San Jorge University

School of Architecture and Technology

Graduate in Computer Engineering

Final Project

Emotion detection, processing and response through human-machine interaction system modelling

Project author: Marcos Caballero

Professional project manager: Rafael del Hoyo Alonso

Academic director of the project: Jesús Carro

Villanueva de Gállego, 17th June 2021

universidad SANJORGE GRUPO SANVALERO

This work constitutes part of my candidature for the degree of Graduate in Computer Engineering from the San Jorge University and has not been submitted previously (or simultaneously) for the award of any other degree.

This document is the result of my own work, except where otherwise indicated and referenced.

I give my consent for this work to be archived in the University Library of San Jorge University, where it can be made available for consultation.

Signature	Date
M Caballero	17th June 2021

Tribute and Acknowledgement

A ITAINNOVA por poder acceder a concederme una beca para PFG aun con los riesgos que supone dada la situación sanitaria que estamos atravesando.

A toda la comunidad open-source con mención especial para Sefik Ilkin Serengil por su framework DeepFace y toda su plataforma e-learning asociada y a Yuichiro Tachibana (Tsuchiya) por su componente Streamlit Web-rtc. Me parece algo heroico que estas personas empleen una cantidad considerable de su tiempo y esfuerzo para mejorar el mundo sin pedir nada a cambio.

A mis tutores Rafael del Hoyo por, aun teniendo que atender mil cosas a la vez, saberme guiar tan bien de manera tan empática; eres con los alumnos como un segundo padre, y a Jesús Carro por aportar siempre ese punto de vista profesional, positivo y con un gran valor añadido que logra elevar la excelencia de la persona y del trabajo que está haciendo.

A mi abuela Pascuala por tanto insistirme en la importancia de "meter codicos" y prepararme lo mejor posible para mi futuro, demostrándome desde su propia vivencia lo que es tener tesón, determinación y fuerza de voluntad.

Summary

Este proyecto consiste en un modelo de detección, procesamiento y respuesta de emociones mediante un sistema de interacción hombre-máquina. Nace de la necesidad de humanizar más la manera en que humanos y máquinas se relacionan entre ellos mejorando los ya existentes asistentes virtuales. Dota a los mismos mediante grafos de conocimiento e inteligencia artificial de una manera de asimilar su pensamiento al de los humanos. Al tratarse de un proyecto de investigación aplicada, se emplea la llamada POC (prueba de concepto en inglés), que pretende estudiar la viabilidad de la idea de negocio sin seguir las mismas directrices metodológicas de un proyecto típico de software. El resultado final ha sido satisfactorio: se ha conseguido interactuar con Pilar mediante un chatbot via Telegram, Web y el mismo avatar pudiéndole preguntar cómo se siente o a quién ha visto, ver las emociones de las personas frente a sus ojos (una webcam), cómo afectan a su temperamento en un dashboard en tiempo real, y extraer valor de los grafos de conocimiento mediante las citadas aplicaciones.

Key Words

Artificial Intelligence, Computer Vision, Virtual Assistants, Face Attributes Detection, Affective Computing, Chatbots, Data Science Dashboards, Avatars, Knowledge Graphs.

Contents Table

Summary **3**
Key Words **3**
Illustrations Table **5**
1. Introduction **11**
2. Background & State of Art **12**
2.1. Computer Vision State of the Art **16**
2.2. Emotion Detection (DeepFace) **18**
2.2.1. Face Recognition pipeline *18*
2.2.2. Which face recognition model is the best? *19*
2.2.3. Facial Attribute Analysis *24*
2.3. Emotion Model (ALMA) **26**
2.4. Events Storage in Graphs (Neo4j) **29**
2.4.1. What, When, How, Why? *29*
2.4.2. Modelling Data *29*
2.4.2.1. Modelling Events in Time *30*
2.4.2.1.1. The event-grammar approach *31*
2.4.2.1.2. The event graph approach *31*
2.4.2.1.3. The 'denormalised' graph approach *31*
2.4.2.2. Modelling Time in Events *32*
2.5. Interactive Dashboards (Streamlit) **32**
2.6. Chatbots (Rasa) **33**
2.6.1. How Rasa works *34*
2.7. Sentiment Analysis **36**
3. Objectives **40**
4. Methodology **41**
4.1. Background information **42**
4.2. Agile methodology **43**
4.3. My custom Scrum solution **44**
4.3.1. Roles *45*
4.3.2. Artifacts *45*
4.3.3. Events *45*
4.4. Work Plan **46**
5. Economic Study **48**
6. Analysis, Design & Development **50**
6.1. Architecture Design **50**
6.2. UML Diagrams **51**
6.2.1. Use Case Diagram *51*
6.2.2. Class Diagram *51*
6.2.3. Sequence Diagram *52*
6.3. Investigation task (Jupyter Notebooks) **54**
6.4. Setting up the environment **54**
6.5. Timeline **56**
6.5.1. Summer – October 2020 .... *57*
6.5.2. October 2020 – April 2021 .... *57*

6.5.3. April – June 2021 ... 58
6.6. Emotion Detection (Deepface) ... 58
6.6.1. Tensorflow + Cuda ... 59
6.7. Emotion Model (ALMA) ... 60
6.8. Events Storage (Neo4j) ... 61
6.8.1. Modelling my use case ... 61
6.8.2. Queries ... 62
6.9. Interactive Dashboard (Streamlit) ... 63
6.9.1. Streamlit at low level ... 63
6.9.2. Streamlit components ... 63
6.10. Chatbot (Rasa) ... 65
6.10.1. Rasa Open Source ... 65
6.10.2. Rasa X ... 66
6.10.3. Rasa Action Server ... 67
6.10.4. Integrations ... 67
6.11. Pilar (Webservice & Unity) ... 68
7. Results ... 68
7.1. Emotion Detection (DeepFace) ... 68
7.2. Interactive Dashboard (Streamlit) ... 71
7.3. Pilar (Unity) ... 72
7.4. Telegram bot ... 73
8. Conclusion ... 73
8.1. Future ... 73
8.1.1. Technological point of view ... 73
8.1.2. Strategical/Business point of view ... 75
8.2. Personal Opinion ... 75
9. Annexes ... 76
9.1. Pilar (Avatar Unity) ... 76
9.2. Logbook ... 76
9.2.1. Summer – October 2020 ... 76
9.2.2. October 2020 – April 2021 ... 77
9.2.3. April – June 2021 ... 77

Illustrations Table

Figure 1 General Project Schema (Own Copyright) ... 14
Figure 2 Modern Face Recognition Pipeline Stages (Copyright Sefik Ilkin Serengil) ... 18
Figure 3 Face Recognition Alignment Accuracy Improvement (Copyright Facenet paper) ... 19
Figure 4 Face Recognition Models & Distance Metrics Comparison (Copyright Sefik Ilkin Serengil) ... 19
Figure 5 Face Recognition Models Testing Metrics Comparison (Copyright Sefik Ilkin Serengil) 20
Figure 6 Computer Vision Models Error Rate Throughout Time (Copyright Sefik Ilkin Serengil) 21
Figure 7 Mean Distribution of Positives and Negatives (Copyright [15]) ... 24
Figure 8 Keras Facial Expression Recognition Confussion Matrix (Copyright [16]) ... 25
Figure 9 Auto-Keras Facial Expression Recognition Confussion Matrix (Copyright [16]) ... 25
Figure 10 Graphical Representation of Regresion for Age and Gender Prediction (Copyright Sefik Ilkin Serengil) ... 25

Figure 11 Confussion Matrix for Race and Ethnicity Prediction Model (Copyright Sefik Ilkin Serengil) 26
Figure 12 Mood Octants of the PAD Space (Copyright ALMA Paper) 27
Figure 13 Affect Space Features Calculation (Copyright ALMA Paper) 27
Figure 14 Mapping of OCC Emotions into PAD Space (Copyright ALMA Paper) 28
Figure 15 Pull and Push Mood Change Functions (Copyright ALMA Paper) 28
Figure 16 Property Graph Model (Copyright [5]) 30
Figure 17 Event Grammar Graph Approach (Copyright [5]) 31
Figure 18 Event Graph Approach (Copyright [5]) 31
Figure 19 Denormalised Graph Approach (Copyright [5]) 31
Figure 20 Modelling Time in Events (Copyright [6]) 32
Figure 21 Rasa Features (Copyright Rasa) 34
Figure 22 Rasa Architecture (Copyright Rasa) 34
Figure 23 Future Direction in Sentiment Analysis (Copyright [15]) 37
Figure 24 Kanban Board (Copyright Visual Paradigm) 44
Figure 25 Lean Principles (Copyright Visual Paradigm) 44
Figure 26 Scrum Components (Copyright Visual Paradigm) 45
Figure 27 Weekly and Daily Meetings (Own Copyright) 46
Figure 28 System Architecture (Own Copyright) 50
Figure 30 Use Case Diagram (Own Copyright) 51
Figure 29 Class Diagram (Own Copyright) 52
Figure 31 Sequence Diagram 1 (Own Copyright) 53
Figure 32 Sequence Diagram 2 (Own Copyright) 53
Figure 33 Fork Repository Sample (Own Copyright) 55
Figure 34 Development Timeline Gannt Diagram (Own Copyright) 57
Figure 35 DeepFace Real Time Analysis Function (Own Copyright) 58
Figure 36 Face Recognition Speed Comparison (Copyright Sefik Ilkin Serengil) 59
Figure 37 ITAINNOVA Computer Technical Specifications (Own Copyright) 59
Figure 38 CUDA, CuDNN, Tensorflow Compatibility Matrix (Copyright Tensorflow Documentation) 60
Figure 39 Graph Modelling Draft 1 (Own Copyright) 61
Figure 40 Graph Modelling Draft 2 (Own Copyright) 61
Figure 41 Final Graph Model Sample (Own Copyright) 62
Figure 42 Web-rtc SDP Graphical Explanation (Copyright Yuichiro Tachibana (Tsuchiya)) 64
Figure 43 Web-rtc Signaling Graphical Explanation (Copyright Yuichiro Tachibana (Tsuchiya)) 64
Figure 44 Rasa Project Directory (Own Copyright) 65
Figure 45 Rasa Action Server Directory (Own Copyright) 67
Figure 46 Rasa & Neo4j Calls from Pilar Web-Service (Own Copyright) 68
Figure 47 DeepFace Real-Time Haarcascade Detector (Own Copyright) 69
Figure 48 Multiple Face Detection Proof (Own Copyright) 69
Figure 49 Memory Outage Proof (Own Copyright) 70
Figure 50 Emotion Detection Under Unpleasant Circumstances (Own Copyright) 70
Figure 51 Streamlit Real-Time Stream Sample (Own Copyright) 71
Figure 52 Streamlit Video Playback Sample (Own Copyright) 72
Figure 53 Streamlit REST Rasa Chatbot Channel Sample (Own Copyright) 72
Figure 54 Pilar Avatar Unity Sample (Own Copyright) 72
Figure 55 Telegram REST Rasa Chatbot Channel Sample (Own Copyright) 73
Figure 56 People Relationship Graph Inference (Copyright Sefik Ilkin Serengil) 74
Figure 57 Dialogflow Streamlit Component (Own Copyright) 74

1. Introduction

The history of computing has been wonderful from the point of view of a computer enthusiast: who doesn't remember the first personal computers, square boxes with no graphical interface and a considerable weight? In such a short period of time, hardware has evolved so much... and this evolution has been accompanied in recent years by software. Humans are now capable of building machines that not so many years ago we would have described as science fiction, and no less can be said for what they are capable of doing with the software and infrastructure associated with them.

However, one of the major concerns since the beginning of computing has been that it should be usable not only by experts but also by the general public. In other words, to make technology part of human life, to be able to equate it with another organ or part of our body, such as the brain or a hand. This is a shocking concept given the reluctance to "machines surpassing humans", but that is a moral debate beyond the scope of this paper. What is intended is that we make technology our own, using it as an assistant or complement to further empower what humans can do.

Crucial to this task is that technology becomes more human and therefore more convenient to use, rather than the other way around (that humans do not have to adapt to the way computers think). This is the basis of human-machine interaction, the branch of computer science in charge, among many other things, of designing usable interfaces (for example, increasing the information that a human being is able to capture by looking at a given screen), improving and researching new ways of interacting with hardware and software (from voice to gestures) or optimising existing ones (such as touching a screen). One of the strategies pursued has been to endow human-machine interaction with sentiment models. This last sentence will be one of the pillars on which this project will be based, so it is convenient to introduce the concept from which it comes:

> Affective Computing [1] is a large field of research related to psychology. Together with AI based on Deep Learning, a machine can be able to detect and process people's emotions and adjust its own functioning as a result of this analysis. Among its applications are the improvement of the aforementioned assistants, treatment therapies for people with difficulties such as Down's Syndrome or being able to conduct a job interview without any human action, among others.

The other mainstay of this work is virtual assistants. You have probably heard about Siri or Alexa. This concept is basically a humanoid inside a screen to aid humans with their daily tasks. In particular, Pilar was developed by ITAINNOVA (Instituto Tecnológico de Aragón) and Imascono. My work will consist of providing Pilar with a model of emotions being able to respond to stimuli and storing/viewing her experiences in a knowledge graph.

2. Background & State of Art

First and foremost, to have a better understanding of the background and scope of the project, it is worth mentioning that it has been developed with the help of ITAINNOVA (Instituto Tecnológico de Aragón). The initial idea of the project came by another student´s work on his Final Degree Project, Martín Cativiela (further referred to as Martín), which will be explained in further detail later.

One of the fields of investigation of ITAINNOVA is Virtual Assistants (Chatbots) and Task Automation through IA. The use of these technologies allows companies to effectively automate repetitive, high-volume tasks, optimising processes and working more intelligently, increasing employee productivity and customer satisfaction. Some examples come from the generation of templates and mail automation to the use of chatbots to assist customer service.

Features:

- Generate fast and reliable information.
- 24h availability; reachable at any time from any place.

Use case:

The idea behind a conversational assistant is to be able to talk to another person as if it were a real person, simulating all of the features that make a conversation between people natural; providing automatic and coherent answers to the questions it receives, whether written or spoken.

Chatbots rely mainly on AI and its subfield Natural Language Processing (NLP) to be able to recognise the meaning of words, adapt to the conversation´s context, detecting intent and obtaining responses adequate to the specific needs of the other speaker.

Sectors:

The application of this type of solutions is oriented to multiple industries and sectors. Some of the most typical ones are customer service and webpage assistants yet in the last few years very innovative applications have come into place such as down syndrome therapy or job interviews assessment.

Inside Virtual Assistants (Chatbots) and Task Automation through AI field we can find Pilar. It is a huge, very ambitious I+D project merging ITA and Imascono knowledge and work used as a proof of concept of what can actually be done in this field, being currently used in a list of projects. Pilar consists of two modules: a 3D avatar and chatbot (voice and text). Each module has in addition other submodules yet for the purpose of this project we will only focus on the concept of Pilar herself (or itself?), the 3D avatar graphical interface and a conversational chatbot (text).

Pilar has been to many technology fairs showcasing what she can do. She is in a vertical screen with a built-in chat or webcam and microphone from which people can directly interact with her. A funny incident was that when returning from a fair, Pilar had learned to swear and spoke with a rude language (mainly because users had taught them to do so). She had to be "re-taught" so that in the next demo people wouldn't get scared.

An idea to continue improving Pilar is to provide her with a sentiment model so she could feel, process and respond to emotions as a human being would do, taking a step further in closing the gap between humans and machines. To do so, we will have a look at the different approaches, technologies, possibilities and state of the art offered by the time the initial idea arose and the project started its planning and early development stages.

In 2019, Martín started his Final Degree Project with this proposal. His initial idea was to examine two conversational characters emotions in a TV show scene, analysing how different events, acts, objects affect their short-term emotions and how this affect as well their mid-term mood ending in changing its long-term personality [2] using ALMA (A Layered Model of Affect) [3] and Adding the Emotional Dimension to Scripting Character Dialogues [4]. Martín spent some time explaining his project idea, scope and technical details which were the starting point.

Once introduced Martín´s previous work, my tutor Rafael del Hoyo (further referred to as Rafa) and I rethought the scope of the project. As my interest computer science fields are Machine Learning (ML), Artificial Intelligence (AI) and Computer Vision, it could be handy instead of focusing on the sentiment analysis part, centring the work into these three fields. Rafa came up with an interesting vision of how we could improve Pilar with a view to the future to leave a path for further scaling up the asset.

This consists of a pipeline which in Analysis, Design & Development will be more technically detailed. It all starts with a real-time video stream (hypothetically Pilar´s eyes). The frames are processed by a face-attribute detection module which extract the following features: person labels, person gender and age and person emotions (Deepface). Next, these features are the input of a sentiment model (ALMA) which makes calculations to determine Pilar´s mood and dominant emotion. The resulting Event is stored in a database (Neo4j) which is thought to be Pilar´s brain. Finally, users can interact with Pilar talking to her via chat (Telegram Bot) and maybe asking how is she feeling or what is she seeing in a current moment or by a dashboard (Streamlit) with which they can track her parameter values through time. A simple schema summarizing the data flow is provided below:

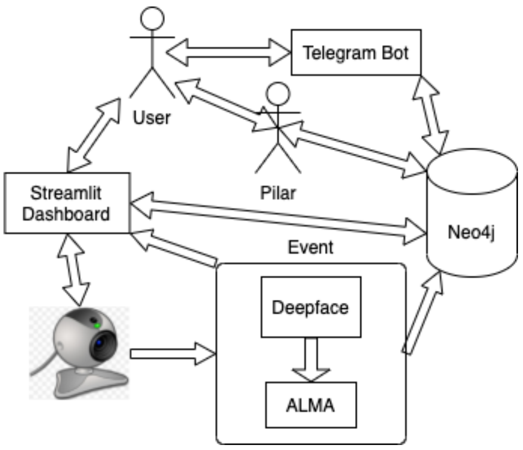


Figure 1 General Project Schema (Own Copyright)

With this schema in mind, Martín´s work had to be adapted to the new requirements. A research task was done to see the actual state of art of different technologies, support and understand the decisions taken of which ones are more convenient for this project and document everything. To summarize what it is needed, let's follow the data flow in the schema previously shown.

The main idea is to empower Pilar to understand the outside world. Sight, hearing, smell, and spatial awareness are the main senses humans and living beings have to do so. Translating this to the computer science discipline, there exist only noticeable advances and easily accessible open-source projects in the sight (vision), hearing and spatial awareness fields which do not depend on expensive and complex equipment (aside a camera, microphone and a LiDAR or 3D camera respectively). This was thought to be the initial scope of the project, yet the 3D camera option was discarded as a matter of low-budget project idea and that the scope would be more extensive than desired.

How all the inputs that come from the face attribute recognition module (and, if done, the object recognition module) could be processed? A sentiment appraisal and processing model is needed. As explained before, Martín chose ALMA [3]: It is a computational model for the real-time simulation of the affect that human beings can experience. Based on a kind of cognitive appraisal, different affect types are simulated in a hierarchical generation process. Hence, the name ALMA, which stands for A Layered Model of Affect.

Behaviour from human beings, especially interpersonal communication behaviour, is essentially influenced by affect. Simulated affect can be exploited for virtual characters used in human computer interfaces to make them more believable and achieving a more natural behaviour.

Psychological theories about affect were investigated for an employment in a computational model. As a result, a model representing emotions, moods, personality and their particular relations with different intensity decay functions, being able to realize the interference of different affect types was created. ALMA supports 24 emotion types, 8 mood types, and 5 personality types that cover short, medium and long term affect.

Could another sentiment processing model have been considered instead of ALMA? Indeed yes, and as of Summer 2020 when the research & documentation part was done, pretty complex and fine-tuned models existed available in open-source repositories. Contrary, this complexity was at the same time an advantage and a drawback as so many parameters had to be taken into account. After trying out performing some tests with other models and considering that Martín had made a simple approach of the ALMA´s Java code in Python, I determined that it was the best choice to continue using ALMA.

Following the schema´s data flow, the next step in the pipeline is to ask: How is all this information going to be stored? The first approach that first comes to mind is a non-relational database. But which of them? Because this is a project made for further development, graphs were thought to be the best option as they resemble a human brain full of neurons. "The world has changed. Knowing is half the battle" [5]. Knowledge graphs allow the use of connections to infer new knowledge. This enables exploration, discovery and decision-making by human, software, AI systems or Pilar. Some use cases are baseline knowledge (aggregate and search large volumes of static internal data), new discoveries and insights (using AI inference for learning new connections or content) and decisioning based on contextual relevance (detangle complex processes or events by analyzing all relationships and establishing new parameters or relations). All of this could be applicable to Pilar.

For querying the non-relational graph database an attractive, easy to interact with solution was wanted. Thereby simply chatting with Pilar was somehow an interesting, curious solution. Since the creation of the first Verbot in 1994, chatbots have undergone an underlying transformation. There exist today hundreds of applications in the vast majority of sectors such as healthcare, politics, education, and a large etcetera. In addition, and as mentioned before, the ITA uses chatbots for a variety of innovative use cases. Even Pilar is a chatbot herself! But to be more concise, it is required a conversational (non-speech, only text) messaging-app chatbot. Doing so in Whatsapp, which is in Spain the most widely used messaging service is a little bit complicated mainly involving payment options. However, Telegram, which is used by a smaller, yet not insignificant community provides an open-source thought-for solution for our scenario. In addition, to provide the bot with some intelligence, ITA has developed multiple times other chatbot-applications with Rasa, an open source chatbot technology made with AI that can understand the conversation context, different ways of requesting information, several intentions throughout the conversation… making it similar to talking with a real person.

Last, but not least, dashboards are in vogue among the data scientists community. They represent a visual, straightforward way to understand complex data at a glance. If configurable, they can also be a good-looking front end for DS applications. Let's cover each of the parts mentioned in this introduction in a more broadly.

2.1. Computer Vision State of the Art

Ideally, performing sentiment analysis fusing audio and video streams is what would provide the best accuracy and desired scenario for this project. However, embedding this into a single system is very difficult and there are only a few, if any, project that meet this feature. I found and tested some projects in which fine-tuned guidelines to make the training yourself were provided with its correspondent code, but I did not dispose the adequate (graphical powerful) equipment to do so and furthermore success of doing such a time-requiring task was not guaranteed. Almost in the end of my investigation task, I found a paper [6] which dramatically got my hopes up, but no code was provided alongside it. Despite this major constraint, it is worth explaining it a little bit as it provides with a very straightforward vision of how this audio-video stream fusion should be handled.

