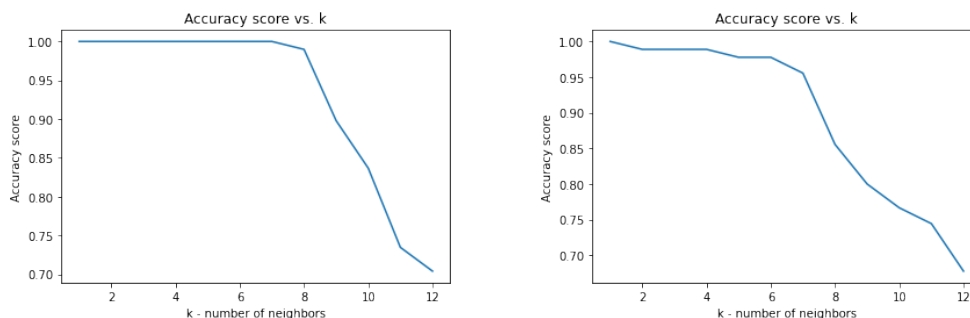


Figure 4.9: The variation of model accuracy based on k parameter of k-NN a) Yale dataset, b) left: on custom service

which showed 0.94 accuracy was Viola-Jones Algorithm. This algorithm has another disadvantage - often predicts false positive faces. HOG and CNN algorithms showed the same accuracy, but HOG outperformed CNN in computational speed being 100 time faster. HOG algorithm was chosen for users' recognition step as best combination of execution time and accuracy.

Facial features extraction

For each image we create 128-d embeddings, using dlib's state-of-the-art face recognition library based on ResNet pretrained deep learning model. The model has an accuracy of 99.38 % on the Labeled Faces on the Wild benchmark dataset.

Face recognition using classifiers

For Face recognition we selected 2 models - kNN and SVM. We performed evaluation of the models in terms of accuracy and execution time. The next subsections first show analysis on optimal parameter selection and later the analysis of overall model performance.

Parameter Sensitivity Analysis

The parameters evaluation in this work was the number of neighbors for kNN classifier, displayed in the Figure 4.9 and the distance threshold used to assign a sample to the class, shown in the Figure 4.10.

To find the optimal number of neighbours we have used grid search with 10-fold cross validation. We searched for the k parameter on the Yale dataset and the one collected by us, the results for both are shown in the left and right plots respectively. As we observe from the plots on Yale dataset the accuracy level is maintained at 1 in up to 6 neighbours, whereas, in case of our self-collected dataset, the accuracy drops already at 2 neighbours. The reason for this can potentially be the number of examples in the training data. Hence, we concluded on optimal k value of 1.

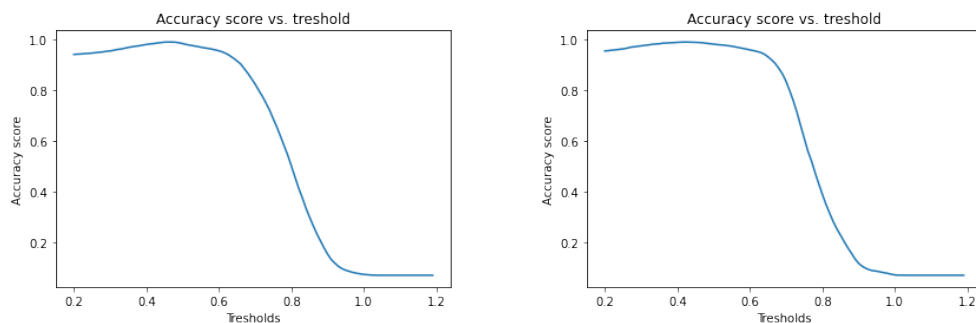


Figure 4.10: The variation of model accuracy based on right a) Yale dataset, b) left: on custom service

Further, we have analysed the optimal threshold value as shown in plots of the Figure 4.10. This analysis was also done for Yale and self-collected dataset separately. From the both plots we can see that with threshold value of 0.42 we are reaching highest accuracy score.

As stated above, we will present evaluation results of our selected model from dlib library when tested on LFW, Yale Face Database A, DAGFiNN dataset and NORA dataset. After that, we will include visualisation of our results on each of the tested dataset. Accuracy was calculated as average accuracy from 100 iterations from kNN-classifier with $k=1$ and using test/train split as 0.1/0.9. The classifier selection and model parameters selection will be discussed further below.

Dataset	Accuracy, %
LFW	99.38%
Yale Face Database A	99.32%
DAGFiNN Dataset	97.44%
NORA Dataset	94.82%

Table 4.1: Results on all available datasets

4.3.2.1 Yale Face Database A

The main purpose of the face recognition implementation was to maximize accuracy and minimize computational costs. To achieve this first we have investigated model performance on the Yale Face Database A, which was close to our needs in terms of amount of samples and number of individuals.

Dataset was split at train/test as 0.1/0.9 ratio and for training every class had only 1 image.

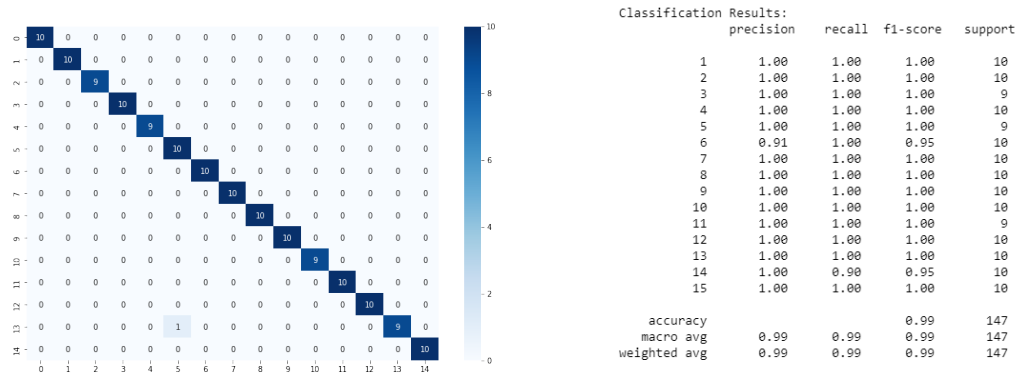


Figure 4.11: a) Confusion matrix, b) Classification report

Model has been trained and test results can be seen in the confusion matrix.

We also compared model accuracy as a function of split ratio and results are shown on the Figure 4.12.