Emotion Recognition in the Wild Challenge (EmotiW) is a challenge celebrated yearly in diverse parts of the world. This paper corresponds to one of the winners of the Audio-Video Sub-Challenge in EmotiW2018 validated on the AFEW database (which is collected from films and TV series to simulate the real world) achieving an accuracy of 62.48%, outperforming the state-of-the-art result. With Automatic Emotion Recognition (AER), human emotional states can be apperceived by auditory and visual systems. Intuitively, not all the frames in a video contain emotion information due to the sparse expression of emotion. To adapt the weight of each frame to the final classifying, according to its importance to emotion, they propose an attention mechanism to detect the emotion-dependent frames in the face image systems. A similar strategy can be adopted in audio to detect the emotion-relevant timestamps and then the separate salient features are fused in a Factorized Bilinear Pooling (FBP) block to deeply capture the association between modalities for final emotion recognition.

Once explained the desired system, a workaround was needed to obtain a similar behaviour. I investigated vision and audio sentiment analysis systems separately and found that the audio side is very difficult and there are only a few projects that met my necessities. I found and tested some projects in which fine-tuned schemas to make the training yourself were provided with its correspondent code (see Summer – October 2020), but I did not dispose the adequate (graphical powerful) equipment to do so and furthermore success of doing such a time-requiring task was not guaranteed. Despite this, I managed to get some culture of how emotion recognition in audio is possible (a very brief summary): depending on the actual meaning of your words, the pitch, tone, speed and other parameters, a statistical model with weights in a deep neural network makes this task possible.

After explaining the different paths I followed, a facial attribute recognition system was needed (I was only left with that possibility if I didn´t want to extend a lot the scope of the project). Google defined the state-of-the-art technology in 2015 with a paper titled FaceNet [7]. It was the starting point of a very extensive future work focused on better understanding the error cases, further improving the model, and reducing model size and CPU requirements. It included the following tasks: face verification (is this the same person), recognition (who is this person) and clustering (find common people among these faces). Another open-source project is Facebook 2014 paper title Deepface [8]. It achieved a lower accuracy (97 vs. almost 100 percent respectively) than Google´s in the LFW dataset and the approach Google´s researchers took went beyond simply verifying whether two faces are the same. Its system could also put a name to a face and even present collections of faces that look the most similar or the most distinct. In contrast, we are in 2021 and both projects have evolved significantly, being Facebook´s the final verdict for its so complete and very easy to use interface around AI tasks. This decision will be further explained in the dedicated 2.3. Emotion Detection (Deepface) as well as in the development part.

In addition to a facial attribute model, something that could come in handy to determine a person´s mood is what surrounds that person: objects or spaces. A custom AI model could be trained to determine known objects (e.g., a kinfe, a pencil, etc.) and determine how these will affect the person´s mood. Unfortunately, this option has been considered for a while in the project but finally could not make it (see Objectives).

2.2. Emotion Detection (DeepFace)

The chess winner algorithm is thought as the most extraordinary challenge of AI studies and classifying images comes after it. DeepFace is a lightweight face recognition and facial attribute analysis (age, gender, emotion and race) framework for python. It wraps state-of-the-art models: VGG-Face, Google FaceNet, OpenFace, Facebook DeepFace, DeepID, ArcFace and Dlib. The library is mainly based on Keras and TensorFlow. Modern face recognition pipeline consists of four common stages. These are detection, alignment, representation and verification. Deepface handles all these common stages in the background. You can just call its verification, find or analysis function in its interface with a single line of code.

- Face verification: this function under the DeepFace interface offers to verify face pairs as same person or different persons.
- Face recognition: requires applying face verification several times.
- Facial attribute analysis: including age, gender, facial expression (including angry, fear, neutral, sad, disgust, happy and surprise) and race (including asian, white, middle eastern, indian, latino and black) predictions.

2.2.1. Face Recognition pipeline

Face recognition models are regular convolutional neural networks (CNN) and they are responsible to represent faces as vector embeddings. Verification task is based on if a comparison of a pair is less than a threshold utilizing different metrics such as Cosine Similarity (with their angle), Euclidean Distance and L2 form (with their distance). Additionally, altitude normalization can be applied to the vectors before comparing them.

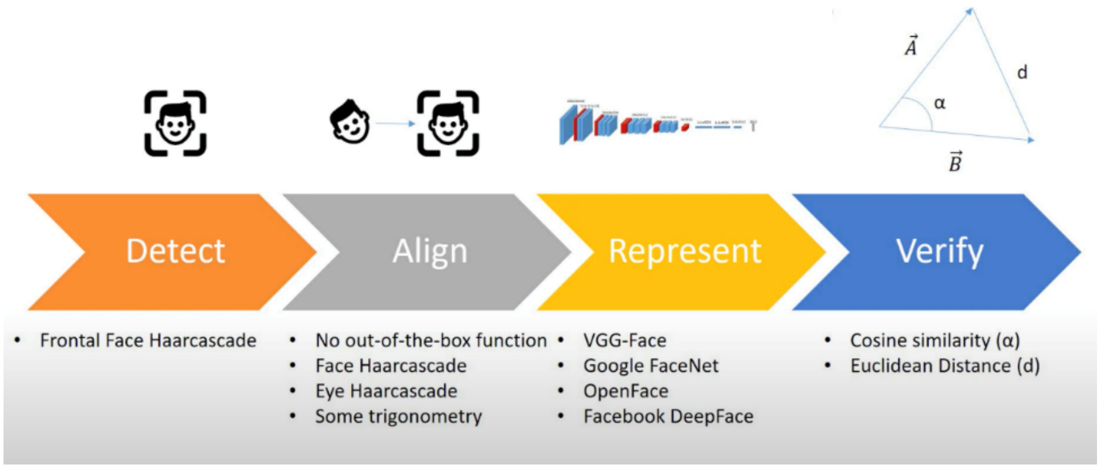
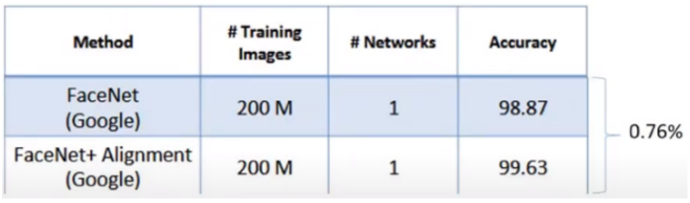


Figure 2 Modern Face Recognition Pipeline Stages (Copyright Sefik Ilkin Serengil)

Face detection can be done with many solutions such as OpenCV, Dlib or MTCNN. OpenCV offers haar cascade, single shot multibox detector (SSD). Dlib offers Histogram of Oriented Gradients (HOG) and Max-Margin Object Detection (MMOD). Finally, MTCNN is a popular solution in the open source community as well. Herein, SSD, MMOD and MTCNN are modern deep learning based approaches whereas haar cascade and HoG are legacy methods. Besides, SSD is the fastest one.

For the alignment step there is not out-of-the-box function in OpenCV to align faces; we can use face or eye haarcascade models and some trigonometry. Google declared that face alignment increases its face recognition model FaceNet almost 1.



Method	# Training Images	# Networks	Accuracy
FaceNet (Google)	200 M	1	98.87
FaceNet+ Alignment (Google)	200 M	1	99.63

0.76%

Figure 3 Face Recognition Alignment Accuracy Improvement (Copyright Facenet paper)

2.2.2. Which face recognition model is the best?

Let´s discuss which face recognition model is the best among the state of the art. Face Recognition research has emerged from the rulers of the tech giants from the top universities in the world. Firstly, Facebook announced its face recognition model Deepface in 2014. After that, in 2015, Google promoted its face recognition model FaceNet. In the same year, University of Oxford Visual Geometry Group introduced VGG-Face model and finally, researchers from the Carnegie Mellon University released its Open Face model. Herein, researchers shared their network architectures publicly. However, from academic perspective, researchers preferred to share pre-trained weights yet from the commercial, they did not want to offer this for free. Thankfully, open-source community (David Sangberg, Swarup Ghosh, among others) pretrained the models and offered them to the public.

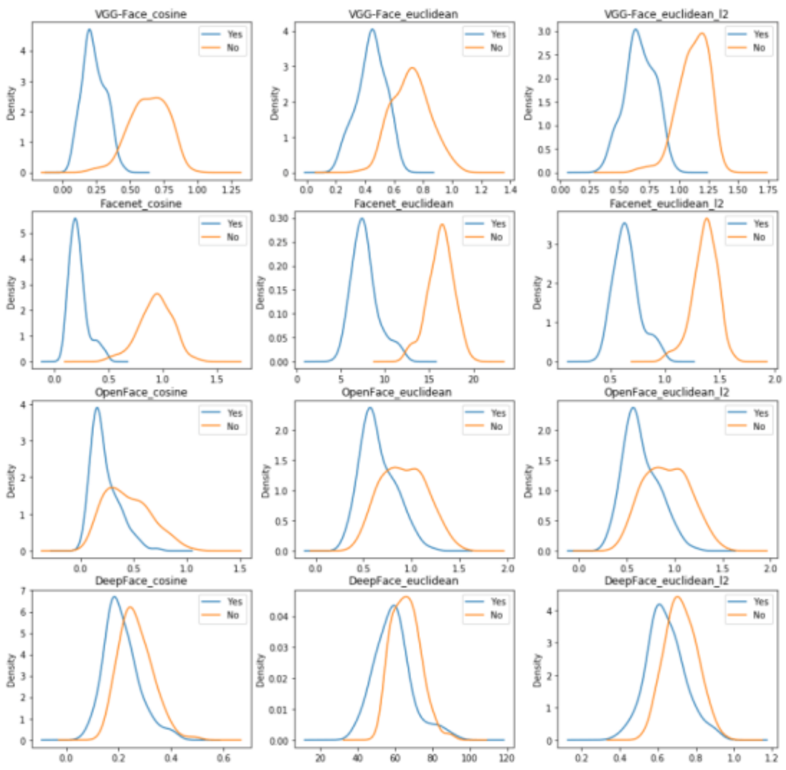
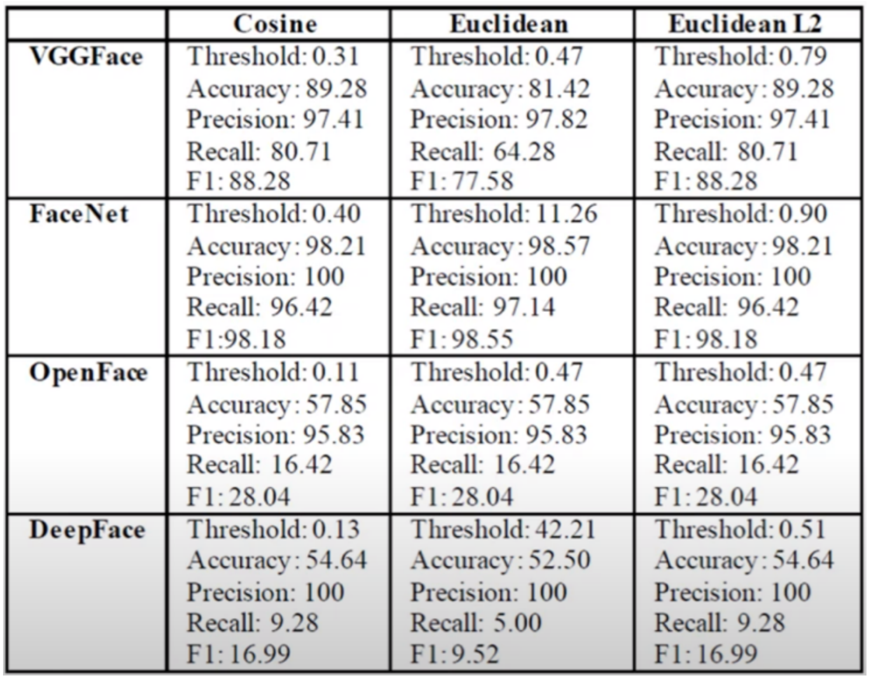


Figure 4 Face Recognition Models & Distance Metrics Comparison (Copyright Sefik Ilkin Serengil)

Distance distribution of positive and negative face pairs is illustrated in this graph: rows represent distance matrix whereas columns state face recognition models. The clearer discrete positive and negative classes are, the better the model is. It seems Google FaceNet offers the most robust model among others besides Euclidean L2 seems more stable than Cosine and regular Euclidean distance.

Following the face verification pipeline, if we feed the individual distance values and the target labels to a decision tree, the algorithm will find the best split point which maximizes the information gain (parameter finetuning). Then, we can verify two faces if the distance is less than the split point found by the decision tree algorithm. This table shows the best split point for each model metric combination. For example, we will verify two faces are the same if the distance is less than 0.31 in VGG-Face and cosine pair. Based on these points, it is possible to calculate the accuracy metrics including precision, recall and F1 score. No matter which distance metric is applied, FaceNet face recognition model is the most accurate one in the list and VGG-Face comes after. Interestingly, FaceNet works better with Euclidean distance whereas VGG-Face works better with both Cosine. Notice that Facebook researchers declared that human beings have 97.53% score for face recognition tasks. FaceNet passed that record as well.



	Cosine	Euclidean	Euclidean L2
VGGFace	Threshold: 0.31 Accuracy: 89.28 Precision: 97.41 Recall: 80.71 F1: 88.28	Threshold: 0.47 Accuracy: 81.42 Precision: 97.82 Recall: 64.28 F1: 77.58	Threshold: 0.79 Accuracy: 89.28 Precision: 97.41 Recall: 80.71 F1: 88.28
FaceNet	Threshold: 0.40 Accuracy: 98.21 Precision: 100 Recall: 96.42 F1:98.18	Threshold: 11.26 Accuracy: 98.57 Precision: 100 Recall: 97.14 F1: 98.55	Threshold: 0.90 Accuracy: 98.21 Precision: 100 Recall: 96.42 F1: 98.18
OpenFace	Threshold: 0.11 Accuracy: 57.85 Precision: 95.83 Recall: 16.42 F1: 28.04	Threshold: 0.47 Accuracy: 57.85 Precision: 95.83 Recall: 16.42 F1: 28.04	Threshold: 0.47 Accuracy: 57.85 Precision: 95.83 Recall: 16.42 F1: 28.04
DeepFace	Threshold: 0.13 Accuracy: 54.64 Precision: 100 Recall: 9.28 F1: 16.99	Threshold: 42.21 Accuracy: 52.50 Precision: 100 Recall: 5.00 F1:9.52	Threshold: 0.51 Accuracy: 54.64 Precision: 100 Recall: 9.28 F1: 16.99

Figure 5 Face Recognition Models Testing Metrics Comparison (Copyright Sefik Ilkin Serengil)

- <u>Models for image recognition</u>: face recognition is a combination of CNN, Autoencoders and Transfer Learning studies. Previously, traditional computer vision algorithms were applied to recognize images. The illustration below shows the ImageNet results throughout time. ImageNet consists of 1.2M images of 1000 different categories.

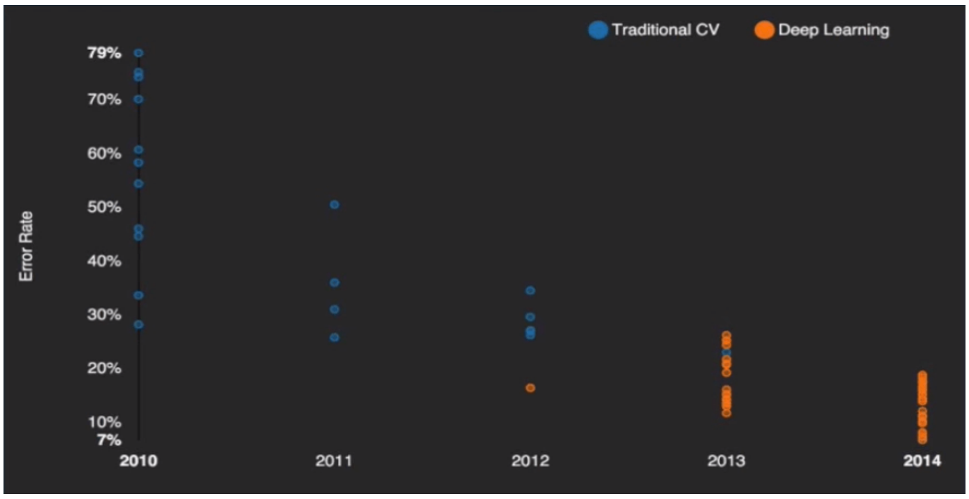


Figure 6 Computer Vision Models Error Rate Throughout Time (Copyright Sefik Ilkin Serengil)

It got stuck in almost 30% error rate with traditional computer vision. Applying deep learning to this field changes the course of history (orange node appearing in 2012 states AlexNet), error rates jumped to 15% in an instant, and today we've gone a long way further: Inception V3 model produced almost 3% error rate in 2014. What´s more, transferring learning has also been a huge milestone in what democratise technology (being able to leverage very well trained models with very powerful GPUs) refers. To continue, the models available in the DeepFace framework will be described:

- **OpenFace** [9]: developed by researchers of Carnegie Mellon University, it is not the best, but a strong alternative to stronger ones such as VGG-Face or FaceNet, to which it resembles. The model is built on Inception Resnet V1 and applies one shot learning. It has 3.7M trainable parameters, whereas 145M in VGG-Face and 22.7M in Facenet. Besides, it is very lightweight with only 14MB, whereas VGG-Face is 566 MB and Facenet is 90 MB. The speed factor is why the adoption of OpenFace is very high, you can deploy it even in a mobile device.
- **DeepFace** [8]: created by a research group at Facebook back in 2014, it identifies human faces with near human-level performance (97.35% in LFW dataset). In the alignment and representation steps Deepface employs a 3D face modelling to apply a piecewise affine transformation (making this previously didn´t provide decent results; face-patterns supervised learning was used instead). Other interesting features are that there is not weight sharing inside the deep network rather than the standard convolutional layers and that the two last layers F7 (Representation) and F8 (SFC labels) are fully connected without losing any performance or having to do excessive adjustments later. All the features mentioned above make the deep network scale well to large (larger than all of the previous work in this field of AI) datasets.

However, Deepface model falls behind Google FaceNet which got 99.65% on the LFW dataset. It also leaves behind the FBI's Next Generation Identification system which have 85% performance. One of the creators of the software, Yaniv Taigman, came to Facebook via their 2007 acquisition of Face.com. Facebook started rolling out the technology to its users in early 2015, with the exception of users in the EU due to data privacy laws there. The Huffpost called the technology “creepy” [10] concerning data privacy, noting that some European governments had already required Facebook to delete facial-recognition data. Back in 2014, Facebook had not released any press announcements concerning DeepFace, although their research paper had been published earlier in the month. This lack of publicity was said to be intentional to avoid another round of 'creepy' headlines.

- **VGG-Face** [11]: Oxford visual geometry group announced its deep face recognition architecture in 2015. This research group is familiar with Kaggle ImageNet competitions. The NN is deeper than Facebook’s DeepFace.
- **FaceNet** [7]: Google announced FaceNet in 2015. It was built based on the Inception model, which is often used in Kaggle challenges. As mentioned before, VGG-Face produces more successful results than FaceNet yet as a price of running slower in real time.
- **DeepID** [12]: China involved in the face recognition competition with its prestigious academic institution as well. Researchers of the Chinese University of Hong Kong announced two different version of DeepID model for face recognition tasks in 2014.
- **ArcFace** [13]: developed by the researchers of Imperial College London, it is a module of InsightFace face analysis toolbox. It got 99.83% accuracy score on LFW data set and is based on ResNet34 model.
- **Dlib**: it is a powerful library having a wide adoption in image processing community similar to OpenCV. Researchers mostly use its face detection and alignment module. Beyond this, Dlib offers a strong out-of-the-box face recognition module as well. Even though it is written in C++, it has a python interface as well. Dlib is mainly inspired from a ResNet-34 model (shown in the ArcFace model). It got 99.38% accuracy in the LFW dataset. Face detection does not have to be applied for rectangle areas. We can do it more sensitive with the facial landmark detection with Dlib. It can find 68 facial landmark points on the face including jaw and chin, eyes and eyebrows, inner and outer area of lips and nose.

- **Ensemble**: Even though all those models perform well, there is no absolute better model. Still, we can apply an ensemble method to build a grandmaster model. In this approach, we will feed the predictions of those models to a boosting model. Accuracy metrics including precision, recall and F1 score increase dramatically in ensemble method whereas running time lasts longer.

- Distance metrics [14]:

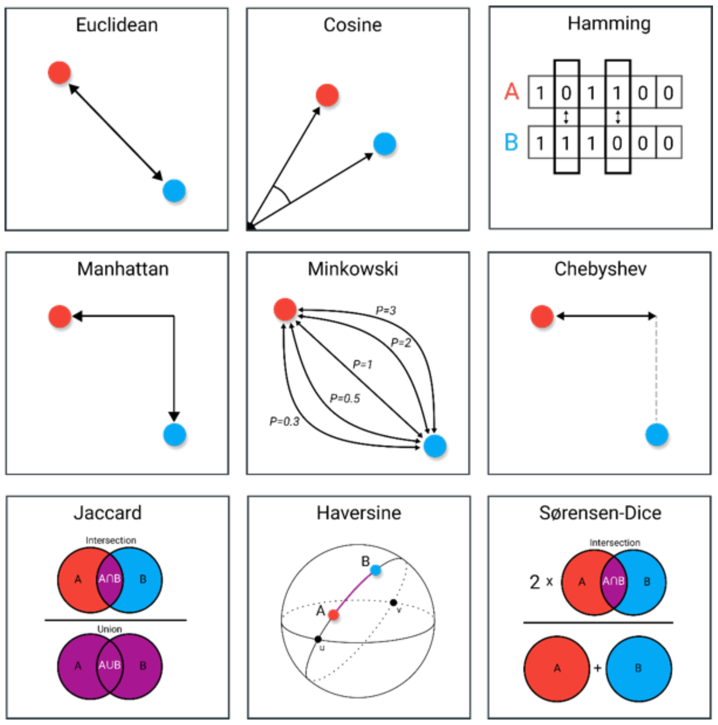


Figure 6 Graphical Representation of different Distance Metrics (Copyright [14])

- Cosine similarity: its formula can be derived from the equation of dot products:
  - $D(x,y) = \cos\theta = \frac{x \cdot y}{|x||y|}$
    - $\cos 0' = 1$ means similarity (same direction).
    - $\cos 90' = 0$ means some similarity (orthogonal).
    - $\cos 180' = -1$ means no similarity (opposite direction).