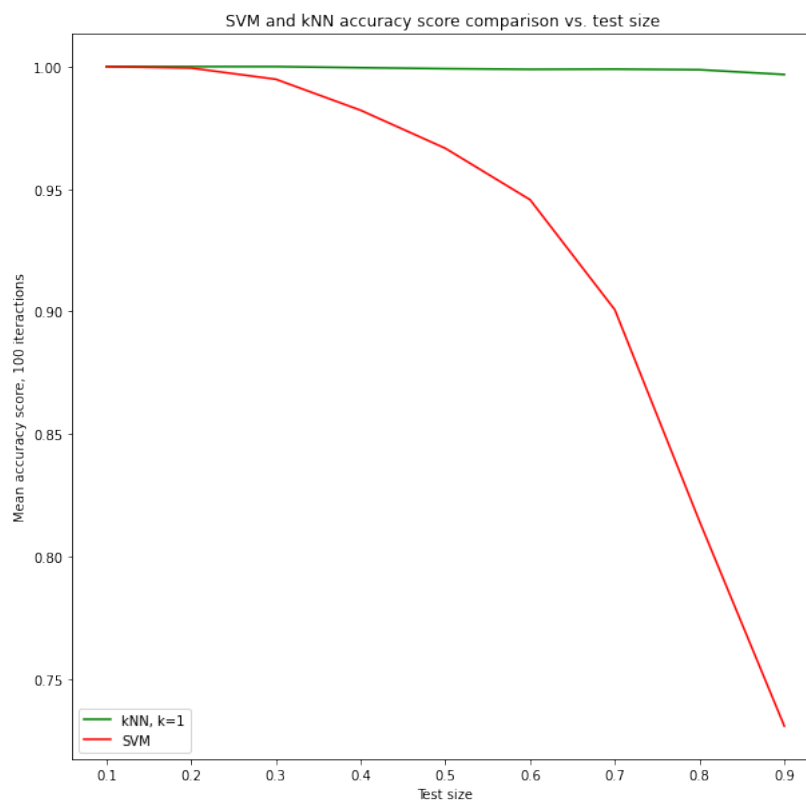


Figure 4.12: Comparison of SVM and kNN classifiers as a function of test size

As shown on the the Figure 4.12 above, kNN classifier performs excellent even on only 1 image training example and able to classify test dataset with nearly perfect accuracy.

4.3.2.2 DAGFiNN dataset

Second experimental dataset was collected using Furhat's camera and due to low camera resolution images were of much smaller size compared to the Yale Face Database A.

For illustration purpose, we have used t-distributed stochastic neighbor embedding (t-SNE) visualization method to be able to reduce 128d high-dimensional data vectors and plot them on a two-dimensional map.

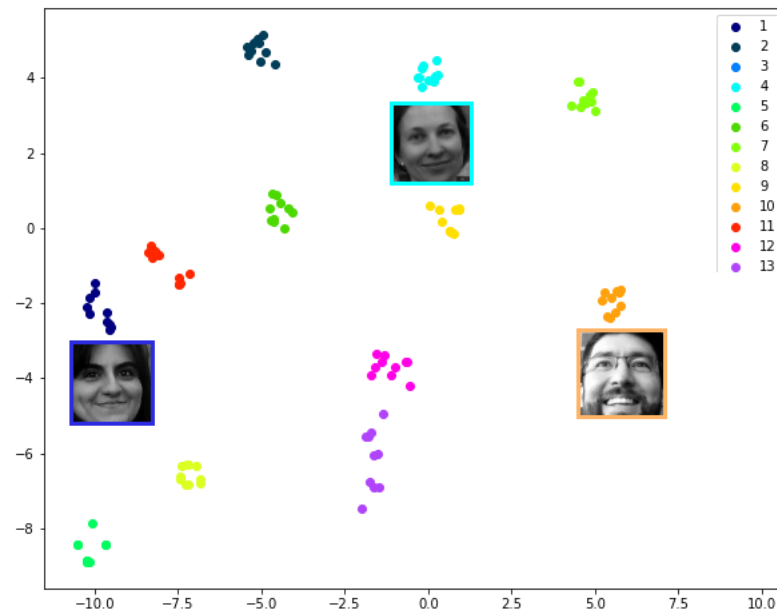


Figure 4.13: t-SNE visualization of DAGFiNN dataset

As can be clearly seen from the Figure 4.15 our classes are very well separated even in a two-dimensional plane. Similar comparisons have been performed on this dataset. We will in addition illustrate coefficient selection for kNN classifier.

As a last step, we have computed the distances between all images and presented as histogram distributions on Figure 4.14.

Success/Failure Analysis

After selecting the parameters we evaluated the kNN and SVM classifiers on its success to correctly assign the detected faces. Both models showed extremely good accuracy of 1 on the DAGFiNN dataset even when only 1 image of each class was used in the training process. The accuracy numbers for both models are shown in tables ?? and ?. We see that in general kNN with $k=1$ slightly outperforms SVM model, which can be due to the