The magnitude is not of importance as this is a measure of orientation, merely their direction. In practice, this means that the differences in values are not fully taken into account. If you take a recommender system, for example, then the cosine similarity does not take into account the difference in rating scale between different users.

- Euclidean: Euclidean distance is the straight line distance between 2 data points in a plane. It is calculated using the Minkowski Distance formula by setting p value to 2, thus, also known as the L2 norm distance metric. The formula is:
  - $d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_1)^2}$

The distance is calculated from the cartesian coordinates of the points using the Pythagorean theorem. Euclidean distance is not scale in-variant which means that distances computed might be skewed depending on the units of the features. Typically, one needs to normalize the data before using this distance measure.

- Finetuning parameters [15]: in the face recognition task, positive classes (two faces are the same) seem to have symmetrical distribution of their mean whereas negative ones (two faces are different) have negative skew. We could separate these two classes with a threshold value.

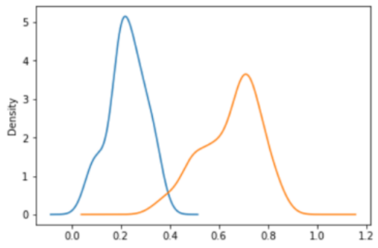


Figure 7 Mean Distribution of Positives and Negatives (Copyright [15])

  - Statistical approach: with 2 standard deviation we can classify pairs as being the same person if their distance is less than 0.3751.
  - Decision trees: as an alternative to set the threshold to 2 sigma, decision tree algorithms split the data set maximixing information gain and resulting in a better method. Chefboost framework is lightweight and allows to read decision trees as if statements: a simple decision stump can be created with the C4.5 algorithm. Now the threshold is 0.3147.

  To conclude, decision tree approach is good at precision whereas 2 sigma method is good at recall. If the target is a security-first application, then precision is more important because it is more confidential when confirming that two people are the same person.

2.2.3. Facial Attribute Analysis

Selfik Ilkin Serengil has made a huge work in adapting other people´s work into his framework for instance by training some models by himself or translating already made work into an easy to use python package. What´s more, with his webpage articles explained that work he has accomplished to make a huge abstraction layer for more unexperienced people who are seeking to learn.

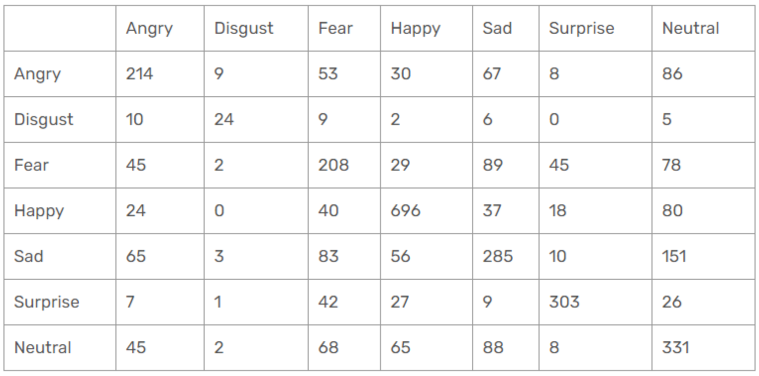
2.2.3.1. Facial Expression Recognition (Emotion Analysis)

Kaggle announced facial expression recognition challenge in 2013. Researchers are expected to create models to detect 7 different emotions from human being faces. However, recent studies are far away from the excellent results even today.

Sefik Ilkin Serengil has provided a pretrained model with that Kaggle competition dataset, which consists of image pixels (48×48=2304 values), emotion of each image and usage type (as train or test instance). He applies CNN with Keras using TensorFlow backend, tests and tunes the training until he achieves 57% accuracy on test set. That can be acceptable because winner of Kaggle challenge has got 34% accuracy. He is also able to improve the accuracy from 57% to

66% with Auto-Keras for the same task. This is a classification problem and pure accuracy couldn't give an idea to evaluate the system. Thereby we have to look at the confusion matrix [16]:

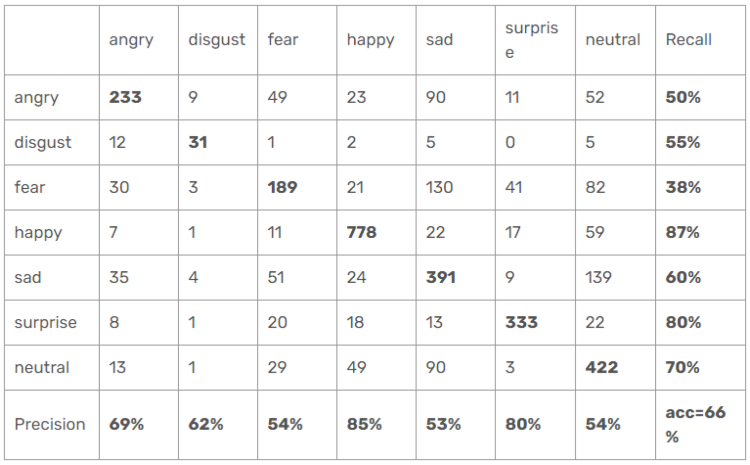
- Custom design:



	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	214	9	53	30	67	8	86
Disgust	10	24	9	2	6	0	5
Fear	45	2	208	29	89	45	78
Happy	24	0	40	696	37	18	80
Sad	65	3	83	56	285	10	151
Surprise	7	1	42	27	9	303	26
Neutral	45	2	68	65	88	8	331

Figure 8 Keras Facial Expression Recognition Confussion Matrix (Copyright [16])

- Auto-Keras design:



	angry	disgust	fear	happy	sad	surpris e	neutral	Recall
angry	**233**	9	49	23	90	11	52	**50%**
disgust	12	**31**	1	2	5	0	5	**55%**
fear	30	3	**189**	21	130	41	82	**38%**
happy	7	1	11	**778**	22	17	59	**87%**
sad	35	4	51	24	**391**	9	139	**60%**
surprise	8	1	20	18	13	**333**	22	**80%**
neutral	13	1	29	49	90	3	**422**	**70%**
Precision	**69%**	**62%**	**54%**	**85%**	**53%**	**80%**	**54%**	**acc=66 %**

Figure 9 Auto-Keras Facial Expression Recognition Confussion Matrix (Copyright [16])

2.2.3.2. Age and Gender Prediction

Computer vision researchers of ETH Zurich University (Switzerland) announced a very successful apparent age and gender prediction models in 2015 with the IMDB-WIKI dataset, winner of the ChaLearn Looking at People (LAP) challenge on Apparent age V1 (ICCV '15). Their choice was VGG with ImageNet and they shared the model structure and pre-trained weights of the production model for Caffe (Convolutional Architecture for Fast Feature Embedding) framework, leaving the door open to applying transfer learning. There are 101 classes in the output layer for ages from 0 to 100. Researchers converted a classification task into regression: multiplying each SoftMax outcome with its label gives the apparent age prediction.

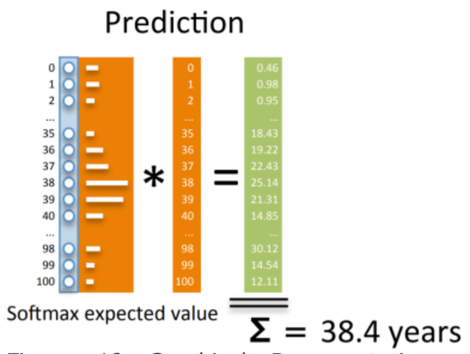


Figure 10 Graphical Representation of Regresion for Age and Gender Prediction (Copyright Sefik Ilkin Serengil)

2.2.3.3. Race and Ethnicity Prediction

Recognizing ethnicity from face photos could make a huge contribution to missing children, search investigations, refugee crisis, genealogy research and a field of study in AI Ethics. There already exist in the market some commercial solutions like Face++, Clarifai, Kairos or a huge list of Computer Vision Startups. Basically two main public data sets including ethnicity labelled face pictures are available: FairFace and UTKFace. Sefik Ilkin Serengil has made his own model using VGG-Face for transfer learning with the following confussion matrix (accuracy does not mean anything, only precision and recall):

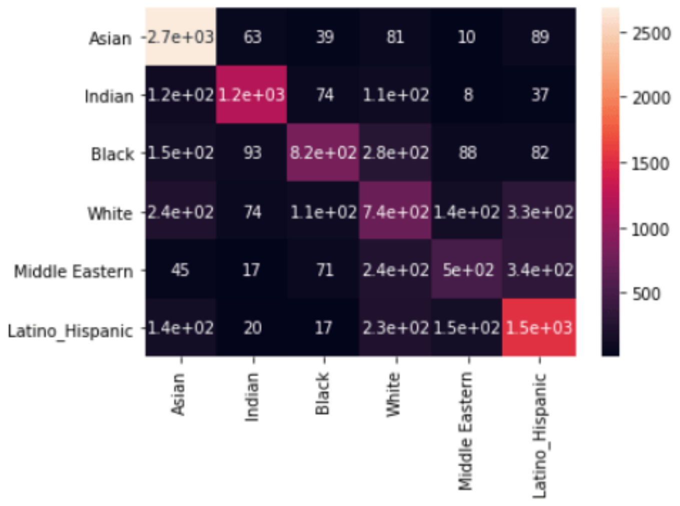


Figure 11 Confussion Matrix for Race and Ethnicity Prediction Model (Copyright Sefik Ilkin Serengil)

2.3. Emotion Model (ALMA)

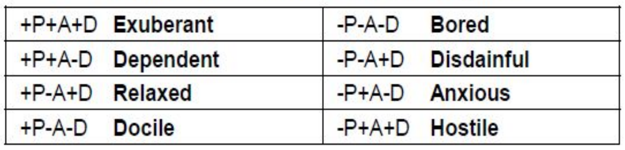
The idea behind ALMA is to integrate the three major affective characteristics: emotions, mood and personality that cover short, medium and long-term affect into a unique model to evaluate and compute input appraisal rules. These as well can be processed by sub-sequent modules that control the cognitive processes and physical behaviour of embodied conversational characters, enriching their lifelike and believable qualities.

There exists a variety of emotional characters based on the so-called OCC model developed by Ortony, Clore and Collins [17], although the approaches differ in the granularity of modelling, the mathematical machinery for computing emotions and in the way of how the model has been implemented on a technical level. It has to be defined how each kind of affect will be computed and how they interact with each other:

- Emotions are caused by a specific event, action or subject. When elicited, they usually decay and disappear over time.
- Moods are longer lasting stable affective states, which have a great influence on human´s cognitive functions. Mehrabian describes them as "an average of a person's emotional states across a representative variety of life situations".

- Personality reflects individual differences in mental characteristics. A common schema is the Big Five model: openness, conscientiousness, extraversion, agreeableness and neuroticism.

OCC is essentially based on the concepts of appraisal and intensity. The individual is said to make a cognitive appraisal of the current state of the world. Mehrabian describes mood with the three traits pleasure (P), arousal (A) and dominance (D). The three traits are nearly independent and form a three-dimensional mood space. The implementation of the PAD mood space uses axes ranges from -1.0 to 1.0 for each dimension. A mood octant stands for a discrete description of a mood. Strength of a current mood is defined by its distance to the zero point of the PAD mood space (algebraically: the norm of the PAD vector). The longest distance in a mood octant is $\sqrt{3}$. To use no numbers for describing strength, the longest distance is divided into three parts and call them: slightly, moderate and fully. When starting a medium-term affect computation, it is essential to define an individual default mood for all characters, as a mood start value. Relationship between big five personality traits and the PAD space.



+P+A+D	**Exuberant**	-P-A-D	**Bored**
+P+A-D	**Dependent**	-P-A+D	**Disdainful**
+P-A+D	**Relaxed**	-P+A-D	**Anxious**
+P-A-D	**Docile**	-P+A+D	**Hostile**

Figure 12 Mood Octants of the PAD Space (Copyright ALMA Paper)

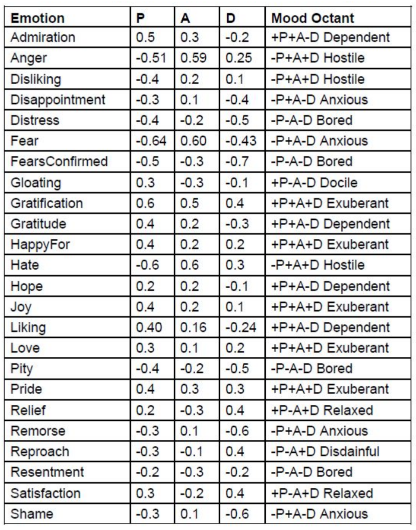
Pleasure := 0.21•Extraversion + 0.59•Agreeableness + 0.19•Neuroticism

Arousal := 0.15•Openness + 0.30•Agreeableness - 0.57•Neuroticism

Dominance := 0.25•Openness + 0.17•Conscientiousness + 0.60•Extraversion - 0.32•Agreeableness

Figure 13 Affect Space Features Calculation (Copyright ALMA Paper)

To leave the modelling of mood changes as lean as possible, emotions are taken as the mood changing factor. In order to realize this, emotions must be somehow related to a character´s mood. While using the PAD space for modelling mood, it is obvious to put emotions in relation to the PDA space too. There are mappings for emotions into the PAD (or comparable) 3-dimensional affect space, which can be used for this purpose. However, not for all 24 emotion types provided by the EmotionEngine exist a mapping into the PAD space. Those that lack a mapping are carefully provided with the missing pleasure, arousal and dominance values by exploiting similarities to comparable emotion types. We can then observe how the intensity of emotions influences the change of the current mood in addition to the aspect that a person´s mood gets the more intense the more experiences the person makes that are supporting this mood.



Emotion	P	A	D	Mood Octant
Admiration	0.5	0.3	-0.2	+P+A-D Dependent
Anger	-0.51	0.59	0.25	-P+A+D Hostile
Disliking	-0.4	0.2	0.1	-P+A+D Hostile
Disappointment	-0.3	0.1	-0.4	-P+A-D Anxious
Distress	-0.4	-0.2	-0.5	-P-A-D Bored
Fear	-0.64	0.60	-0.43	-P+A-D Anxious
FearsConfirmed	-0.5	-0.3	-0.7	-P-A-D Bored
Gloating	0.3	-0.3	-0.1	+P-A-D Docile
Gratification	0.6	0.5	0.4	+P+A+D Exuberant
Gratitude	0.4	0.2	-0.3	+P+A-D Dependent
HappyFor	0.4	0.2	0.2	+P+A+D Exuberant
Hate	-0.6	0.6	0.3	-P+A+D Hostile
Hope	0.2	0.2	-0.1	+P+A-D Dependent
Joy	0.4	0.2	0.1	+P+A+D Exuberant
Liking	0.40	0.16	-0.24	+P+A-D Dependent
Love	0.3	0.1	0.2	+P+A+D Exuberant
Pity	-0.4	-0.2	-0.5	-P-A-D Bored
Pride	0.4	0.3	0.3	+P+A+D Exuberant
Relief	0.2	-0.3	0.4	+P-A+D Relaxed
Remorse	-0.3	0.1	-0.6	-P+A-D Anxious
Reproach	-0.3	-0.1	0.4	-P-A+D Disdainful
Resentment	-0.2	-0.3	-0.2	-P-A-D Bored
Satisfaction	0.3	-0.2	0.4	+P-A+D Relaxed
Shame	-0.3	0.1	-0.6	-P+A-D Anxious

Figure 14 Mapping of OCC Emotions into PAD Space (Copyright ALMA Paper)

Affect is viewed as a result of cognitive appraisal. Each appraisal of an action, event or object, lets the EmotionEngine generate an active emotion that once generated, is decayed over a certain amount of time. All active emotions are used as input of the pull and push mood change function. It first computes the virtual emotion centre of all currently active emotions in the PAD space by using the above mapping. The virtual emotion centre represents a point in the PAD space and has an intensity that is the average of all active emotions' intensity. Its maximum intensity is 1.0. If no active emotions exist, no virtual emotion centre exists and the current mood is not influenced by active emotions.

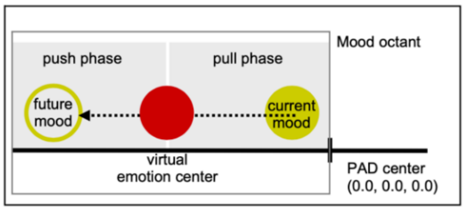


Figure 15 Pull and Push Mood Change Functions (Copyright ALMA Paper)

Usually the virtual emotion centre jumps from one mood octant to another one in very short time periods, caused by new elicited emotions. The intensity of the virtual emotion centre defines how strong the current mood is attracted respectively pushed away. For being flexible at mood changes, we define a usual mood change time. This defines the amount of time the pull and push mood change function needs to move a current mood from one mood octant centre to another one, presumably that a suitable virtual emotion centre exists that long, what is usually not the case. Our character's usual mood change time is 10 minutes.

With view to the future, the proposed affect generation by ALMA is a first approach for generating different temporal affects yet should be enhanced in several ways. For instance, an emotion´s intensity is not influenced by the current mood.

2.4. Events Storage in Graphs (Neo4j)

2.4.1. What, When, How, Why?

Graph databases are a type of NoSQL database, created to address the limitations of relational databases, intending to hold data without constricting it to a pre-defined model. They are designed to treat the relationships between data as equally important to the data itself [18]: while the graph model explicitly lays out the dependencies between nodes of data, the relational model and other NoSQL database models link the data by implicit connections. We live in a connected world. There are no isolated pieces of information, but rich, connected domains all around us. Only a database that natively embraces relationships is able to store, process, and query connections efficiently. Graph databases excel at managing highly connected data and complex queries.

Neo4j initial development began in 2003, but it has been publicly available since 2007. The source code, written in Java and Scala, is available for free on GitHub or as a user-friendly desktop application download. Neo4j has both a Community Edition and Enterprise Edition of the database. The Enterprise Edition includes all that Community Edition has to offer, plus extra enterprise requirements such as backups, clustering, and failover abilities. Neo4j is referred to as a native graph database because it implements the property graph model down to the storage level. This means that the data is stored exactly as someone would whiteboard it, and the database uses pointers to navigate and traverse the graph. It also provides full database characteristics, including ACID transaction compliance, cluster support, and runtime failover - making it suitable to use graphs for data in production scenarios. Some of the following particular features make Neo4j very popular among developers, architects, and DBAs:

- Cypher, a declarative query language similar to SQL, but optimized for graphs. Now used by other databases like SAP HANA Graph and Redis graph via the openCypher project.
- Constant time traversals in big graphs for both depth and breadth due to efficient representation of nodes and relationships. Enables scale-up to billions of nodes on moderate hardware.
- Flexible property graph schema that can adapt over time, making it possible to materialize and add new relationships later to shortcut and speed up the domain data when the business needs change.
- Drivers for popular programming languages, including Java, JavaScript, .NET, Python, and many more.

2.4.2. Modelling Data

Data modelling effort when using a Graph database follows a different paradigm than how you usually model the data stored in Relational or other NoSQL databases like Document databases, Key Value data stores, or Column Family databases. Graph data models can be used to create rich and highly connected data to represent the real world use cases and applications [19].

The graph data model is often referred to as being "whiteboard-friendly". Typically, when designing a data model, people draw example data on the whiteboard and connect it to other data drawn to show how different items connect. Instead of modifying the data model to fit a normalized table structure as in the ERD, the graph data model stays exactly as it was drawn on the whiteboard [20].

In the property graph model, data is organized as nodes, relationships, and properties (data stored on the nodes or relationships).

- Nodes are the entities in the graph. They can hold any number of attributes (key-value pairs) called properties. Nodes can be tagged with labels, representing their different roles in your domain. Node labels may also serve to attach metadata (such as index or constraint information) to certain nodes.
- Relationships provide directed, named, semantically-relevant connections between two node entities (e.g. Employee WORKS_FOR Company). A relationship always has a direction, a type, a start node, and an end node. Like nodes, relationships can also have properties.

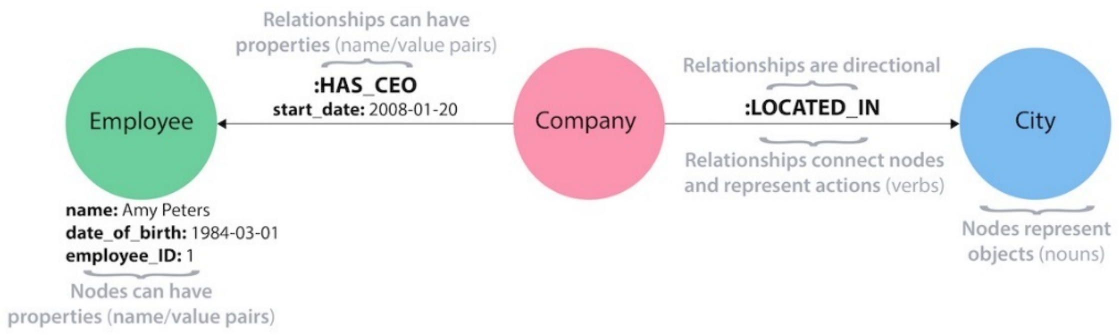

Relationships can have properties (name/value pairs)
:HAS_CEO
start_date: 2008-01-20
Employee
Company
Relationships are directional
:LOCATED_IN
City
Relationships connect nodes and represent actions (verbs)
name: Amy Peters
date_of_birth: 1984-03-01
employee_ID: 1
Nodes can have properties (name/value pairs)
Nodes represent objects (nouns)

Figure 16 Property Graph Model (Copyright [5])

2.4.2.1. Modelling Events in Time

A lot of approaches have been stated when it comes to modelling events in time. However, it is still a relatively unexplored field and different method can be mixed and matched. The initial constraint is that in a graph database the query syntax depends on whether those dimensions have been modelled as nodes, relationships, or properties of nodes or relationships [21].

```
MATCH (u:User)-[:VIEWS]->(p:Page)
RETURN u.id, p.url;
```

```
MATCH (e:Event)
RETURN e.user_id, e.page_url;
```

2.4.2.1.1. *The event-grammar approach*

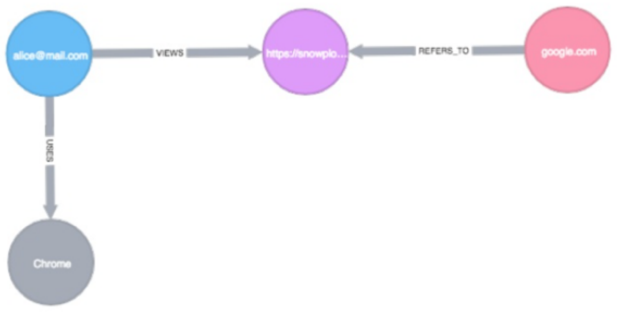


Figure 17 Event Grammar Graph Approach (Copyright [5])

In the event-grammar model, an event is a snapshot of a set of entities in time. This model is already a graph, with nodes representing the various entities and relationships between the nodes.