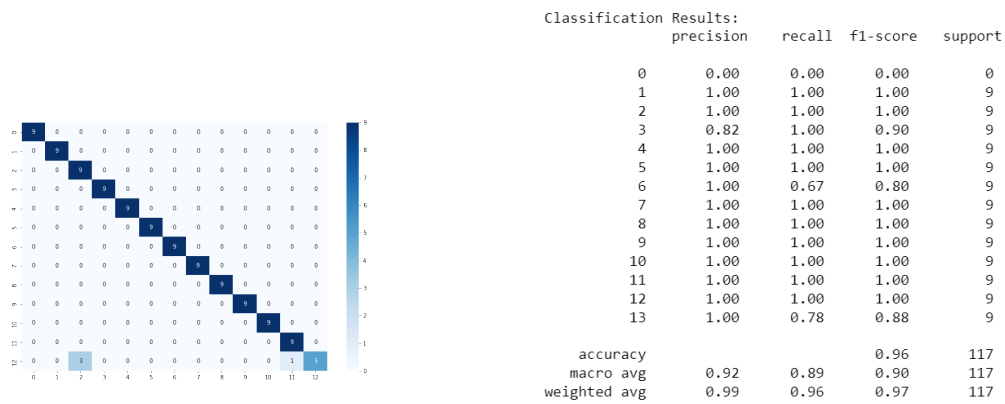


Figure 4.14: a) Confusion matrix, b) Classification report

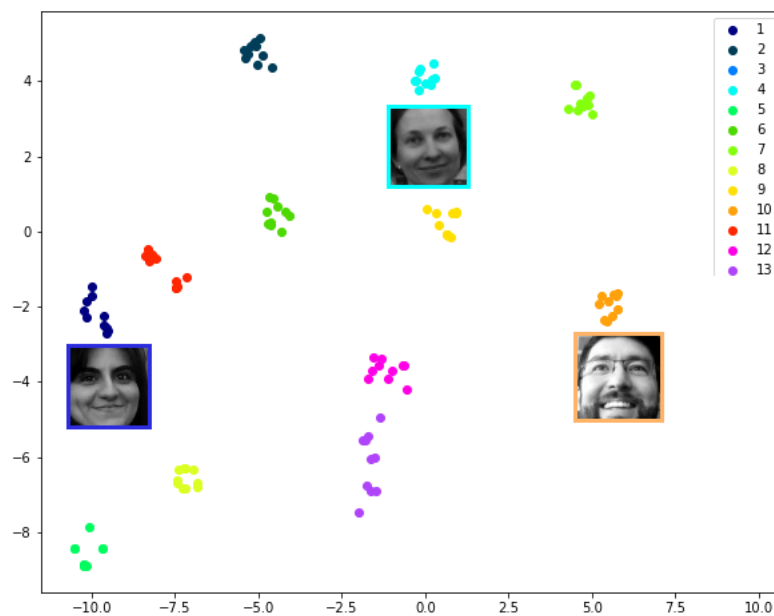


Figure 4.15: t-SNE visualization of DAGFiNN dataset

small amount of training examples. Main goal of the model selection was to maximize accuracy and minimize computational cost.

Model	Accuracy, %	5-fold cross validation, %
kNN, k=1	99.32%	100%
SVM	84%	98%

Table 4.2: Results on Yale dataset

Model	Accuracy, %	5-fold cross validation, %
kNN, k=1	98%	100%
SVM	99%	89%

Table 4.3: Results on DAGFiNN dataset

Comparison of QR code and Face Recognition

As it was mentioned before in this work there were two routes takes for participants recognition: QR code Reading and Recognition based on model. In this subsection we will compare both methods on their speed and will list advantages and disadvantages of both.

In this work we have tested QR codes of varying size at different distances from the Furhat camera to determine the optimal qr code size and distance. To results showed that a qr code of size 5×5 is readable on the distance of 53 sm, whereas qr of the size 2×2 - 10 sm. Overall, it takes around 1 seconds to recognize the code at the distance of 53 sm. When it comes to the 10 sm distance, due to the wide field of view (FoV) of Furhat's camera, reading takes more time, approximately 20 seconds. Model recognition whereas, takes on average 0.7 seconds.

Based on this we can conclude that model recognition is faster. At the same time, QR code recognition is more robust and secure, as it does not require the share of personal information.

4.3.3 Name Recognition

To compare name recognition ability of DAGFiNN we have chosen 100 representative international names and recorded their utterance. Next we tested their recognition on Google Cloud and Microsoft Azure Recognizers. First we tried to utter just the names, then we pronounced names within the expression "My name is ...". We have noticed the increase in recognition rate for later scenario. The results for this are shown in the Figure 4.16 for Google Cloud and the Figure 4.17 for Microsoft Azure. The results for different utterance phrases are color-coded.

From both plots we see that for Google Cloud the original recognizer is able to understand the names with 46% of accuracy, and the phrase "My name is ..." with 54% accuracy. These numbers are improved by sending list of the names to NLU component before utterance. Then numbers become 76% and 80% respectively. Further improvement is achieved by use of metaphones, which results in 81% and 88% accordingly. This shows

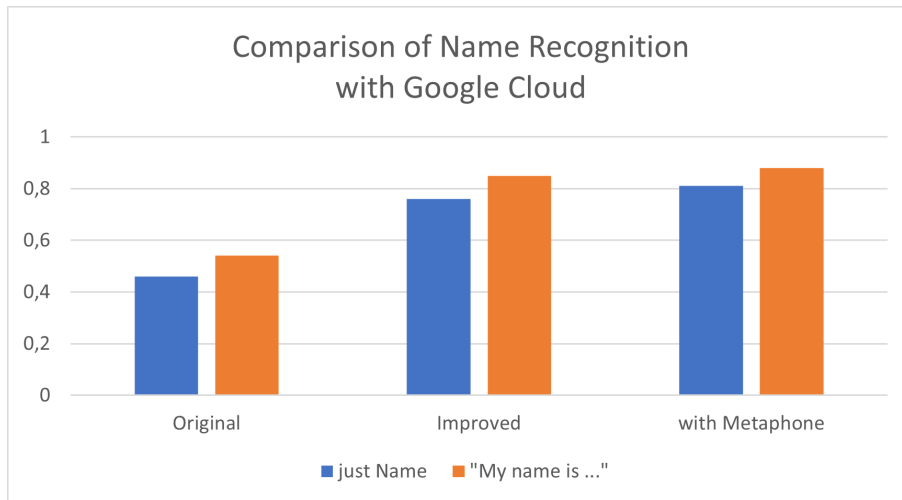


Figure 4.16: The accuracy of name recognition with Google Cloud

that we were able to achieve significant improvement by make it recognize twice more names.

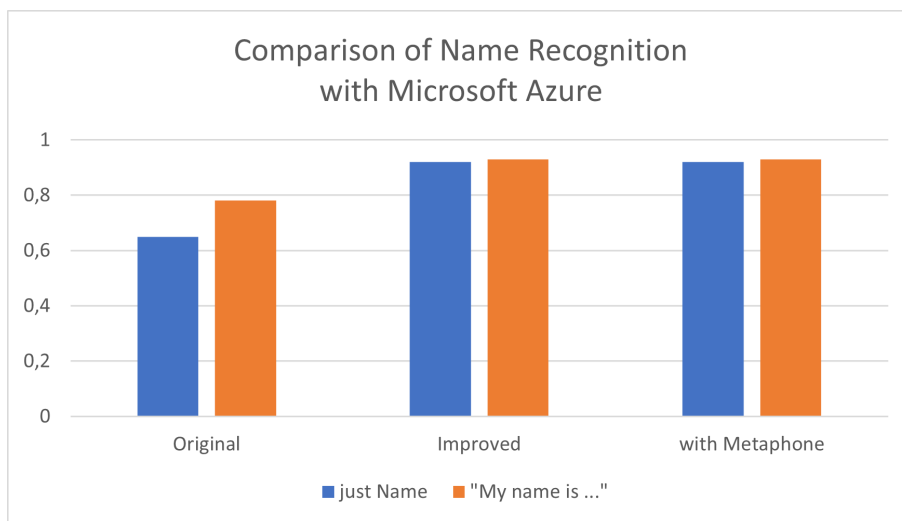


Figure 4.17: The accuracy of name recognition with Microsoft Azure

The same trend is observed for Microsoft Azure Recognizer with the main difference that Azure was more accurate from the beginning. Hence, the numbers are 65% and 78% for just name and phrase "My name is ...". For second improvement 92% for both and 93% for third improvement. Hence, we decided to choose Microsoft Azure as the recognizer for experiment.