•This model makes it hard to find all events by the same user.

```
MATCH (u:User)-[r:VIEWS]->(p:Page)
WHERE u.email = 'alice@mail.com'
RETURN COUNT (r)
```

But what if we have more than just page views, e.g., also link clicks, downloads, form submits,...?

2.4.2.1.2. *The event graph approach*

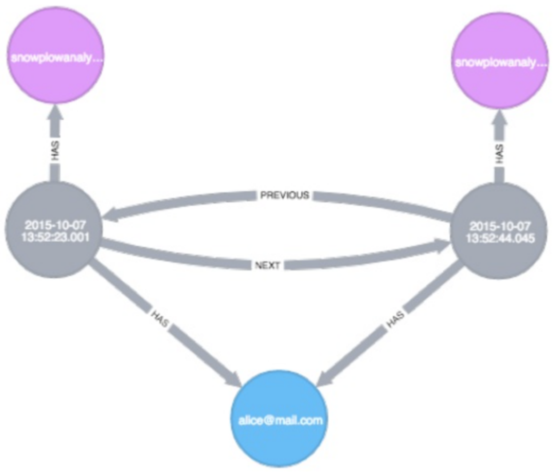


Figure 18 Event Graph Approach (Copyright [5])

A popular option for modelling events in a graph is to make each event a node that is related to the event that happened immediately before it and after it through a NEXT / PREVIOUS relationship. The event node then has outgoing HAS relationships to all of its entities, such as user nodes, context nodes, etc. This is an easy model to do path analysis.

```
MATCH  (u:User)<-[:HAS]-(e:Event)-
[:HAS]->(p:Page)
WHERE u.email = 'alice@mail.com'
RETURN COUNT (DISTINCT p)
```

We can still find all pages visited by the user, but we always have to add a 'hop' in the query, because entities are related to each other only though the event node they belong to.

2.4.2.1.3. *The 'denormalised' graph approach*

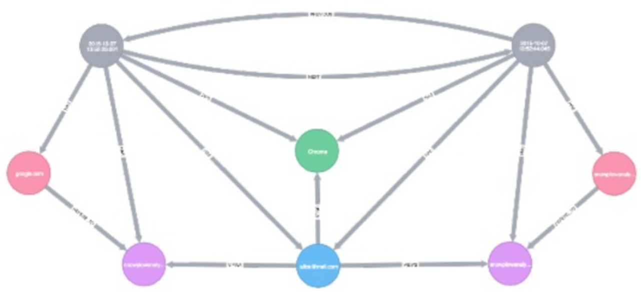


Figure 19 Denormalised Graph Approach (Copyright [5])

There is also the option to "denormalise" the data, i.e. represent the same data in different ways. An example would be a model where each event is a node in a time series, with outgoing relationships to all entities, but there are also relationships between the entities. This adds complexity and redundancy to the model but makes queries easier.

```
MATCH (u:User)-[:VIEWS]->(p:Page)
WHERE u.email = ‘alice@email.com’
RETURN COUNT (DISTINCT p)

MATCH p = (u:User)<-[:HAS]-(Event)-[NEXT*1.5]->(Event)
WHERE u.email = ‘alice@mail.com’
RETURN p
```

- How is modelling event-level data as a graph valuable? One key advantage is that any insight you glean from analysing the relationships of entities in your events, can be readily attached to your existing data set.
- To illustrate, here is a popular use case: an Identity Resolution Graph. Users log in with different accounts, on multiple devices and across multiple networks.
- Contrast that with building the ID graph on top of your existing event graph. You will be able to easily:
    - Find all events for a specific user/device/network.
    - Build relationships that link all known aliases for this user/device/network to the same events.
    - Quickly discover all the user/device/network history, regardless of which alias they are using at the moment.

2.4.2.2. Modelling Time in Events

Again, time can be modelled as many ways as different problems and solutions may emerge. However, the path many people decide to follow and what I think it is the most common is to simply creating a time tree down to the day [22].

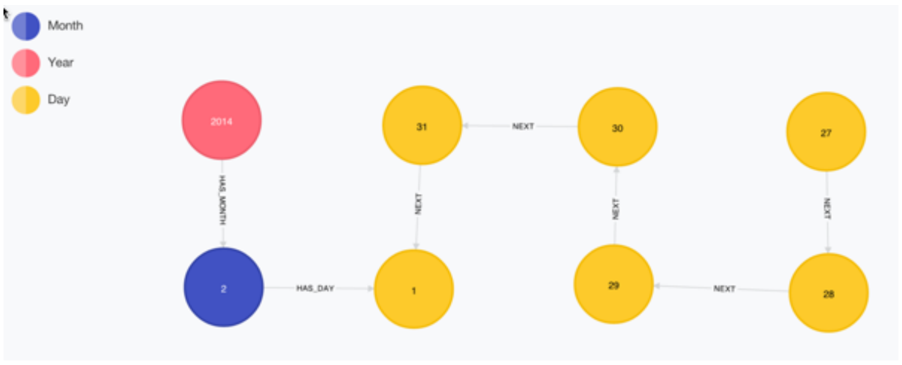


Figure 20 Modelling Time in Events (Copyright [6])

2.5. Interactive Dashboards (Streamlit)

Building and sharing data apps can sometimes be an inconvenient for data scientists and ML engineers, who usually put their focus and time in working with data. Some small teams who do not dispose of front-end engineers or want to view their ML application in a fast, straightforward way have been demanding in recent years a solution to help them. It is then when the first frameworks and libraries begin to appear: some of them integrate directly in Jupyter Notebooks and other allow building and deploying complex apps in just a matter of minutes.

This project´s approach looked for a dashboard (information management tool that visually tracks, analyzes and displays key performance indicators (KPI) allowing users to understand the analytics, serving as an effective foundation for further dialogue) and a web-app interface to configure and execute the project even if you don´t know nothing about it.

2.6. Chatbots (Rasa)

The modern concept of chatbot brings about more than one possible definition (What are their real boundaries? Made with AI?). Despite this we can agree that, generally, a chatbot is a computer program that simulates human conversation through voice commands or text chats or both. These can be classified into 12 major not exclusive between each other categories: AI Chat, Web Chat, Messenger Chat, Customer Support Chat, Live Chat Software, Enterprise Chatbots, SMS Marketing, Marketing Bots and Chatbot Builders. When designing a chatbot, it really all comes down to what your goals are. A well designed & built chatbot will:

- Use existing conversation data (if available) to understand the type of questions people ask.
- Analyse correct answers to those questions through a 'training' period.
- Use ML & NLP to learn context, and continually get better at answering those questions in the future.

The adoption of chatbots was accelerated in 2016 when Facebook opened up its developer platform and showed the world what is possible with chatbots through their Messenger app. Google also got in the game soon after with Google Assistant. Since then, there have been a tremendous amount of chatbot apps built on websites, in applications, on social media, for customer support, and countless other examples.

Rasa is an open-source conversational AI platform. It is composed of the following components:

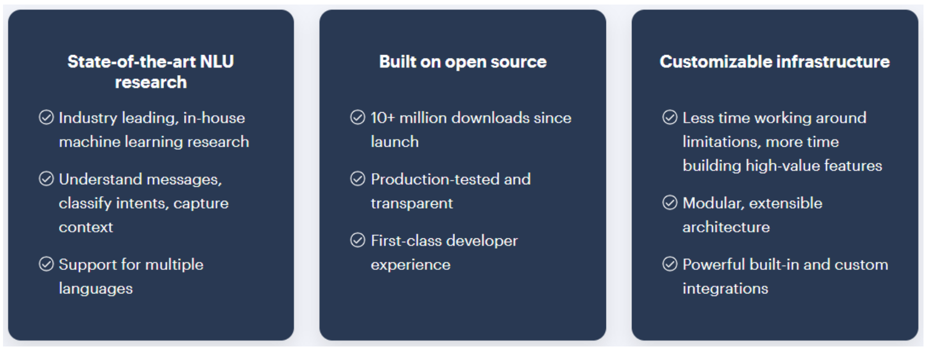
- **Rasa Open Source** is a framework for natural language understanding, dialogue management, and integrations.
- **Rasa X** is a free toolset used to improve virtual assistants built using Rasa Open Source.
- **Rasa Enterprise** subscription with additional features and services that support team collaboration and enterprise-grade deployment to develop and ship customer experiences at a superior scale.

Together, they include all the features to create powerful text and voice based assistants and chatbots. Its main features are:

- **Extract meaning from messages**: turn free-form text in any language into structured data. Supports multiple intents and both pre-trained and custom entities. Fully customizable NLU for any domain, industry, or use case.

- **Hold complex conversations**: retain important context and hold back-and-forth conversations using ML-based dialogue management. Smoothly handle topic changes and seamlessly integrate business logic into conversation flows.
- **Interactive learning**: generate training data by talking to your assistant and provide feedback when it makes an error. Easily share your assistant with test users.
- **Integrate API calls**: use Rasa's custom actions to interact with APIs, databases, and other systems. Connect with knowledge bases, content management systems, and CRMs.
- **Leverage conversation-driven development**: build customer-centered virtual assistants by incorporating user insights and engineering best practices into every part of your team's workflow.
- **Version and manage models**: track and manage your models: promote to production or easily roll back. Integrate with automated testing and CI/CD.
- **Deploy anywhere**: ready-to-deploy Docker containers and orchestration to run Rasa on-prem, or via a preferred cloud provider.

To sum up, Rasa stands out due to its high scalability, community, documentation, easiness and that it is backed up by big companies such as N26, Airbus, Toyota or Adobe.



State-of-the-art NLU research

✓ Industry leading, in-house machine learning research

✓ Understand messages, classify intents, capture context

✓ Support for multiple languages

Built on open source

✓ 10+ million downloads since launch

✓ Production-tested and transparent

✓ First-class developer experience

Customizable infrastructure

✓ Less time working around limitations, more time building high-value features

✓ Modular, extensible architecture

✓ Powerful built-in and custom integrations

Figure 21 Rasa Features (Copyright Rasa)

2.6.1. How Rasa works

Rasa consists of two main components.

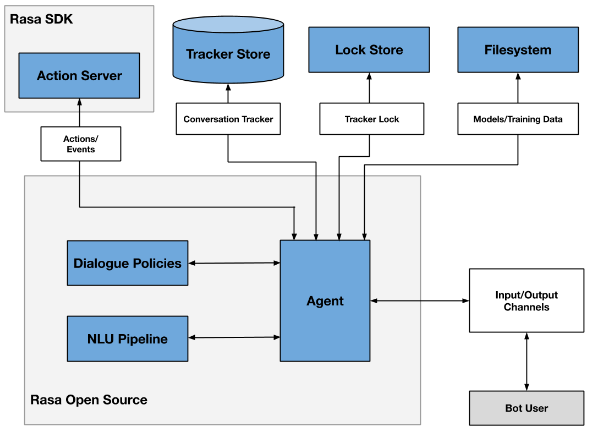


Figure 22 Rasa Architecture (Copyright Rasa)

2.6.1.1. NLU

Natural language processing is used anywhere an application needs to take raw user text as input: whether it's a voice assistant receiving input from speech-to-text software, or a chatbot asking a user to type in their question. Natural language processing is the essential step that turns a string of words into a form that can be interpreted and acted upon by other systems in the application [23]. Natural language understanding is a subset of NLP that classifies the intent, or meaning, of text based on the context and content of the message.

In the insurance industry, a word like "premium" can have a unique meaning that a generic, multi-purpose NLP tool might miss. Regional dialects and language support can also present challenges for some off-the-shelf NLP solutions. Rasa modular architecture and open code base mean it can be plugged custom pre-trained models and word embeddings, built custom components, and tuned models with precision for a unique data set. It works out-of-the box with pre-trained models like BERT, HuggingFace Transformers, GPT, spaCy, and more, and there can be incorporated custom modules like spell checkers and sentiment analysis.

Furthermore, Rasa has support for:

- Multiple intents and hierarchical entities in a single message to model complex transactional conversation flows.
- Control of data privacy following industry regulations like GDPR and HIPAA as it deploys on premises or on a custom private cloud.
- Validating and testing of models measuring F1 score, model confidence, and comparing the performance of different NLU pipeline configurations aside support integration with Git version control, CI/CD and DevOps tools.

2.6.1.1.1. Tuning NLU model

An NLU pipeline allows you to customize your model and finetune it on your dataset [24]. Rasa Open Source will provide a suggested NLU config on initialization of the project, but as a project grows, it's likely that there will be a need to adjust the config to suit training data. Each component in the NLU pipeline processes an input and/or creates an output. Their order is determined by the order they are listed. There are components for entity extraction, for intent classification, response selection, pre-processing, and others. A pipeline usually consists of three main parts: tokenization, featurization and intent classification/response selectors.

2.6.1.2. Core

A framework for ML-based, contextual decision making. The dialogue management component decides the next action in a conversation based on the context (displayed as Dialogue Policies in the diagram).

2.7. Sentiment Analysis

On Summer 2020 when the research & documentation part was done, I found two papers that gathered the actual progress that has and is being done in this field, predicted its future and made some suggestions of possible things to improve and new paths to follow.

Emotion is intrinsic to humans and consequently, emotion understanding is a key part of human-like AI. Sentiment Analysis as a field has come a long way since it was first introduced as a task nearly 20 years ago [25]. It has widespread commercial applications in various domains like marketing, risk management, market research and politics, to name a few. Only in the past few years has emotion recognition in conversation (ERC) gained attention from the NLP community due to the growing availability of public conversational data (Facebook, YouTube, Reddit, Twitter, and others) [26].

Throughout the development of sentiment analysis, ML-based approaches (both supervised and unsupervised) have employed myriad of algorithms that include SVMs, Naïve Bayes Classifiers, nearest neighbours, combined with features that range from bag-of-words, lexicons to syntactic features such as parts of speech. Within neural methods, much like other fields of NLP, present trends are dominated by the contextual encoders. Models like BERT or ELMo, and their adaptations have achieved the state-of-the-art performance on multiple sentiment analysis datasets and benchmarks.

As for the upcoming trends in sentiment analysis, we can distinguish between aspect-based (when a piece of text comprises of multiple aspects with varied sentiments associated to them) and multimodal sentiment analysis. Multimodal fusion is at the heart of multimodal sentiment analysis with an increasing number of works proposing new fusion techniques. These include Multiple Kernel Learning, tensor-based non-linear fusion, memory networks, amongst others. The granularity at which such fusion methods are applied also varies from word to utterance-level. These are three key directions that can aid future research: complex fusion methods vs simple concatenation, lack of large datasets and fine-grained annotation.

In contextual sentiment analysis we can distinguish between influence of topics (employ contextual language models to decipher word senses in contexts and assign the corresponding polarity), sentiment analysis in monologues and conversational context (contextual sentence and word-embeddings can improve the performance of the state of the art NLP systems by a significant margin utilizing surrounding utterances in a video as context or inferring implicit sentiment from context), user, cultural and situational context and role of common-sense knowledge in sentiment analysis (utilizing common-sense associating aspects with their sentiments can be highly beneficial for sentiment analysis. This kind of knowledge-graphs connect the aspects to various sentiment-bearing concepts via semantic links. Additionally, semantic links between words can be utilized to mine associations between the opinion target and the opinion bearing word).

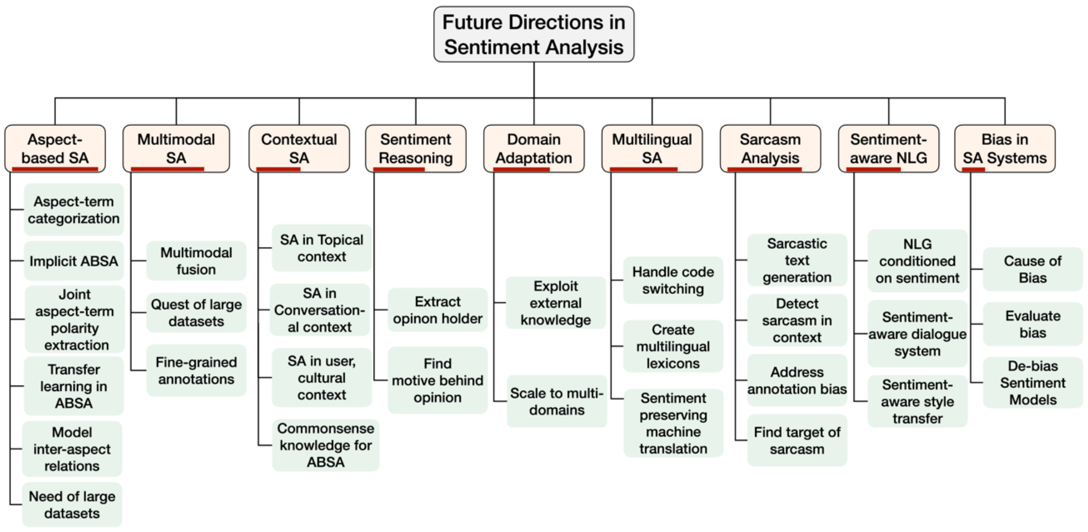

Future Directions in Sentiment Analysis
Aspect-based SA
Aspect-term categorization
Implicit ABSA
Joint aspect-term polarity extraction
Transfer learning in ABSA
Model inter-aspect relations
Need of large datasets
Multimodal SA
Multimodal fusion
Quest of large datasets
Fine-grained annotations
Contextual SA
SA in Topical context
SA in Conversation-al context
SA in user, cultural context
Commonsense knowledge for ABSA
Sentiment Reasoning
Extract opinon holder
Find motive behind opinion
Domain Adaptation
Exploit external knowledge
Scale to multi-domains
Multilingual SA
Handle code switching
Create multilingual lexicons
Sentiment preserving machine translation
Sarcasm Analysis
Sarcastic text generation
Detect sarcasm in context
Address annotation bias
Find target of sarcasm
Sentiment-aware NLG
NLG conditioned on sentiment
Sentiment-aware dialogue system
Sentiment-aware style transfer
Bias in SA Systems
Cause of Bias
Evaluate bias
De-bias Sentiment Models

Figure 23 Future Direction in Sentiment Analysis (Copyright [15])

In sentiment reasoning it is important to distinguish who detects the entity whose sentiment is being determined, whereas why reveals the stimulus/reason for the sentiment. Most of the sentiment analysis research works to date are about classifying contents into positive, negative, and neutral. This oversimplification of the analysis task has resulted in the saturation of any breakthrough. The future research in sentiment analysis should focus on what drives a person to express positive or negative sentiment on a topic or aspect.

In domain adaptation, curating large amounts of training data for every domain is impractical. The use of external knowledge with multi-relational knowledge graphs would be a fancy solution. Scaling up to many domains requires retraining as and when the target domain changes.

In sarcasm analysis, detecting sarcasm is highly challenging due to the figurative nature of text, which is accompanied by nuances and implicit meanings. In addition, sarcasm also depends on a person´s personality, intellect, and the ability to reason over common-sense. In the literature, these aspects of sarcasm remain under-explored. The quest for better contextual modelling is open (one that can explicitly understand facts and incongruity across sentences. These models are also not interpretable; hence, they fail to explain when and how they rely on the context. The main contributions have emerged from the speech and text community. Change of tone, overemphasis on words, straight face, are some such cues that indicate sarcasm.

In sentiment-aware natural language generation (NLG), present-day models are not trained to produce affective content that can emulate human communication. Some applications are emotional chatbots, personality-conditioned text generation or pattern-based approaches for the generation of emotional sentences. We, human beings, count on several variables such as emotion, sentiment, prior assumptions, intent, or personality to participate in dialogues and monologues. In other words, these variables control the language that we generate. The area of controlled text has also percolated into dialogue systems. The aim here is to equip emotional intelligence into these systems to improve user interest and engagement. Two key functionalities are important to achieve this goal:

- Given a user-query, anticipate the best emotional/sentiment response adhering to social rules of conversations.
- Generate the response eliciting that emotion/sentiment.

In bias in sentiment analysis systems, an author´s stylistic sense of writing can also be one of many sources of bias.

Emotion Recognition in Conversation (ERC) ideally requires context modelling of individual utterances in order to identify emotions in each of them. Conversations are broadly categorized into two categories: task oriented and chit-chat (also referred to as non-task oriented).

In research challenges, recent works on ERC strive to address several key research challenges that make the task of ERC difficult to solve:

Modelling emotions categorically classifies them into a fixed number of discrete categories. In contrast, dimensional model describes emotion as a point in a continuous multidimensional space. In the categorical front, Plutchik´s wheel of emotions defines eight discrete primary emotion types, each of which has finer related subtypes. On the other hand, Ekman concludes six basic emotions (anger, disgust, fear, happiness, sadness, and surprise). Most dimensional categorical models adopt two dimensions (valence and arousal), mapping emotion into a continuous spectrum rather than hard categories. This enables comparison of two emotional states using vector operations, whereas comparison is not trivial for categorical models. However, Ekman´s model has a major drawback as it is unable to ground complex emotions. On the other hand, complex emotion models such as Plutchik´s make it very difficult for the annotators to discriminate between related emotions. Newer ERC datasets have employed only categorical model due to its more intuitive nature. Most of the available datasets for emotion recognition in conversation adopted simple taxonomies, which are slight variants of Ekman´s model.