4.4 Discussions

In this section we will summarize the results of the experiment and discuss the findings. As it was mentioned before, at the end of the experiment we have asked the participants to rate the experience with DAGFiNN on how it was perceived, such as friendly/unfriendly, intelligent/unintelligent. Also, we have mentioned at the beginning of this work that DAGFiNN was presented at two conferences. The first conference did not include personalized conversations and when we compare the statistics for both of this conferences, we can see a definite trend. The conversations during second conference have included more turns, which allows us to conclude that personalisation did have influence on engagement of participants. This could also have been observed from the high positive ratings on overall impression of DAGFiNN as well. However, it worth also mentioning that people approaching were also impressed by the robot itself, hence the rating cannot be analysed as independent index of satisfaction level with DAGFiNN.

Chapter 5

Conclusions

This Chapter summarizes the work done in our thesis by reflecting back to RQs defined at the beginning of the journey in Section 5.1. We conclude the thesis by setting directions for future work in Section 5.2.

5.1 Conclusion

Personalised conversational systems are used to assist people to extend their capabilities through utilization of advanced technologies. One potential application of such systems is a conference assistant, directing the participants through the busy schedule of academic conferences and giving constructive recommendations. To make the experience with this type of assistant natural and positive in this thesis we tried to personalise the human-robot dialogues by means of participant recognition, his/her name memorization, and reference to the personal details mentioned in the course of conversation.

To construct personalised dialogues we memorized the face of participants after acquiring their consent. In case of denial, participants were recognized by reading QR code printed on their conference badges. The performance characteristics of both of these methods were later compared to list the advantageous scenarios for both. Next step on the way to personalisation involved name memorization. Due to the speech recognizer limit of used social robot, amount of work was spent on identification of its improvement methods, and the based on the comparison the optimal recognizer was identified and utilized further in the project.

As the last step of personalisation, participants related data was collected during the conversations and was used along with recognised face and name to personalise the uttered responses of the robot. The performance of the robot was evaluated during the

two conferences in Stavanger, Norway. Additionally, an experiment was set up at the University of Stavanger. In both cases, user's satisfaction level was inquired through designed questionnaire. Participants responses was lastly analysed to infer the influence of personalised conversations on their satisfaction level with robot interaction. What exactly we have done?

RQ1a: How can the name recognition using a black box speech-to-text component be improved?

We have showed that name recognition ability of robot can be improved first by sending a list of common international names to NLU component and second by comparing the metaphones of candidate names with the database of known names before choosing a potential name.

RQ1b: How well does out-of-the-box recognition work for recognizing participants' names?

We demonstrated that in general Microsoft Azure Speech Recognizer has better performance in terms of speed and accuracy with no cost of additional charge. Moreover, we found that by using the method mentioned in previous paragraph one can significantly improve the recognition ability of already good Azure Recognizer.

RQ2: How do participants recognition using a pre-trained neural model with facial encodings compare to the QR-code-based recognition?

We found that QR code reading is more robust in participants correct recognition. However, in terms of speed it loses to recognition through a facial encodings. Due to the wide angle of robot's camera in some scenarios it can take significantly higher amount of time to locate and decode QR code. The advantage of this method is its GDPR compliance.

RQ3: How does personalization in conversation influence the participants' satisfaction level?

Based on the ratings provided by experiment participants and judging from the perspective of DAGFINN's popularity on the conference and experiments, we inferred that personalisation increases the participants satisfaction level with robot interaction.

5.2 Future Directions

There are several directions this thesis can be taken forward. We are listing the ones we came up with.

- The experiment in this thesis considered the overall impression of the robot. The rating of the participants could have been influenced by the effect of the robot itself. One potential future work can be conduction of experiments on controlled groups to acquire more independent view of user's satisfaction.
- It took some time for us to set up the whole system, hence we were not able to construct the working personalised assistant for the first conference. For that conference we had a database with personal information of participants available beforehand. This information could have been used to construct more personalised dialogue flows, such as the robot's response to question "How are you?" could have been different for each of participant type: sponsor, organizer, participant. It could have uttered to participant "Great, as I am meeting so many interesting people as you". In case of organiser the response could have been "I am tired, but guess not as much as you". This in theory would have made the dialogue even more natural.

Bibliography

- [1] Eva Blessing Onyeulo and Vaibhav Gandhi. What makes a social robot good at interacting with humans? *Information*, 11(1):43, 2020.
- [2] Cynthia Breazeal. Emotion and sociable humanoid robots. *International journal of human-computer studies*, 59(1-2):119–155, 2003.
- [3] Yasushi Nakauchi and Reid Simmons. A social robot that stands in line. *Autonomous Robots*, 12(3):313–324, 2002.
- [4] Kathrin Pollmann and Daniel Ziegler. Social human-robot interaction is personalized interaction. *BEHAVIORAL PATTERNS AND INTERACTION MODELLING FOR PERSONALIZED HUMAN-ROBOT INTERACTION 2020*, 2020.
- [5] Nikhil Churamani, Paul Anton, Marc Brügger, Erik Fließwasser, Thomas Hummel, Julius Mayer, Waleed Mustafa, Hwei Geok Ng, Thi Linh Chi Nguyen, Quan Nguyen, et al. The impact of personalisation on human-robot interaction in learning scenarios. In *Proceedings of the 5th International Conference on Human Agent Interaction*, pages 171–180, 2017.
- [6] Maarten H Lamers and Fons J Verbeek. *Human-Robot Personal Relationships: Third International Conference, HRPR 2010, Leiden, The Netherlands, June 23-24, 2010, Revised Selected Papers*, volume 59. Springer, 2011.
- [7] Min Kyung Lee, Jodi Forlizzi, Sara Kiesler, Paul Rybski, John Antanitis, and Sarun Savetsila. Personalization in hri: A longitudinal field experiment. In *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 319–326. IEEE, 2012.
- [8] Deborah L Johanson, Ho Seok Ahn, Craig J Sutherland, Bianca Brown, Bruce A MacDonald, Jong Yoon Lim, Byeong Kyu Ahn, and Elizabeth Broadbent. Smiling and use of first-name by a healthcare receptionist robot: effects on user perceptions, attitudes, and behaviours. *Paladyn, Journal of Behavioral Robotics*, 11(1):40–51, 2020.