Annotation with emotion labels is challenging as the label depends on the annotator's perspective. The annotators need to be aware of the interlocutor´s perspective for situation-aware annotation.

Context is at the core of the NLP research. The notion of context can vary from problem to problem. Emotional dynamics of conversations consists of two important aspects: self and inter-personal dependencies. Self-dependency deals with the aspect of emotional influence that speakers have on themselves during conversations. Interpersonal dependencies relate to the emotional influences that the counterparts induce into a speaker. During the course of a dialogue, speakers also tend to mirror their counterparts to build rapport. Modelling self and inter-personal relationship and dependencies may also depend on the topic of the conversation as well as various other factors like argument structure, interlocutors´ personality, intents, viewpoints on the conversation, attitude towards each other, etc. Hence, analysing all these factors is key for a true self and interpersonal dependency modelling that can lead to enriched context understanding. The contextual information can come from both local and distant conversation history. Distant contextual information is useful mostly in the scenarios when a speaker refers to earlier utterances spoken by any of the speakers in the conversational history.

Individuals have their own subtle way of expressing emotions. For instance, some individuals are more sarcastic than others. For such cases, the usage of certain words would vary depending on if they are being sarcastic. Speaker profiling based on preceding utterances often yields improved results.

In presence of emotion shift, the state-of-the-art methods keep mimicking the same emotion for a particular party, since an abrupt change of emotion is unlikely. Hence, these methods fail in most cases where a change occurs. To tackle this, a new problem of detecting emotion shift can be framed:

- Based on the historical and the present utterance, is there an emotion shift (binary classification)?
- If there is a shift then what is the target emotion (multi-label classification)?

Fine-grained emotion recognition involves a deeper understanding of the topic of the conversation, interlocutor opinion and stand.

Multiparty conversation is more challenging in comparison with dyadic conversations due to the difficulty in tracking individual speaker states and handling co-references.

Unlike context modelling, emotion reasoning does not only find the contextual utterances in conversational history that triggers the emotion of an utterance, but also determines the function of those contextual utterances on the target utterance. It is hard to define a taxonomy or tagset for emotion reasoning.

Recognizing emotion of an utterance in a conversation primarily depends on these following three factors:

1. The utterance itself and its context defined by the interlocutor´s preceding utterances in the conversation, as well as intent and topic in the conversation.
2. The speaker´s state comprising variables like personality and argumentation logic.
3. Emotions expressed in the preceding utterances.

To conclude, an effective emotion-shift recognition model and context encoder can yield significant performance improvement over chit-chat dialogue, and even improve some aspects of chat-oriented dialogue. Moreover, challenges like topic-level speaker-specific emotion recognition, ERC on multiparty conversations and conversational sarcasm detection can form new research directions. Additionally, fine-grained speaker-specific continuous emotion recognition may become of interest for the purpose of tracking emotions during long monologues.

3. Objectives

For a start, the main objective of this project as of 6th October 2020 was to develop, using affective computing and deep learning techniques, an intelligent system that allows both the detection and generation of emotions and their storage in knowledge graphs. The project would consist of at least 5 modules:

- Emotion and facial attributes detection using deep learning.
- Element detection in a scenario using deep learning.
- Emotion model with ALMA paper.
- NoSQL knowledge graph database storage and retrieval.
- Output elaboration* to the received stimuli.

The project itself has undergone several, not very dramatic transformations with a view to adapt to each moment circumstances. By and large, the moment that has had more impact in the project execution has been the contract with ITAINNOVA, which began the 6th of April and gave remarkable motivation and powerful tools in its development. The project stages are explained in further detail in 5.and it would be convenient to have a look at them before continue reading.

Next in the timeline, in the October 2020 – April 2021 phase, the project had to be cut-off in two objectives:

- Element detection in a scenario using deep learning as it requires a powerful machine to train a custom ML model.
- Output elaboration to the received stimuli: again, a more or less powerful machine is needed in addition to my little experience with Unity which finally resulting in letting this work if I definitely joined the ITAINNOVA team who could explain me with further detail how Pilar worked plus having a powerful tool to manage it.

In the last phase, April 2021 – June 2021, two more objectives were added aside from recovering a previously cut-off one:

- Output elaboration to the received stimuli.
- Rasa chatbot to be able to interact with the project components.
- Streamlit web-app dashboard to be able to interact with the project components.

4. Methodology

Software Engineering is the systematic application of engineering approaches to the development of software. It includes branches like software design, development, maintenance, requirements and testing. The software development lifecycle is the process of dividing software development work into smaller, parallel or sequential steps. The methodology may include the pre-definition of specific deliverables and artifacts that are created and completed by a project team to develop or maintain an application. Most modern development processes can be vaguely described as agile. Other methodologies include waterfall, prototyping, iterative and incremental development, spiral development, rapid application development, and extreme programming.

Agile software development methodologies have been around for quite a long time since software development began to be seen more as a mindset or way of thinking [27]. According to the Agile Alliance: "Agile Software Development is an umbrella term for a set of methods and practices based on the values and principles expressed in the Agile Manifesto.". There exists a number of popular agile development approaches including the following: Scrum, XP, Kanban, Lean and more. I won´t be digging into each of them in detail but only justify that I´ve considered their advantages, drawbacks and finally chose a final way of working.

4.1. Background information

As I work for ITAINNOVA´s Big Data and Cognitive System´s department, I´m surrounded by data scientists and ML engineers, among others. I considered the possibility to research what is their workflow and how could I benefit from it in my project aside the Agile methodology contents I studied in the Software Engineering subject in the computer science degree.

What about Agile in the ML, AI or Data Science discipline? Can the same principles as in Agile be extrapolated? Long debate has concerned project managers and developers in these fields. What makes things complex is the fact that the roles in the organization between the application-focused Agile groups and the data-focused methodologies groups are not the same. While frequently the project manager is the centre of the Agile universe, connecting the sides of business and technology development, the data organization is the centre of the data methodology universe, connecting the roles of data scientists, data engineer, business analyst, data analyst, and the line of business. Frequently the language of communication is not the same, with Agile sprints focused on functions and features, and data "sprints" focused on data sources, data cleansing, and data models [28]. Clearly the two parts of the organization serve the same overall master so we need to combine these two approaches into a cohesive whole that provides organizations the power they need to deliver AI projects reliably.

The answer, of course, is a blended methodology that starts from the same root of business requirements and splits into two simultaneous iterative loops of Agile project development and Agile-enabled data methodologies. We can think of this as an Agile Cross-industry standard process for data mining (CRISP-DM) enhanced Agile approach. It's quite likely that CRISP -DM is not the only data methodology we can use here, but it is certainly suitable. However, there are some parts of AI project development that are not addressed by either methodology including:

- Development of conversational applications and dealing with conversational model development.
- Challenges around bias in model development and iterative de-biasing.
- Hardware-centric model deployment challenges and iterative loops around that.
- Simultaneous AI algorithm evaluation and ensembling which imposes additional methodology challenges.

To that end, there are approaches and methodologies that fill in these gaps with an AI-centric approach. Methodologies such as Cognitive Project Management for AI (CPMAI) made specific enhancements to the methodology to meet AI-specific requirements, especially as they pertain to the above requirements, and as they can be implemented in organizations with already-running

Agile teams and already-running data organizations. Introducing something new and foreign is a sure way to get resistance. So, the key is to provide a blended approach that simultaneously delivers the expected results to the organization and provides a framework for continued iterative development at the lowest risk possible.

Translating all this information to my current scenario, it is not required to follow a pure nor hybrid CPMAI as the project is not leading with a classic ML development methodology flow: the models currently used have already been trained and developed by third parties (see objectives and implementation sections). Concerning this fact, focus will be in consequence put to an agile methodology yet having the knowledge that ML development methodologies are there and they are able to merge with agile approaches resulting in very interesting hybrid approaches.

4.2. Agile methodology

Before digging more into what has been applied in this project, let´s give Agile a brief introduction: Agile methodology is a practice that helps continuous iteration of development and testing in the software development process. Agile breaks the product into smaller builds. Scrum is just one of the many iterative and incremental agile software development process that allows us to focus on delivering the business value in the shortest time. The Scrum Framework usually deals with the fact that the requirements are likely to change or most of the time not known at the start of the project (adaptation). That has been exactly my project´s state throughout its entire lifetime and why Scrum was chosen as my daily driver agile methodology. Furthermore, the advantages of Scrum are clear compared to other methodologies: transparency allows for the project to be followed by all members, there is a lot of motivation to meet deadlines, focus on quality delivery and allowance of priorities reorganisations ensuring sprints that have not yet been completed get more attention (inspection).

Conversely, as exposed before, I took some other interesting features from other approaches, which may also could have fitted to manage this project yet did not provide the flexibility Scrum does. It is indeed worthwhile mentioning them:

- I didn´t want to lose the ability to more detailly track my task flow as sometimes a block in one of them could last either one day to more than a week so the Kanban board is something that I love (and once tried for the first time, I cannot live without) to use in my projects.

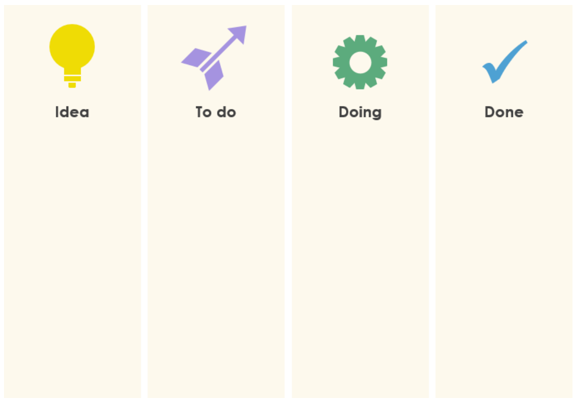


Figure 24 Kanban Board (Copyright Visual Paradigm)

- Lean philosophy created by Toyota in the 1970s focuses on the delivery of value to the customer and on eliminating waste from the process. These principles can be applied to software as Mary and Tom Poppendieck proved in their 2003 publication, Lean Software Development.

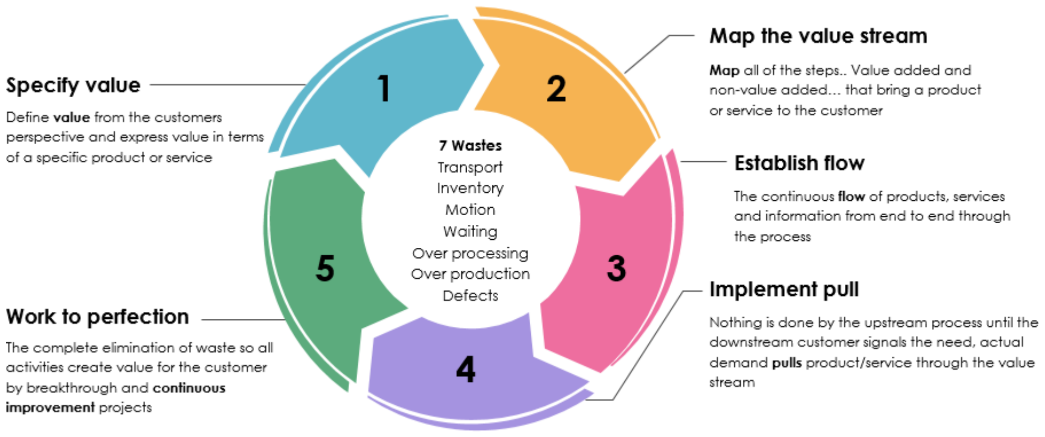

Specify value
Define value from the customers perspective and express value in terms of a specific product or service
Map the value stream
Map all of the steps.. Value added and non-value added... that bring a product or service to the customer
Establish flow
The continuous flow of products, services and information from end to end through the process
Implement pull
Nothing is done by the upstream process until the downstream customer signals the need, actual demand pulls product/service through the value stream
Work to perfection
The complete elimination of waste so all activities create value for the customer by breakthrough and continuous improvement projects
1
2
3
4
5
7 Wastes
Transport
Inventory
Motion
Waiting
Over processing
Over production
Defects

Figure 25 Lean Principles (Copyright Visual Paradigm)

- ML data-flow perspective. Although I have focused more on the software-side implementation rather on the modelling, parameter finetuning, mathematician-statistical side of ML, I undoubtedly had to look at my project from some of the referenced ML development workflows perspectives.

4.3. My custom Scrum solution

Once put into context, let´s dive in how I implemented my personalized Scrum solution. The first consideration to be made is that a typical Scrum methodology cannot be applied to this project: it is meant to be a POC (Proof of Concept) in order to assess the viability of an application by investigating and generate knowledge; the scope was changed every time my tutor Rafa found limitations.

More concisely, this project is not a huge enterprise-level project, only consisting of three people involved in it, its main purpose is academical, not commercial, so it has to be a solid product of course, but it is more thought to be something with which the student who is developing it is able to learn-by-doing with. In addition, time is important, and it is not desired to spend more time in methodology rather than doing the real work. Besides, it is followed a special (not pure) Scrum process consisting of the following components [29].

4.3.1. Roles

- Scrum Master: Rafael del Hoyo Alonso.
- Scrum Product Owner: ITAINNOVA.
- Scrum Team: Marcos Caballero.

4.3.2. Artifacts

As exposed before in My custom Scrum solution and in In the above table it can be seen a colour differentiation corresponding to the different stages the project has undergone (see Logbook).

, there are only three people in the project, it is not a very big problem and there have been a lot of external factors that have influenced its development. Thereby, artifact haven´t been developed from the start in a standardized way.

4.3.2.1. Product backlog

Based on the 3. Objectives section. Instead of creating user stories, we took 2. Background & State of Art architecture schema and divided assigned each of the boxes a user story.

4.3.2.2. Sprint backlog

Clear sprints were not defined. For instance, it was not known when could I gain access to the Unity code of Pilar, to the GPU server, to ITAINNOVA, to a certain virtual machine, etc. Every two weeks the scrum team and scrum master met and refined the project´s course based on the current circumstances.

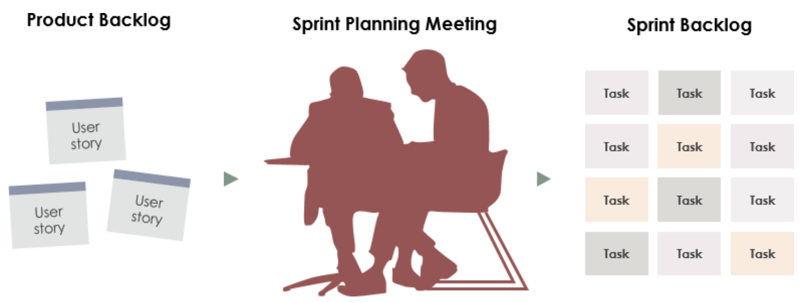
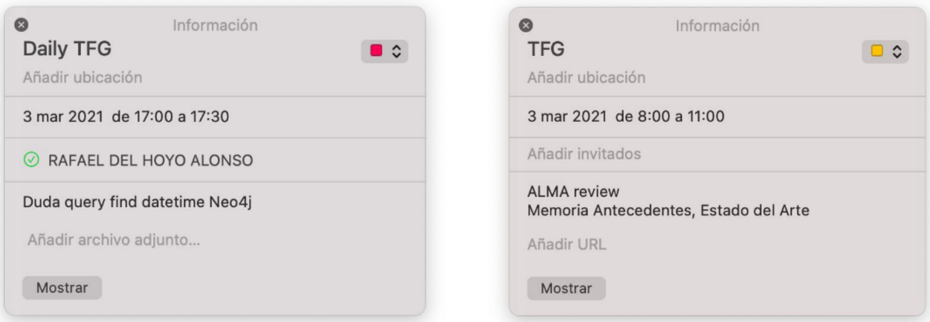


Figure 26 Scrum Components (Copyright Visual Paradigm)

4.3.3. Events

- Sprint planning: happened every two weeks unless some exception like exams dates or holidays (see Work Plan).

- Daily stand-up: The scrum team (Marcos Caballero) met with the project master on a weekly basis or every time they had a specific doubt that caused a block. They (he) also held daily meeting (with himself) organized in their (his) calendar. A sample calendar event was a working session throughout the day (usually 2 or 3 hours before a break) with the tasks to do and observations of each task.


Información
Daily TFG
Añadir ubicación
3 mar 2021 de 17:00 a 17:30
RAFAEL DEL HOYO ALONSO
Duda query find datetime Neo4j
Añadir archivo adjunto...
Mostrar

Información
TFG
Añadir ubicación
3 mar 2021 de 8:00 a 11:00
Añadir invitados
ALMA review
Memoria Antecedentes, Estado del Arte
Añadir URL
Mostrar

Figure 27 Weekly and Daily Meetings (Own Copyright)

- Sprint review & retrospective: this happened mainly once a month (we skipped one every two meetings mainly because we did not have much to make comments on).
- Backlog refinement meeting: happened twice: at the very first meeting of the project and when the scrum team entered working in ITAINNOVA.

4.4. Work Plan

Day + Hours	Meeting Description	Work Description
22 July 2020 **2 hour**	Initial work was planned: I was proposed with four final degree project alternatives and asked to choose one while Rafa talked with ITAINNOVA Human Resources department to ask if they were going to hire new interns in October.	Documentation labour for the four projects to see which one could benefit more my career path.
5 August 2020 **16 hours**	I chose Pilar project and Rafa suggested me to ask Martín about his previous work. We also started to figure out a new approach to the project and I was given the opportunity to do some research into which path of AI I would like to put more focus on provided its main idea and that we disposed of time.	Investigation labour reading papers and testing codes.
26 August 2020 **12 hours**	We discussed the general architecture schema with all the components the project would have. In the backend side, I was introduced to Apache Tinkerpop (Neo4j) and in the frontend side with Pilar Unity project.	Work through Apache Tinkerpop, investigation labour in how I could take the most of graphs and test Pilar Unity project.
9 September 2020 **12 hours**	We reviewed the work up to date regarding Apache Tinkerpop (Neo4j) and Pilar Unity project, and I was also introduced to the Deepface framework. We started thinking about changing Tinkerpop for Neo4j Python driver and leaving Pilar aside for a while until I had a more powerful machine to work with at ITAINNOVA.	Investigate Neo4j with its Python driver and continue getting graphs culture reading ML/DS/AI articles about use cases with them.

23 September 2020 **16 hours**	We reviewed the first draft of the Deepface code and thought the way to customize it with the ALMA code. In addition, we started to formalize a project proposal and continue the conversations with ITAINNOVA to see if they were going to hire internships for October.	Investigate Deepface framework.
7 October 2020 **16 hours**	Work continued reviewing the Deepface + ALMA code.	Investigate Deepface framework and ALMA code.
21 October 2020 **16 hours**	Work continued reviewing the Deepface + ALMA code.	Investigate Deepface framework and ALMA code.
4 November 2020 **16 hours**	Work continued reviewing the Deepface + ALMA code and reviewing the Neo4j database schema.	Investigate Deepface framework, ALMA code and Neo4j graph schema.
18 November 2020 **12 hours**	Work continued reviewing the Deepface + ALMA code and reviewing the Neo4j database schema.	Investigate Deepface framework, ALMA code and Neo4j graph schema.
2 December 2020 **16 hours**	Work continued reviewing the Deepface + ALMA code, reviewing the Neo4j database schema and the queries to be made with inserting events and extracting information depending on the different use cases.	Investigate Deepface framework, ALMA code and Neo4j queries with python driver.
16 December 2020 **12 hours**	This meeting was held but skipped because not a lot of advances had been made and I was in the first exams round.	Investigate Deepface framework, ALMA code and Neo4j queries with python driver.
2 January 2021 **8 hours**	Review regarding general system´s solidity.	Code refactoring/optimization to ensure each of the parts connected well and everything worked as expected.
20 January 2021 **12 hours**	Review Neo4j in general (schema, queries, Python driver).	Re-do some schema, queries and code adjustments for further project scalability (future versions or ampliations of the project).
3 February 2021 **16 hours**	Review Neo4j queries + review Telegram bot + Heroku deploy. Jesús: Present Introduction part.	Revision of the graph database stuff and investigate Telegram bots deploying one in Heroku (not very complex one). Write Introduction part.
20 February 2021 **16 hours**	Rafa: Review regarding general system´s solidity + review Neo4j queries with the Python driver. Jesús: Present Background & State of Art.	Code refactoring/optimization to ensure each of the parts connected well and everything worked as expected. Write Background & State of Art.
10 March 2021 **20 hours**	Review Neo4j queries with the Python driver + review deploy instance to free Amazon EC2 cloud instance + Test system´s accuracy. These tests were inaccurate so an adjustment of the ALMA + Deepface was needed in addition to refactor the whole code into classes.	Re-do some Neo4j stuff (schema adjustments, queries and Python driver), deploy database, do the tests, start making refinements in general to the system to improve accuracy and classify .py files. Write background & State of Art.
24 March 2021 **12 hours**	Review the whole code from the previous meeting to-dos + prepare to move to ITAINNOVA for the next month.	ITAINNOVA job interview, re-do tests to ensure everything worked accurately as expected. Write background & State of Art.
7 April 2021 **60 hours**	To-do: real time statistics report + config file + CUDA + Streamlit.	Figure out how to show in real time a dataframe with each execution´s data + configuration file with different parameters and app modes + optimize execution for CUDA + develop Streamlit web-app. Write background & State of Art.

21 April 2021 **60 hours**	Review previous meeting to-do + Streamlit web-rtc component + Rasa chatbot + Streamlit data plotting.	Investigate Streamlit web-rtc component + develop Rasa chatbot + refine data plotting with Streamlit. Write background & State of Art.
5 May 2021 **60 hours**	Streamlit web-rtc component + Rasa chatbot/custom actions + review data plotting in Streamlit. Jesús: Present Objectives & Methodology	Investigate Streamlit web-rtc component + develop Rasa chatbot/custom actions. Write background & State of Art.
19 May 2021 **56 hours**	Rasa chatbot/custom actions/ Duckling + video playback. Rasa X into local mode. Added Telegram + web Rasa integrations + return to Pilar Unity Project.	Integrate custom actions with Duckling date extractor and Neo4j queries, move Rasa X into local mode, add Telegram + Web Rasa integrations and investigate Pilar Unity Project. Write Objectives & Methodology + Development.
2 June 2021 **50 hours**	Pilar Unity project + Development explanations.	Write Development + start preparing demo with Pilar Unity project.
12 June 2021 **50 hours**	Final report revisions + demo revisions.	Write Conclussion, Results, Economic Study + wrap everything up + prepare demo.