- [9] Bahar Irfan, Mehdi Hellou, Alexandre Mazel, and Tony Belpaeme. Challenges of a real-world hri study with non-native english speakers: Can personalisation save the day? In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, pages 272–274, 2020.
- [10] Claudio Baecchi, Andrea Ferracani, and Alberto Del Bimbo. User profiling and context understanding for adaptive and personalised museum experiences. *DigitCult-Scientific Journal on Digital Cultures*, 4(2):15–28, 2019.
- [11] Angelo Costa, Ester Martinez-Martin, Miguel Cazorla, and Vicente Julian. Pharos—physical assistant robot system. *Sensors*, 18(8):2633, 2018.
- [12] Yasuyuki Sumi and Kenji Mase. Digital assistant for supporting conference participants: An attempt to combine mobile, ubiquitous and web computing. In *International Conference on Ubiquitous Computing*, pages 156–175. Springer, 2001.
- [13] Daniel Jurafsky James H. Martin. *Chatbots Dialogue Systems*, volume 59. Springer, 2011.
- [14] Krisztian Balog, Lucie Flekova, Matthias Hagen, Rosie Jones, Martin Potthast, Filip Radlinski, Mark Sanderson, Svitlana Vakulenko, and Hamed Zamani. Common conversational community prototype: Scholarly conversational assistant, 2020.
- [15] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35, 2017.
- [16] What is bio tagging? . Accessed: 2022-02-28.
- [17] Rasa. <https://rasa.com/docs/>. Accessed: 2022-02-28.
- [18] Leonardo’s robot. https://en.wikipedia.org/wiki/Leonardo%27s_robot. Accessed: 2022-06-06.
- [19] Hanson robotics. <https://www.hansonrobotics.com/>. Accessed: 2022-06-06.
- [20] Moxie. <https://embodied.com/>. Accessed: 2022-06-06.
- [21] Furhat developers docs. <https://dblp.org/rec/bib/journals/tdsc/AndersonMRVM17>. Accessed: 2022-06-06.
- [22] Raul Benites Paradedda, Mojgan Hashemian, Rafael Afonso Rodrigues, and Ana Paiva. How facial expressions and small talk may influence trust in a robot. In *International Conference on Social Robotics*, pages 169–178. Springer, 2016.

- [23] Christian F Hempelmann. Computational humor: Beyond the pun? *The Primer of Humor Research. Humor Research*, 8:333–360, 2008.
- [24] Sara Moussawi and Raquel Benbunan-Fich. The effect of voice and humour on users' perceptions of personal intelligent agents. *Behaviour & Information Technology*, 40(15):1603–1626, 2021.
- [25] Amanda Purington, Jessie G Taft, Shruti Sannon, Natalya N Bazarova, and Samuel Hardman Taylor. " alexa is my new bff" social roles, user satisfaction, and personification of the amazon echo. In *Proceedings of the 2017 CHI conference extended abstracts on human factors in computing systems*, pages 2853–2859, 2017.
- [26] History of qr code. <https://www.qrcode.com/en/history/>, . Accessed: 2022-06-06.
- [27] The history of qr code. <https://www.youtube.com/watch?v=k09ip9Z6TCk>, . Accessed: 2022-06-06.
- [28] The components of qr code. <https://www.beaconstac.com/what-is-qr-code>, . Accessed: 2022-06-06.
- [29] Qr code correction factor. <https://scanova.io/blog/blog/2018/07/26/qr-code-error-correction/>, . Accessed: 2022-06-06.
- [30] The correction in qr code. <https://blog.qrstuff.com/qrstuff-features/qr-code-error-correction>, . Accessed: 2022-06-06.
- [31] What is micro qr code. <https://www.qrcode.com/en/codes/microqr.html>, . Accessed: 2022-06-06.
- [32] Proyecto Fin De Carrera and Ion Marques. Face recognition algorithms. *Master's thesis in Computer Science, Universidad Euskal Herriko*, 1, 2010.
- [33] The minimum size of qr code. <https://blog.qrstuff.com/qrstuff-features/qr-code-minimum-size>, . Accessed: 2022-06-06.
- [34] Comparison of face recognition libraries. <https://towardsdatascience.com/what-is-the-best-facial-recognition-software-to-use-in-2021-10f0fac51409>. Accessed: 2022-06-06.
- [35] Comparison of face recognition libraries. <https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cfff>. Accessed: 2022-06-06.
- [36] Comparison of face recognition libraries. <https://www.width.ai/post/facial-detection-and-recognition-with-dlib>. Accessed: 2022-06-06.

-
- [37] Comparison of face recognition libraries. <http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html>. Accessed: 2022-06-06.
- [38] Comparison of face recognition libraries. <https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4>. Accessed: 2022-06-06.
- [39] Comparison of face recognition libraries. <https://developers.google.com/machine-learning/gan/generative>. Accessed: 2022-06-06.
- [40] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.
- [41] Peter N. Belhumeur, Joao P Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):711–720, 1997.