Table 1 Sprint planning and work

In the above table it can be seen a colour differentiation corresponding to the different stages the project has undergone (see Logbook).

5. Economic Study

Making a budget estimation of this project is a challenging task because it is made purposely to study the viability of a real business need. In such case that this POC is successful, it won't be sold as an only asset; it will be embedded into other ITAINNOVA´s assets as complementary modules.

Budgeting a project starts with the company´s culture: were the technologies utilized for this project specifically thought for it or were they already thought in the past? Estimating the real time spent by other people with their corresponding salaries is also a difficult task. In this case, it can be said that the different technologies used in this project come from previously acquired knowledge in the company by other people, so this amount of time/money will be omitted.

In the next table the total cost is broken down into sections:

Strategy	
Business idea	500€
Requirement definition	500€
Production	
Architectural Design	1000€
Development	16472.66€
Deployment	14.85€

Documentation	42€
Summary	
Total	18529.51€

Table 2 Economic Study Summary

In the project strategy it can be found:

- Business idea: market research is made with view to provide an accurate business analysis of the current situation of human machine interaction systems, in particular avatars capable of replicating human emotions with the business opportunities they could provide. How can value be created with the latest technologies by embedding them in something that no one has yet done? Scope, risk, integration, time, cost, stakeholders, quality, communication and procurement have to be evaluated.
- Requirement definition: A research exercise is undertaken as a basis for meeting that business need: the different technologies featuring in the project will be chosen. Knowledge management is crucial in companies because it can save lots of time taking already made company´s assets instead of developing new ones.

In the project production it can be found:

- Architectural design: deep analysis of the different tech stack components, how they fit in together, etc. Special attention has to be paid to the scalability and maintenance of the product as it is designed for the long-term future.
- Development: according to indeed, a junior software developer [30] and Data Scientist [31] salaries in Zaragoza are about 17.5K€ and 25.3K€ (brute). If we consider this project has lasted around a year and that 75% of the work has been data science, we can approximate that quantity to 23.4K€ and with tax reduction applied (30% aprox), 16.3K€.
    - Hardware: my personal computer is valued at 1000€ as of 2015 and the GPU server approximately at 2000€ as of 2017. If the average amortisation time of a computer is 4 years and in both cases this is fulfilled, we can say that the computer cost has been 0. Nevertheless, a Logitech C270 HD webcam costs 25€.
    - Software: all of the software I´ve been using comes with a community edition for academical purpose. However, if the project intends to make a profit, a fee is charged for each of the software used.
        - Fork: $49.99 –> 41.24€
        - StarUML: $129 –> 106.42€.
        - The rest is open-source or costless.
    - Learning & Training: done in open-source or costless platforms such as YouTube.
- Deployment:

  - On-premise virtual machine instance for the database hosting: unknown cost yet if Amazon EC2 charges $19,73 for 1 year of a 4Gb RAM, 2-core, 10Gb HDD Linux machine, we could estimate as a bit less, $18 –> 14.85€.
  - Rasa is deployed in the local machine (see Rasa X for the explanation): 0€
- Documentation: throughout my time at ITAINNOVA and in San Jorge University, I have been utilizing Microsoft Office 365 Suite (especially Teams, Excel, PowerPoint and Word). 4,20€*10months(aprox)=42€.

To sum up, pricing estimates have been made approximately according to my previous experiences working in consulting companies and with the knowledge acquired in Software Engineering and Project Management subjects. *Note: price exchange at the time of conversion: $1 = 0.82€.*

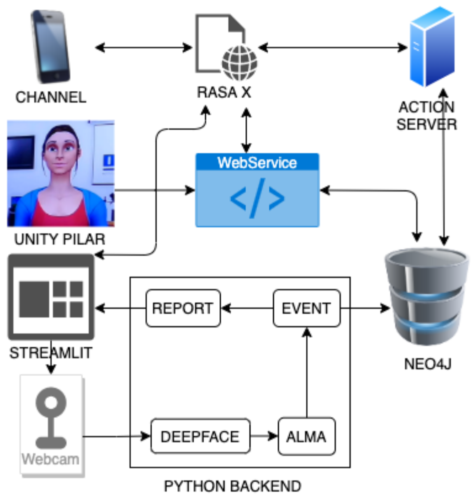
6. Analysis, Design & Development

The target system follows the same approach as exposed in Background & State of Art: this means that there will be a computer vision system with built-in AI to capture emotions, an emotion processing model, a database, a chatbot, an avatar and a dashboard.

To introduce the analysis and design part, an architecture schema will depict each of the system´s components and a series of UML (Unified Modelling Language) diagrams will provide a standard way to tackle the design. Secondly, it will be explained how coding was made (tools): from the ones used for an early investigation task to try out, familiarize and learn the capabilities of the technologies (which most of them were new to me) to the development and production ones which helped by providing a fast, clear and reliable way to code, test, correct and deploy.

From that point onwards, a Gannt diagram and a general overview of the technical development matching Work Plan and Logbook with a deeper implementation description of the specific technology stack matching see Architecture Design will be provided.

6.1. Architecture Design



There will be three backends:

- PYTHON BACKEND: in charge of the emotion-side calculations.
- RASA X & ACTION SERVER: in charge of the chatbot.
- WebService: in charge of communicating with UNITY PILAR front-end side.

There will be one non-relational, Graph database (NEO4J).

Figure 28 System Architecture (Own Copyright)

There will be three frontends:

- STREAMLIT: the Dashboard.
- UNITY PILAR: only code that runs on C#.
- CHANNEL: Telegram. REST channels are also available for STREAMLIT and UNITY PILAR but are not included for a matter of simplicity.

6.2. UML Diagrams

For having a complete project vision, there have been designed three kinds of diagrams: use case, class and sequence.

6.2.1. Use Case Diagram

A user has three ways of interacting with Pilar: the Rasa chatbot, Streamlit dashboard and avatar Unity project. Meanwhile, Pilar makes use of two additional modules to make her emotion and mood calculations: Deepface and ALMA.

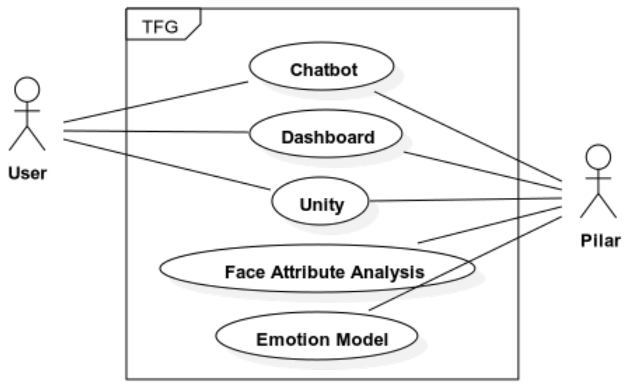


Figure 29 Use Case Diagram (Own Copyright)

6.2.2. Class Diagram

99% of the code is made in Python (see the final note).

- Main: the main class from which the Streamlit dashboard is launched. It includes two app modes: real-time stream and video playback. Real Time calls the web-rtc component in DeepFaceNew1 and Video Playback in DeepFaceNew2. Data is accessed in different ways, it is also displayed in different ways having its proper functions but they both lay in the same principle: displaying a dataframe with 6 different graphs showing facial attributes of a person (emotions, age and gender) and Pilar´s dominant emotion, mood and desirability, praiseworthiness and liking. A thread is created for this commitment. The config file can also be changed to customize execution of the code.
- DeepFaceNew 1 and 2: they are basically the same files as for inference matters yet 1 calls transform interface from web-rtc to display it in Streamlit whereas 2 generates an OpenCV window to display the video inference output. In the stream() or _annotate_image(), the bag of detected faces (which frames threshold can be adjusted) is run the multiple facial attributes ML models and then displayed a square with that inference information for an adjustable number of seconds.
- ALMA: following ALMA´s paper [3] and Martin´s implementation, a demography object is created from DeepFace´s data and calculations made to update Pilar´s mood, dominant emotion and appraisal rules. When ALMA class is first called, its elements are initialized by a default value which evolves as DeepFace detections are being made.
- Event: the previous demography object together with a timestamp is inserted into the database (if debug mode is on).
- Report: a report is shown in the dashboard containing the dataframe and graphics if report option is on.
- Helpers: auxiliary file containing functions oftenly utilized over the project.
- Neo4jCypher: as its name suggest, it is responsible of querying and inserting data into the graph database following the Python driver guidelines.
- app_pilar: webserver called from the avatar Unity project. Mainly just the text_bot() function is called which makes Pilar available for chat and expressing her emotions in real time.
- Actions: getName returns the name of the last person Pilar has seen. WhatDidYouSee and HowDidYouFeel return the person and mood/dominantEmotion of the time specified extracted by duckling.

Note: there has not been included Pilar avatar Unity project due to a matter of simplicity. Just a few lines of TextBot.cs have been changed and the file is attached in the repository.

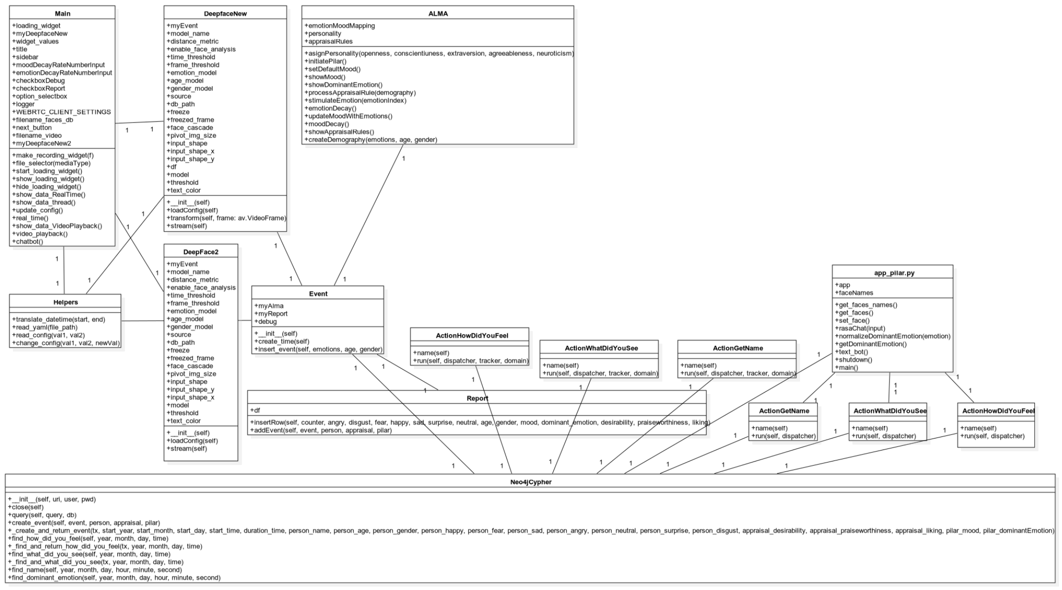


Figure 30 Class Diagram (Own Copyright)

6.2.3. Sequence Diagram

There are at least two execution modes. However, the code provides lots of possibilities more as it will be shown in the demo.

6.2.3.1. Streaming a video from the dashboard

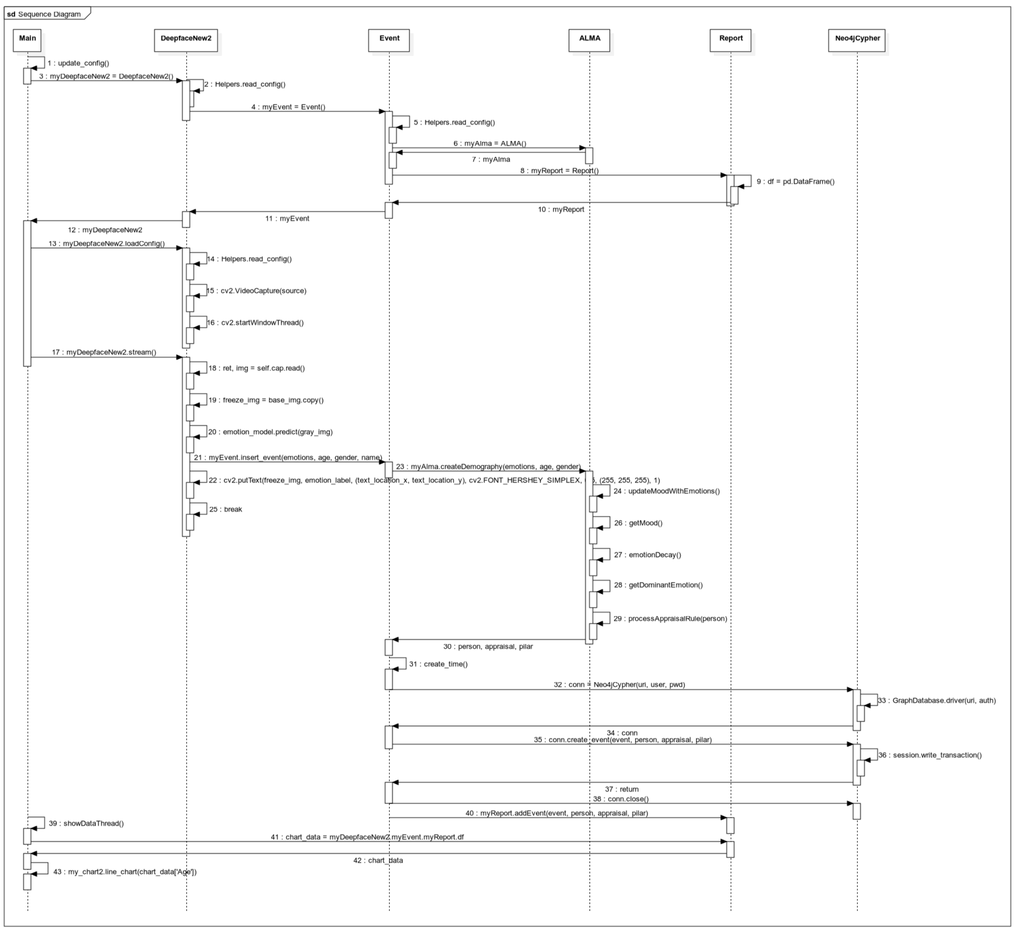

sd Sequence Diagram
Main
DeepfaceNew2
Event
ALMA
Report
Neo4jCypher
1 : update_config()
3 : myDeepfaceNew2 = DeepfaceNew2()
2 : Helpers.read_config()
4 : myEvent = Event()
5 : Helpers.read_config()
6 : myAlma = ALMA()
7 : myAlma
8 : myReport = Report()
9 : df = pd.DataFrame()
10 : myReport
11 : myEvent
12 : myDeepfaceNew2
13 : myDeepfaceNew2.loadConfig()
14 : Helpers.read_config()
15 : cv2.VideoCapture(source)
16 : cv2.startWindowThread()
17 : myDeepfaceNew2.stream()
18 : ret, img = self.cap.read()
19 : freeze_img = base_img.copy()
20 : emotion_model.predict(gray_img)
21 : myEvent.insert_event(emotions, age, gender, name)
23 : myAlma.createDemography(emotions, age, gender)
22 : cv2.putText(freeze_img, emotion_label, (text_location_x, text_location_y), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 1)
24 : updateMoodWithEmotions()
25 : break
26 : getMood()
27 : emotionDecay()
28 : getDominantEmotion()
29 : processAppraisalRule(person)
30 : person, appraisal, pilar
31 : create_time()
32 : conn = Neo4jCypher(uri, user, pwd)
33 : GraphDatabase.driver(uri, auth)
34 : conn
35 : conn.create_event(event, person, appraisal, pilar)
36 : session.write_transaction()
37 : return
38 : conn.close()
40 : myReport.addEvent(event, person, appraisal, pilar)
39 : showDataThread()
41 : chart_data = myDeepfaceNew2.myEvent.myReport.df
42 : chart_data
43 : my_chart2.line_chart(chart_data['Age'])

Figure 31 Sequence Diagram 1 (Own Copyright)

6.2.3.2. Asking Pilar how did she feel yesterday (avatar + chatbot)

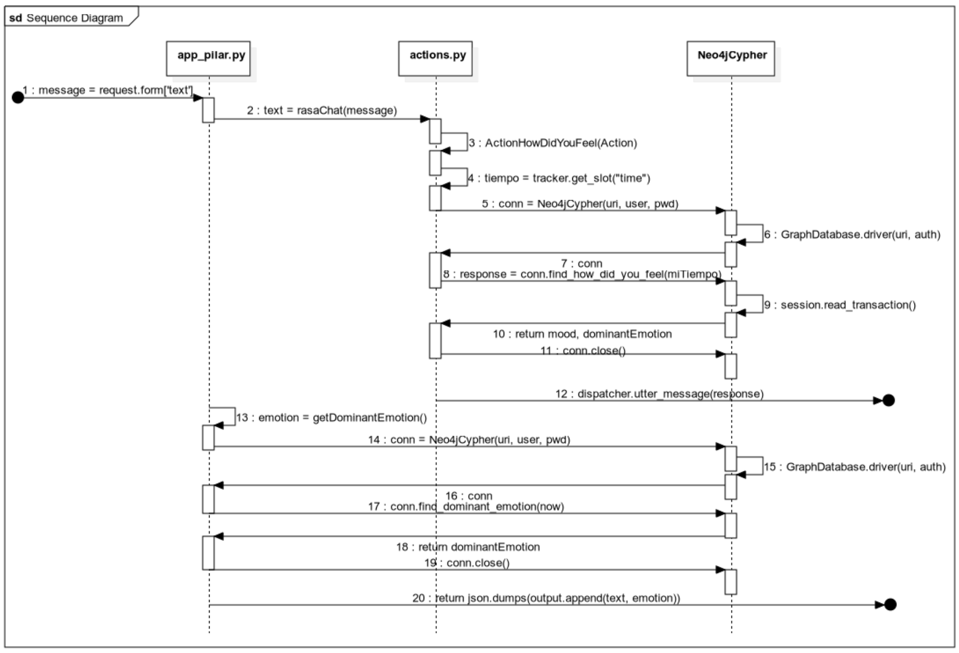

sd Sequence Diagram
app_pilar.py
actions.py
Neo4jCypher
1 : message = request.form['text']
2 : text = rasaChat(message)
3 : ActionHowDidYouFeel(Action)
4 : tiempo = tracker.get_slot("time")
5 : conn = Neo4jCypher(uri, user, pwd)
6 : GraphDatabase.driver(uri, auth)
7 : conn
8 : response = conn.find_how_did_you_feel(miTiempo)
9 : session.read_transaction()
10 : return mood, dominantEmotion
11 : conn.close()
12 : dispatcher.utter_message(response)
13 : emotion = getDominantEmotion()
14 : conn = Neo4jCypher(uri, user, pwd)
15 : GraphDatabase.driver(uri, auth)
16 : conn
17 : conn.find_dominant_emotion(now)
18 : return dominantEmotion
19 : conn.close()
20 : return json.dumps(output.append(text, emotion))

Figure 32 Sequence Diagram 2 (Own Copyright)

6.3. Investigation task (Jupyter Notebooks)

Jupyter Notebooks are an open-source web application that allows to create and share documents that contain live code, equations, visualizations and narrative text. I coded little snippets as explained in Analysis, Design & Development which I´ll later insert in my production application. They can be found in the folders with an "_old" including ALMA, TelegramBot, Streamlit, Neo4jCyper, and Deepface all ending in underscore old.

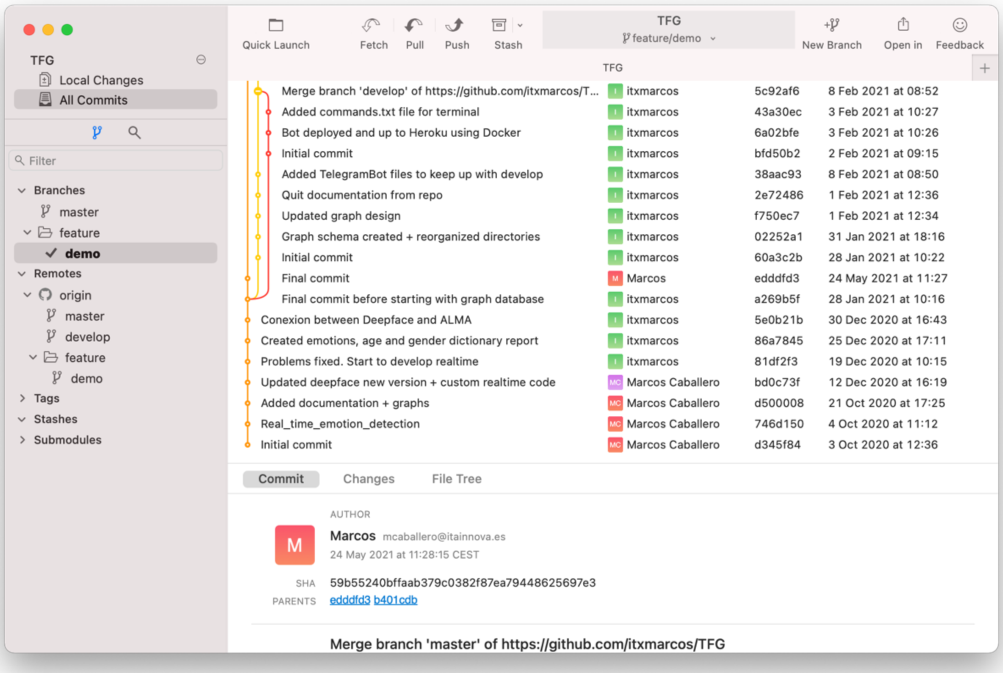
6.4. Setting up the environment

The tools used in the daily basis once the investigation part was done will be described here. As mentioned in Analysis, Design & Development, the objective to achieve by having a correct environment setup is that after the analysis and design, the developer should only focus in thinking, coding, testing and deploying as more efficiently as possible.

As for my IDE or code editor of choice, I began using PyCharm Community, which is a very powerful, thought for professionals IDE made by JetBrains. Its major downside was that it consumes a lot of resources as it is majorly written in Java and therefore had to prioritize execution time and performance. I switched to Visual Studio Code, which is still not very performance-friendly compared to a native app yet provides a handful of interesting functionalities (not as much as PyCharm, to be said).

Conda is an open-source package and environment manager running on Windows, macOS and Linux. It easily creates, saves, loads and switches between environments and quickly installs, runs and updates packages and their dependencies. Miniconda is a free minimal installer for Conda. As 99% of the project´s code is made in Python (except the C# for Pilar´s frontend, which is minimal), Anaconda suite is highly recommended and used all over AI developers and is well suited for this kind of workflows.

My version control system of choice is Git and instead of using its built in installation command-line tool, I worked with Fork, a fast and friendly git client for Mac and Windows.



Merge branch 'develop' of https://github.com/itxmarcos/T...	itxmarcos	5c92af6	8 Feb 2021 at 08:52
Added commands.txt file for terminal	itxmarcos	43a30ec	3 Feb 2021 at 10:27
Bot deployed and up to Heroku using Docker	itxmarcos	6a02bfe	3 Feb 2021 at 10:26
Initial commit	itxmarcos	bfd50b2	2 Feb 2021 at 09:15
Added TelegramBot files to keep up with develop	itxmarcos	38aac93	8 Feb 2021 at 08:50
Quit documentation from repo	itxmarcos	2e72486	1 Feb 2021 at 12:36
Updated graph design	itxmarcos	f750ec7	1 Feb 2021 at 12:34
Graph schema created + reorganized directories	itxmarcos	02252a1	31 Jan 2021 at 18:16
Initial commit	itxmarcos	60a3c2b	28 Jan 2021 at 10:22
Final commit	Marcos	edddfd3	24 May 2021 at 11:27
Final commit before starting with graph database	itxmarcos	a269b5f	28 Jan 2021 at 10:16
Conexion between Deepface and ALMA	itxmarcos	5e0b21b	30 Dec 2020 at 16:43
Created emotions, age and gender dictionary report	itxmarcos	86a7845	25 Dec 2020 at 17:11
Problems fixed. Start to develop realtime	itxmarcos	81df2f3	19 Dec 2020 at 10:15
Updated deepface new version + custom realtime code	Marcos Caballero	bd0c73f	12 Dec 2020 at 16:19
Added documentation + graphs	Marcos Caballero	d500008	21 Oct 2020 at 17:25
Real_time_emotion_detection	Marcos Caballero	746d150	4 Oct 2020 at 11:12
Initial commit	Marcos Caballero	d345f84	3 Oct 2020 at 12:36

AUTHOR
Marcos mcaballero@itainnova.es
24 May 2021 at 11:28:15 CEST
SHA 59b55240bffaab379c0382f87ea79448625697e3
PARENTS edddfd3 b401cdb

Merge branch 'master' of https://github.com/itxmarcos/TFG

Figure 33 Fork Repository Sample (Own Copyright)

- Git Large File Storage (LFS) replaces large files such as audio samples, videos, datasets, and graphics with text pointers inside Git, while storing the file contents on a remote server like GitHub. I included there my demo videos as they are weighty files. Fork comes with LFS extension so it works out of the box without having to configure anything.
- Gitflow Workflow is a Git workflow that helps with continuous software development. It defines a strict branching model designed around the project release. This provides a robust framework for managing larger projects [32]. I did not utilize DevOps nor CI/CD best practices but adapted them to my own specific needs: a POC to study the technological viability of the proposal generating knowledge. Basically I could be working

at several feature branches at the same time if I had work blocks and I could continue committing without disrupting the other branches code in an organized, schematical way.

6.5. Timeline

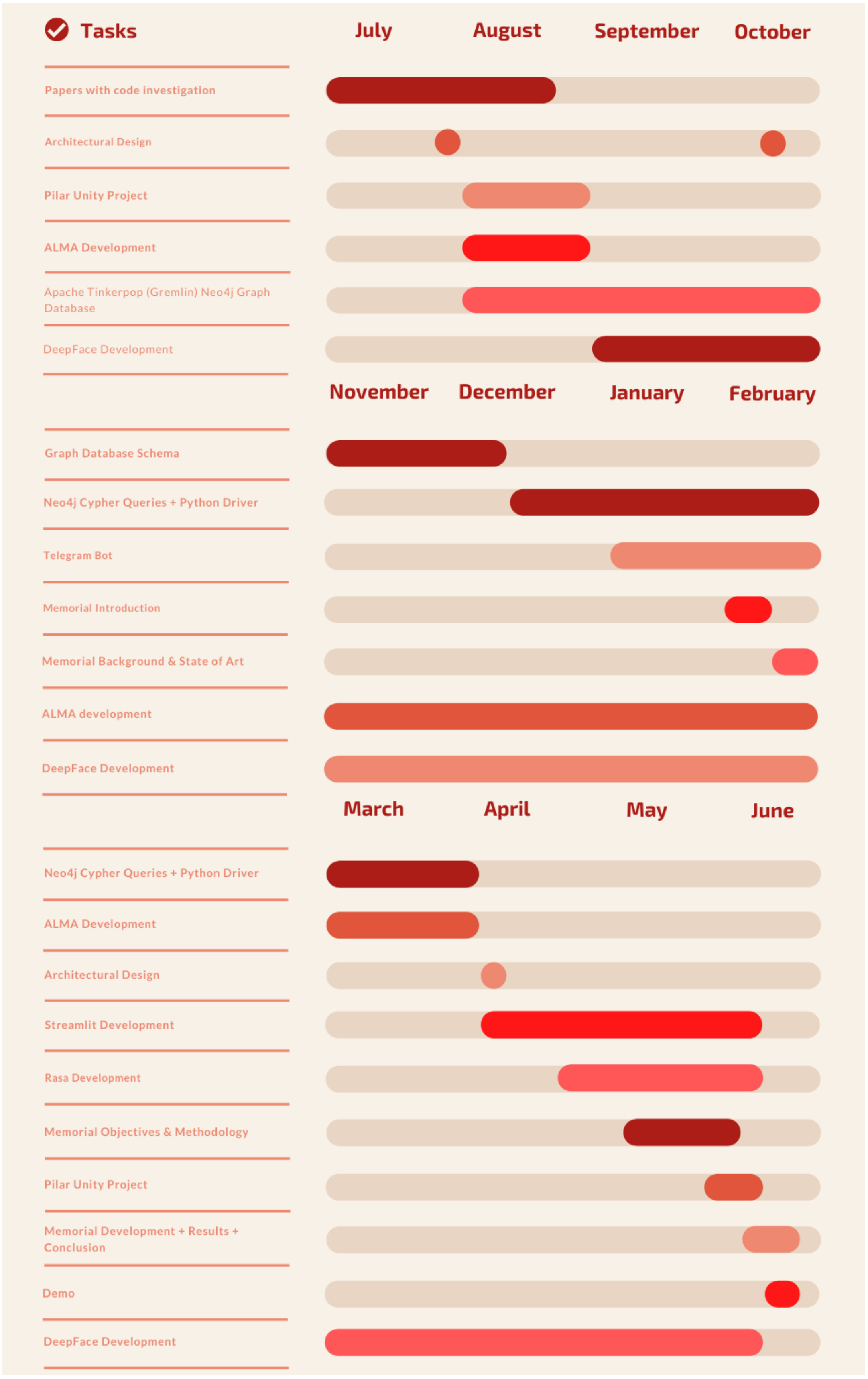

Tasks
July
August
September
October
Papers with code investigation
Architectural Design
Pilar Unity Project
ALMA Development
Apache Tinkerpop (Gremlin) Neo4j Graph Database
DeepFace Development
November
December
January
February
Graph Database Schema
Neo4j Cypher Queries + Python Driver
Telegram Bot
Memorial Introduction
Memorial Background & State of Art
ALMA development
DeepFace Development
March
April
May
June
Neo4j Cypher Queries + Python Driver
ALMA Development
Architectural Design
Streamlit Development
Rasa Development
Memorial Objectives & Methodology
Pilar Unity Project
Memorial Development + Results + Conclusion
Demo
DeepFace Development

Figure 34 Development Timeline Gannt Diagram (Own Copyright)

The above table aims to give a schematic description of the project tasks timeline with a view to generally developing them matching each project´s phase as well as more accurately technically speaking in the following sections.

6.5.1. Summer – October 2020

Since this was a mostly holiday period, a more recreational activity was carried out consisting of reading papers, testing out their codes and really learning to target the direction of the project as mentioned in Background & State of Art. The most outstanding contributions I found as of that date were Emotion Recognition in Conversations [33], Awesome Emotion Recognition in Conversations [34] and Emotion Recognition [35]. The webpage Papers with Code provides a handful of interesting contributions to multiple computer science fields, among which these two searches were the most useful for me: [36], [37].

I also tested out Pilar Unity Project in my personal laptop. It is very complex and making it work and understand the basics took me some time. Meanwhile I started having Meetings with Martin in which he explained technical details about what he had done until he quitted from the project regarding ALMA and started thinking how I could adapt his work to my desired project path. In addition, I also played around with Apache Tinkerpop and read tons of things about graphs to know why it was worth working with them compared to other non-relational databases for the specific needs of this project. When September started, Rafa introduced me to the DeepFace framework.

6.5.2. October 2020 – April 2021

Constant work was performed in this phase resulting in a quality execution of most of the objectives. This includes designing the graph database schema (which if done for the first time makes you think in a very different way compared to other databases), dealing with Cypher (Neo4j graph language) and later with its Python driver. Cypher queries are simple yet they can easily get really complex depending on the value you want to extract from each query (the more far or convoluted a connection is, the more difficult it gets getting the query right). ALMA and Deepface development continues, mainly by refactoring code, adjusting calculations, improving performance and classifying everything to meet the Object Oriented Programming standards. On February I meet for the first time with Jesús Carro and we begin to organise how the report will be written.

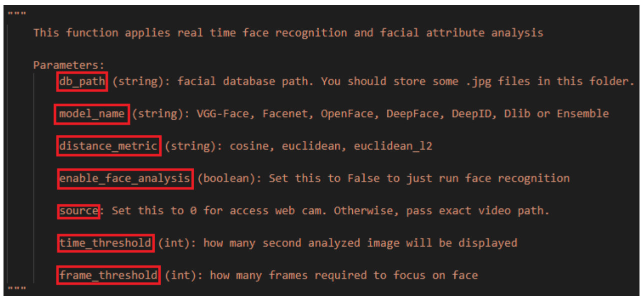
Some considerations had to be made: my laptop is actually not very powerful (not to say it struggles) to handle a ML training process (for object detection custom ML model) nor inference nor Unity edition of Pilar´s project so I was a bit limited by the circumstances. The executions of the project despite the little change I made to the code took ages (from 45 seconds to a minute and a half) and this fact made the development stage a little bit tedious. In spite of the cut-off mentioned in Objectives, Rafa and I thought that it would be better to instead of adding another input to the ALMA model (element detection custom ML model), it would be nicer to add another way to interact with the virtual assistant Pilar rather than with a simple Unity 3D model. Thereby, Rafa showed me a framework called Rasa with which ITAINNOVA had developed several successful projects such as a healthcare COVID-19 assistant for the Aragon Government. It was not until April when I began to work in this part of the project.

6.5.3. April – June 2021

This is by far the phase in which most condensed work has taken place. I migrated my environment to the powerful Windows GPU server I was assigned with and noticed how fast it could execute by enabling CUDA (which configuring can sometimes be a bit painful). I took advantage of all the work done in adjusting well DeepFace before entering ITAINNOVA (refactorings, calculations refinement, etc.). I also started my two new objectives: the Rasa chatbot and the Streamlit dashboard. Both were challenging tasks because I had to start from scratch and the deployment of the chatbot in RasaX and making the Web-rtc component work caused a lot of blocks and a lot of research to understand how they worked at a lower level to try to fix problems at a higher level. Report (Development, Results and Conclusion) also took a noticeable amount of time.

6.6. Emotion Detection (Deepface)

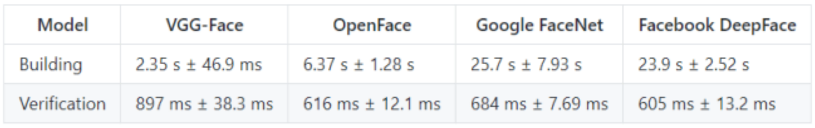
Before starting this section, it is important to discern between the DeepFace model for face recognition and the DeepFace framework which uses this and other modules. When starting to implement the Deepface framework, I dived into its Github repository [38] and watched its main contributor´s (Sefik Ilkin Serengil) videos in YouTube [39]. The first thing that comes into mind when initially having a quick look to the mentioned resources is how little steep the learning curve is; it adjusts very well to the different learning styles developers may have. I started following the YouTube videos tutorials playing around with the different components of the library in a Jupyter Notebook. When I felt confident enough, I created a Python file with a call to DeepFace.stream(), which is the Streaming and Real Time Analysis.



```
"""
    This function applies real time face recognition and facial attribute analysis

    Parameters:
        db_path (string): facial database path. You should store some .jpg files in this folder.

        model_name (string): VGG-Face, Facenet, OpenFace, DeepFace, DeepID, Dlib or Ensemble

        distance_metric (string): cosine, euclidean, euclidean_l2

        enable_face_analysis (boolean): Set this to False to just run face recognition

        source: Set this to 0 for access web cam. Otherwise, pass exact video path.

        time_threshold (int): how many second analyzed image will be displayed

        frame_threshold (int): how many frames required to focus on face
"""
```

Figure 35 DeepFace Real Time Analysis Function (Own Copyright)

The documentation says "Calling stream function under the DeepFace interface will access your webcam and apply both face recognition and facial attribute analysis. Stream function expects a database folder including face images. VGG-Face is the default face recognition model and cosine similarity is the default distance metric similar to verify function. The function starts to analyse if it can focus a face sequentially 5 frames. Then, it shows results 5 seconds." Another positive side about DeepFace is that it is so flexible; provided you don´t want to use the default configuration as in the above paragraph, it allows you to customize a lot of parameters to better fit the use case of your ML application.



Model	VGG-Face	OpenFace	Google FaceNet	Facebook DeepFace
Building	2.35 s ± 46.9 ms	6.37 s ± 1.28 s	25.7 s ± 7.93 s	23.9 s ± 2.52 s
Verification	897 ms ± 38.3 ms	616 ms ± 12.1 ms	684 ms ± 7.69 ms	605 ms ± 13.2 ms

Figure 36 Face Recognition Speed Comparison (Copyright Sefik Ilkin Serengil)

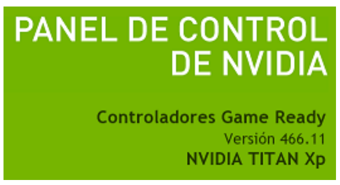
In addition to the verification time metrics, Facebook´s Deepface proved in real-time testing to be less laggy than the rest of the models.

Both DeepFaceNew and DeepFaceNew2 have a constructor, a load_config method and a transform & _annotate_image or stream methods which are in charge of displaying frames and in case of detecting a face for x frames, display its facial attributes detection for y seconds (both x and y are customizable). OpenCV library aimed at real-time computer vision that allows "putting boxes" around the detected faces in the scene and displaying a handy chart with a demography report which will be explained later.

Why are there two DeepFaceNew classes? DeepFaceNew extends and modifies the original code from the DeepFace framework package. It does so in two different ways because of Streamlit´s web-rtc limitations which will later be discussed. Basically, I was able to embed a real time webcam stream in the browser but could not manage to do the same with a video playback (note this functionality is not natively supported). Therefore, I had to differ two different ways on the behaviour in capturing frames.

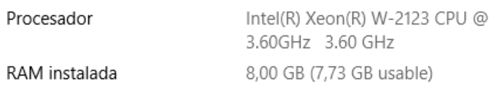
6.6.1. Tensorflow + Cuda

Sefik Ilkin Serengil´s Deepface framework runs using a TensorFlow backend. As ML computation is based on operations with matrix (or more specifically, tensors), GPUs and TPUs gain performance over general-purpose CPUs in this specific task. Thankfully, ITAINNOVA provided a powerful computer with these specifications:



PANEL DE CONTROL DE NVIDIA

Controladores Game Ready
Versión 466.11
NVIDIA TITAN Xp



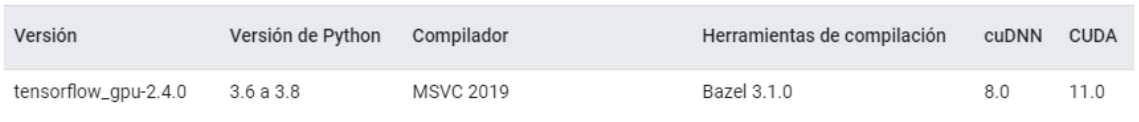
Procesador	Intel(R) Xeon(R) W-2123 CPU @ 3.60GHz 3.60 GHz
RAM instalada	8,00 GB (7,73 GB usable)

Figure 37 ITAINNOVA Computer Technical Specifications (Own Copyright)

CUDA [40] is a parallel computing platform and programming model developed by NVIDIA for GPU-accelerated applications. With CUDA, the sequential part of the workload runs on the CPU – which is optimized for single-threaded performance – while the compute intensive portion of the application runs on thousands of GPU cores in parallel. The NVIDIA CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers. cuDNN accelerates widely used deep learning frameworks, including Caffe2, Chainer, Keras, MATLAB, MxNet, PaddlePaddle, PyTorch, and TensorFlow.

Configuring your application to run with CUDA usually requires a big dose of patience, specially if it has not been done before. However, some life saving tutorials [41] are available out there which are much better (at least from my starter´s point of view) than TensorFlow's official documentation. In the end, it all comes to some basic steps:

1. According to Tensorflow´s compatibility matrix [42], install the proper Python, CUDA and cuDNN versions.



Versión	Versión de Python	Compilador	Herramientas de compilación	cuDNN	CUDA
tensorflow_gpu-2.4.0	3.6 a 3.8	MSVC 2019	Bazel 3.1.0	8.0	11.0

Figure 38 CUDA, CuDNN, Tensorflow Compatibility Matrix (Copyright Tensorflow Documentation)

2. Finally, install tensorflow-gpu using Miniconda package manager.
3. Check if succeeded with TFG/README.md code snippet.

Issues [43] may arise during this process. In such case, doing some workarounds [44] will be required. Github´s forum [45] was also useful for facing some issues.

6.7. Emotion Model (ALMA)

This implementation was already developed by Martín. Summarizing, I just refactored the code (classified it, adjusted numbers for calculations and general optimization) and added the part which will tell how Pilar would react depending on what she is seeing through DeepFace´s eyes.

- In the processAppraisalRules method which as well calls translateEmotion, I defined my own rules to set how, for instance, seeing a <18 year old woman could affect Pilar.
- In the createDemography function, a demography report is created based on ALMA calculations in order to show it in the Streamlit dashboard.
- Parameters such as the mood or emotion decay rates to observe different emotional behaviours.

6.8. Events Storage (Neo4j)

From all the available graph databases, Neo4j and Apache Tinkerpop (an abstraction layer over graph databases such as Neo4j) were thought to be the best fitting options as they provided straightforward to use yet complete capabilities. It was finally Neo4j due to a high learning curve Apache Tinkerpop has and that Neo4j itself is so simple to use. Despite completing Apache Tinkerpop GettingStarted tutorial [46], it was thought that the Neo4j Python driver [47] was simple enough for the concerning application.

6.8.1. Modelling my use case

I took Learn how to model your application data as a Neo4j graph data model [48] course which explained the fundamentals to create a graph database using Neo4j best practices. In addition, the modelling documentation [49] was also a big support. Doing this task was a bit confusing because as explained in 2.4, the classic SQL modelling approach can´t be followed. Thanks to the course, I discovered that a whiteboard with a pen and Arrow tool were perfect mates for having a first draft of my database and insert it in Neo4j with the Cypher language. Following 2.4.2.3 and 2.4.2.4 approaches, I created a time tree down to the day [50] and events in a "denormalised" way.

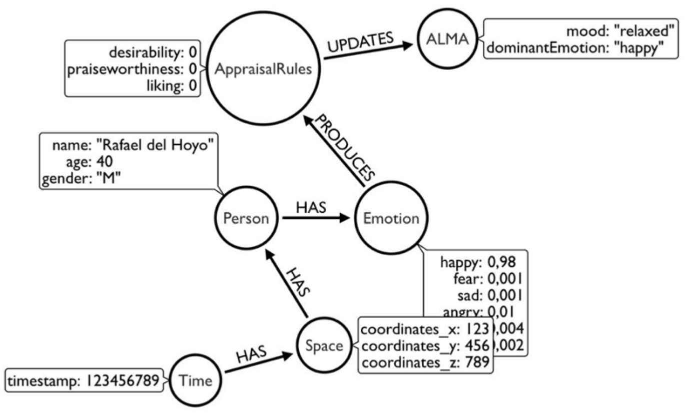


Figure 39 Graph Modelling Draft 1 (Own Copyright)

Here are included the first and second drafts of the database schema.

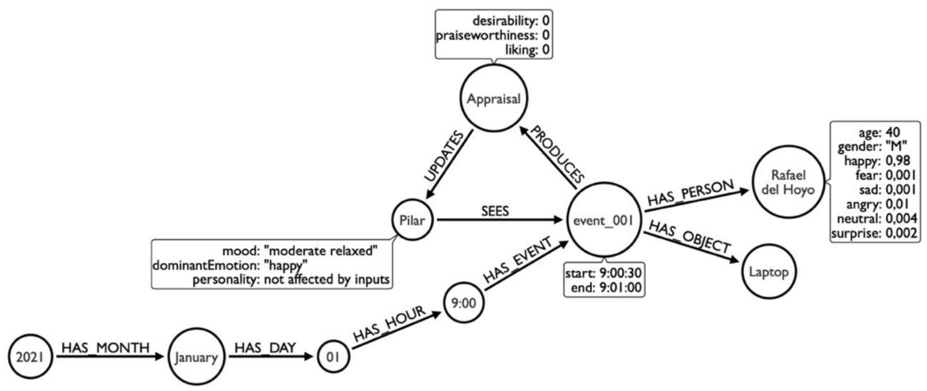


Figure 40 Graph Modelling Draft 2 (Own Copyright)

Once the final design was finished and the demo was made with a variety of eliciting emotion videos, the graph looked "mathematically beautiful" like this:

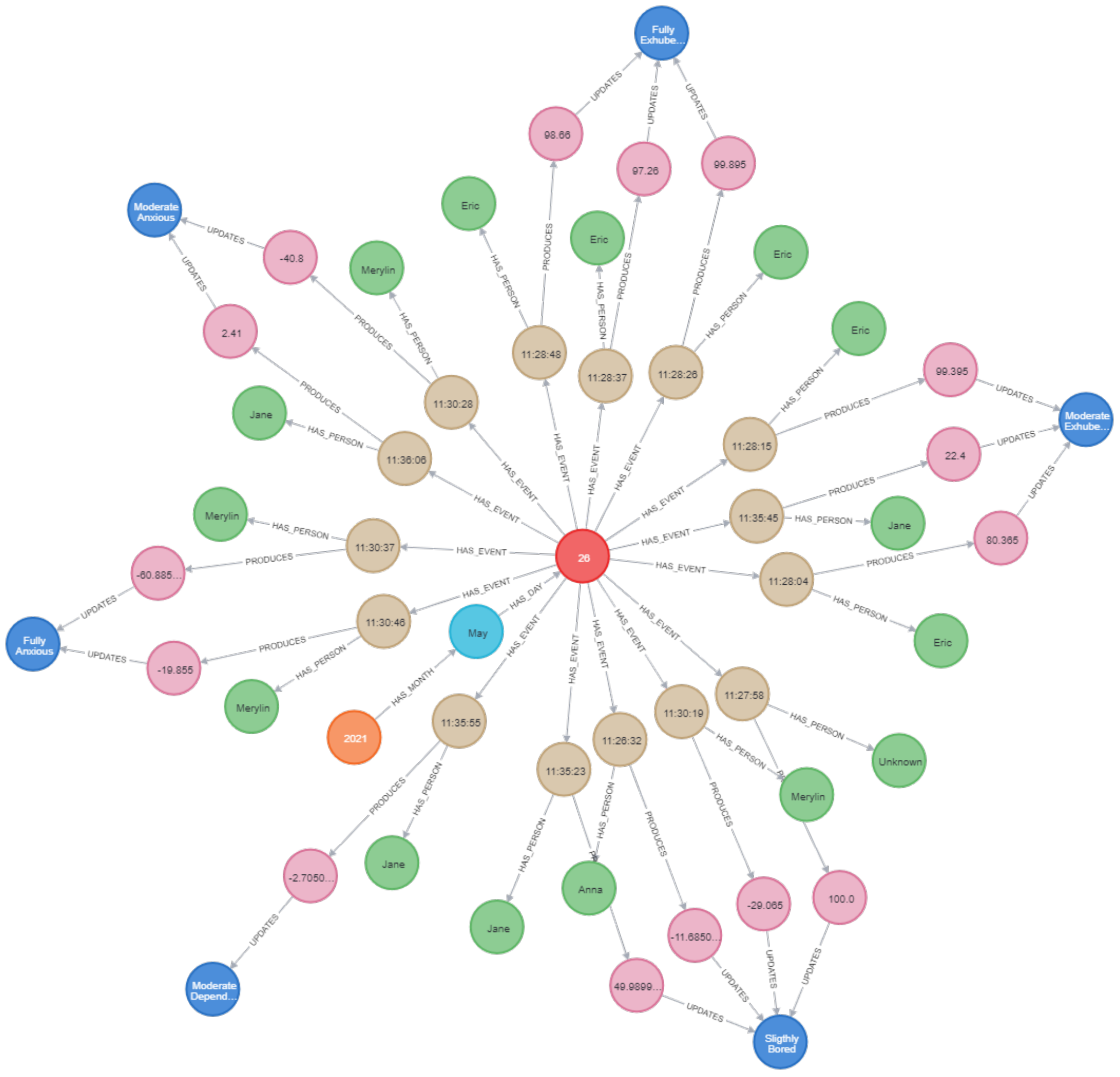


Figure 41 Final Graph Model Sample (Own Copyright)

6.8.2. Queries

Neo4j offers a very complete documentation about clauses [51], dates, datetimes and durations [52] and accessing components of temporal instants [53]. Furthermore, I also utilized its cheatsheet [54] which comes in handy when you are familiarizing with the keywords.

6.8.2.1. Insert

The most important things of this key were:

- MERGE: allows not to insert repeated nodes. Without it, we will be constantly inserting for instance year 2021, month June and day 17th.
- Neo4j datetime data type conversion: allows to query events by datetime.

6.8.2.2. Retrieve

I can query using datetime events using comparison operators.

- *How did you feel?* My mood was Slightly Bored and my dominant emotion was admiration. This query is for the rasa chatbot conversation.
- *What did you see?* I saw Rafael del Hoyo. This query is for the rasa chatbot conversation.
- *Name?* Rafael del Hoyo. This query is for the rasa chatbot name entity.
- *Dominant emotion?* Admiration. This query is for Pilar Unity emotion expression.

6.9. Interactive Dashboard (Streamlit)

When looking for inspiration about how the dashboard web app could look like, Awesome Streamlit [55] makes a huge contribution to the community by presenting a list of curated projects. While surfing through its forum, I found a lot of contributors share very useful projects too. Streamlit is a very new yet promising project and it is the community besides its creators who make it grow exponentially from release to release. These are just a few examples:

- How to save a widget state? [56]
- Vertically center the content of columns. [57]
- Per-session persistent state. [58]
- How to use Streamlit with VS Code. [59]

Without these contributions, I would not have been able to develop my application as is and would have had to look for more complex workarounds or choose another framework.

6.9.1. Streamlit at low level

As mentioned before, Streamlit is a very new project and there are things that have not been developed yet and need a workaround. In consequence, I have been required to understand how it works at low level. Upon each execution, Python script is executed from top to button. Each execution of the Python script renders the frontend view sending data from Python to JS as args. to the component. The frontend triggers the next execution via Streamlit.setComponentValue() sending data from JS to Python as component value.

6.9.1.1. Threads

Threading is available with Streamlit but declarations are made with an additional layer of abstraction over Python´s threading library. This is due to the fact that Streamlit app already runs 0n its own thread. Refreshing the dataframe information to plot emotion data in the multiple graphs required this functionality.

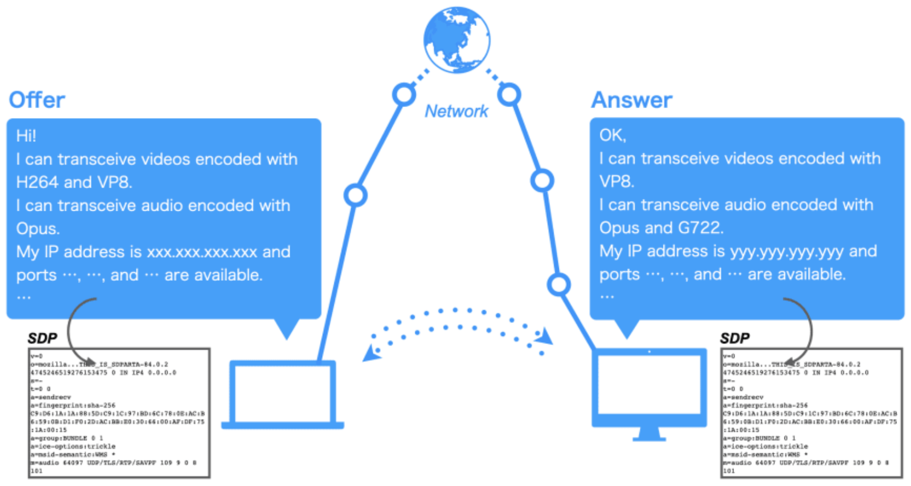
6.9.2. Streamlit components

6.9.2.1. Lottie

Lottie is an Airbnb project thought for easily adding high-quality, rendered in After Effects animations in real time. In this case there has been included a loading widget [60] for the video playback processing app mode.

6.9.2.2. Web-rtc

Yuichiro Tachibana (Tsuchiya) announced [61] on January 10th 2021 her new component streamlit-webrtc, a new way to deal with real-time media streams. She prepared a tutorial [62] and a demo [63] so that alongside the Github repository source code [64], developers could more easily learn how to integrate this component into their Streamlit apps. Furthermore, she did a low-level depiction article [65] in which she describes in more detail the implementation based on Aiortc.


Network
Offer
Hi!
I can transceive videos encoded with H264 and VP8.
I can transceive audio encoded with Opus.
My IP address is xxx.xxx.xxx.xxx and ports ···, ···, and ··· are available.
···
SDP
Answer
OK,
I can transceive videos encoded with VP8.
I can transceive audio encoded with Opus and G722.
My IP address is yyy.yyy.yyy.yyy and ports ···, ···, and ··· are available.
···
SDP

```
v=0
o=mozilla...THIS_IS_SDPARTA-84.0.2
4745246519276153475 0 IN IP4 0.0.0.0
s=-
t=0 0
a=sendrecv
a=fingerprint:sha-256
C9:D6:1A:1A:88:5D:C9:1C:97:BD:6C:78:0E:AC:B
6:59:0B:D1:F0:2D:AC:BB:E0:30:66:00:AF:DF:75
:1A:00:15
a=group:BUNDLE 0 1
a=ice-options:trickle
a=msid-semantic:WMS *
m=audio 64097 UDP/TLS/RTP/SAVPF 109 9 0 8
101
```

```
v=0
o=mozilla...THIS_IS_SDPARTA-84.0.2
4745246519276153475 0 IN IP4 0.0.0.0
s=-
t=0 0
a=sendrecv
a=fingerprint:sha-256
C9:D6:1A:1A:88:5D:C9:1C:97:BD:6C:78:0E:AC:B
6:59:0B:D1:F0:2D:AC:BB:E0:30:66:00:AF:DF:75
:1A:00:15
a=group:BUNDLE 0 1
a=ice-options:trickle
a=msid-semantic:WMS *
m=audio 64097 UDP/TLS/RTP/SAVPF 109 9 0 8
101
```

Figure 42 Web-rtc SDP Graphical Explanation (Copyright Yuichiro Tachibana (Tsuchiya))

WebRTC has a preparation phase called "signalling", during which the peers (JS frontend and Python server) exchange metadata (available media codecs, the IP address of the peer, available ports, etc.) called "offers" and "answers" in order to gather necessary information to establish the connection. It all works under SDP (Session Description Protocol) and HTTP req/res mechanism. Network connectivity information is called ICE candidate, which contains the information about the methods available for the peer to use to make a connection.

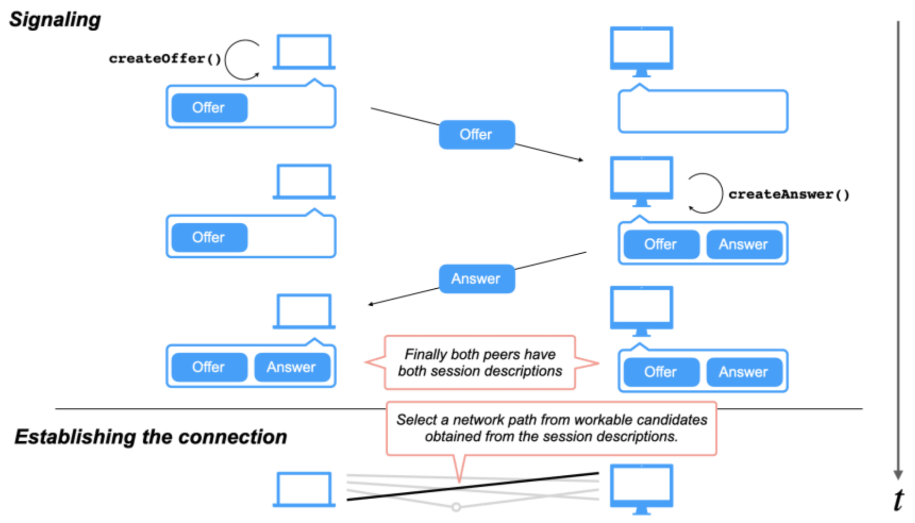

Signaling
createOffer()
Offer
Offer
createAnswer()
Offer
Offer
Answer
Answer
Offer
Answer
Finally both peers have both session descriptions
Offer
Answer
Establishing the connection
Select a network path from workable candidates obtained from the session descriptions.
t

Figure 43 Web-rtc Signaling Graphical Explanation (Copyright Yuichiro Tachibana (Tsuchiya))

After signalling is complete, the browser's WebRTC API and server-side aiortc library automatically start trying to establish a connection. They try to find a workable network path based on the gathered ICE candidates included in the session descriptions. Finally, media streams start to be transmitted.

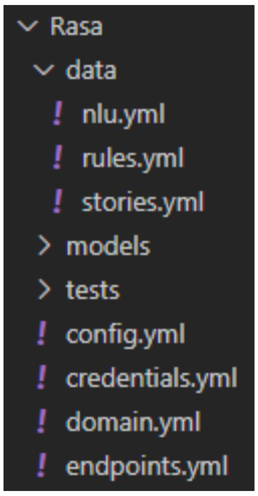
6.10. Chatbot (Rasa)

My Rasa journey began with a deep study of the Rasa Masterclass [66]. It is the starting point where I think every developer should begin, guiding you through all the knowledge and skills to build the desired assistant. The major problem I found from the beginning is that this class was recorded as of version 1.8, and when I was dealing with my project there were 2.5 and 2.6 versions. In spite of the general concepts being the same, a lot of things had changed and I found myself having to rewrite all the code again several times. I have read that they are announcing a 2.x version video of the Masterclass soon yet it has not been released yet. In addition, since syntax changes (not dramatically but indeed noticeably) from version to version, I would suggest to state the version number for the record aside from the documentation, in the forum as it causes some headaches not doing so.

Meanwhile, on May 20th 2020 the "PyData Hamburg Meetup - May the Source be with you" event was held in which Vincent Warmerdam (research and developer advocacy teams at in Rasa) made a Talking/Demo live coding [67] chatbot that allows you to ask anything about Pokémon. I followed along the video and made what will be my starting point chatbot.

6.10.1. Rasa Open Source

This is the directory tree in a typical Rasa project generated by the "rasa init" command. We will cover the most significant files:



```
Rasa
  data
    nlu.yml
    rules.yml
    stories.yml
  models
  tests
  config.yml
  credentials.yml
  domain.yml
  endpoints.yml
```

Figure 44 Rasa Project Directory (Own Copyright)

- Config [68]: the NLU pipeline for inference is specified here. There are components for entity extraction, intent classification, response selection, pre-processing, and more. The NLU config can be adjusted to best fit the project needs; in my case I have tuned it for the Spanish language (I took it from a course ITAINNOVA held about Rasa) and Duckling entity extractor (see 6.3.3). I also adjusted policies for the Duckling entity extractor.
- Domain [69]: this is the core of the bot where intents, slots (remember details provided by the user), responses, forms, and actions are specified. In addition, I have specified two entities: name, which will be the person who Pilar is talking with, and time, which will be the range of the event/s the person is making a query to.

- NLU [70]: NLU training data stores structured information about user messages. This usually includes the user's intent and any entities their message contains. It is recommended 10-15 examples per intent, high-quality data and varied examples.
- Rules [71]: they describe short pieces of conversations that should always follow the same path.
- Stories [72]: representation of a conversation between a user and an AI assistant, converted into a specific format where user inputs are expressed as intents (and entities when necessary), while the assistant's responses and actions are expressed as action names. They can be used to train models that are able to generalize to unseen conversation paths.
- Credentials: this file contains the credentials for the voice & chat platforms (see Integrations).
- Endpoints: this file contains the different endpoints the bot can use.

6.10.2. Rasa X

The challenge with building great AI assistants is that it’s impossible to anticipate everything users might say. The opportunity is that in every conversation, users are telling you exactly what they want. Once an assistant can handle the most important happy path stories, Rasa X is used to improve it, being a tool for Conversation-Driven Development (CDD), the process of listening to users and using those insights to improve your AI assistant.

Rasa X offers a local, server, helm chart and docker compose installation guides depending on scalability needs. ITAINNOVA gave me an Ubuntu virtual machine in a cloud instance with a pre-trained model in Rasa X docker compose installation so that I could start off more easily. I trained the model once successfully for the first time but from the second one on, it all messed up appallingly. I could not simply move the files from my previous Rasa Open Source to Rasa X: there are file formatting templates which make impossible simply copying/pasting or importing from an external repository code that works fine in Rasa Open Source to Rasa X. I started from scratch several times and lost a lot (and when I mean a lot, believe me a lot) of time trying to fix things never achieving a workable version. I was totally desperate trying to fix whatever was causing everything to corrupt in docker compose VMs files, logs, etc. that I ended up installing Rasa X in my local machine. This was also another headache as the official documentation installation guide does not work, yet after some workarounds, I managed to achieve success and had my model trained correctly. In the end I managed to make things work but at the cost of not having that service deployed in a cloud instance: this would have saved computation cost in my local machine and improve general performance.

6.10.3. Rasa Action Server

As explained before, I will run my action server locally. This is the directory tree in a typical Rasa action server. We will cover the most significant files:

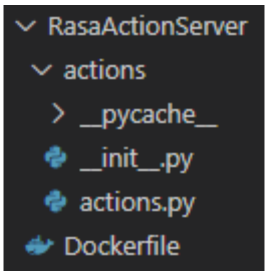


Figure 45 Rasa Action Server Directory (Own Copyright)

- Actions: a custom action can run any code, including API calls, database queries etc.
  - Name function: name of the action itself.
  - Run function: code of the action.
  - Dispatcher: sends response back to user.
  - Tracker: what happens in each dialogue stage.
- Dockerfile: actions can be run from a different machine, especially if the chatbot has been deployed with RasaX in a cloud instance. A detailed guide can be found in [73].
  - Requirements: in case of using external libraries or packages not included by the default python installation, these should be included here specifying the version.

6.10.3.1. Duckling Entity Extractor

Duckling is a Haskell library that parses text into structured data. Depending on the language support, it can extract entities like time, durations, e-mails, phone numbers, temperatures, etc. It is pretty straightforward [74] to implement it.

6.10.4. Integrations

Rasa Open Source provides many built-in connectors [75] to connect to common messaging and voice channels. Using ngrok, with just a single command (ngrok http 5005) you can make a port on your local machine publicly available on the internet. With the free version the limit is 40 connections/minute, which is more than enough. Although seeming complicated at a first glance, integrations just occupy a few lines of code in the application and configuration files, are easy to manage and offer the conversation content information as if it were held in Rasa X. I´ve implemented two kinds:

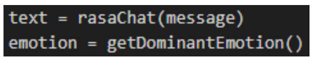
- Telegram´s Bot API allows developers to connect bots to the system. Telegram Bots are special accounts that do not require an additional phone number to set up. These accounts serve as an interface for code running somewhere on the developer´s server (third party applications that run inside Telegram). To use this, the developer needn´t know anything about how Telegram´s MTProto encryption protocol works (their intermediary server will handle all encryption and communication with the Telegram API for developers). Developers communicate with this server via simple HTTPS (interface that offers a simplified version of the Telegram API). Users can interact with bots by sending them messages, commands, and inline requests, offering lots of possibilities.
- Streamlit & Unity embed will work with pre-configured REST channels.

6.11. Pilar (Webservice & Unity)

First, it is important to remark that for this project Pilar is just a frontend for embedding the Rasa chatbot and current emotion expressions querying the database. I have only changed a limited number of lines in its code, not developing anything new.

Pilar avatar is a 5,21 GB Unity project (see Pilar (Avatar Unity)). It is undoubtedly far complex for the two optative 3-credit subjects I have had throughout the degree about modelling & animation and computer graphics. However, Rafa managed to go through the essentials for the current project needs and transfer that knowledge to me. Magic happens with the Animator.SetTrigger, which is responsible of activating an animation trigger, to cause a change in flow in the state machine of an animator controller. This results in that Pilar can express joy, sadness, anger, disgust, fear and other pre-programmed movements for other demos ITAINNOVA has done in the past. This is all in the AnimatorControllerScript class.

This movements expressing emotions are called from the TextBot class, which has a method called WebRequestSuccesHandle that depending on the message intent coming from a web service switches between one or another. Now the C# part is over. The web service is programmed in Python using Flask, a micro web framework. It basically calls /textBot POST method every time an interaction is made with Pilar (talking to her and by being detected and processed with the facial attributes framework). A query to the graph database is made which returns Pilar´s current emotion as well as a REST call is made to Rasa.



```
text = rasaChat(message)
emotion = getDominantEmotion()
```

Figure 46 Rasa & Neo4j Calls from Pilar Web-Service (Own Copyright)

7. Results

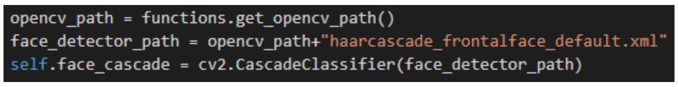
In the end, the only initial objective that has not been achieved is the object detection module. This has mainly been due to a lack of time once the powerful equipment was available to work with, there were other priorities to cover first. For the rest of the objectives, they have been covered in a way that (at least for my personal point of view) surpasses the initial expectations I had of what could someone do with that technology available. Implementing a solution of whatever technology can be more easy or more difficult, but the real challenge is to extract value from it. Squeezing, connecting dots, making new features, making the most out of the different things that are already there without reinventing the wheel. Each module of the final product will be divided and explained in detail.

7.1. Emotion Detection (DeepFace)

As this is something located in the backend, there won´t be included any showcase screenshots to proof it works but later in conjunction with the frontend.

- **Is DeepFace capable of detecting multiple people in a scene?**

Indeed, it is, as seen in the code. However, doing some real-life testing it is difficult to achieve "viewing two different squares" when there are multiple faces to detect. This fact is due to the face detector used:



```
opencv_path = functions.get_opencv_path()
face_detector_path = opencv_path+"haarcascade_frontalface_default.xml"
self.face_cascade = cv2.CascadeClassifier(face_detector_path)
```

Figure 47 DeepFace Real-Time Haarcascade Detector (Own Copyright)

As mentioned in 2.2.1, SSD is the fastest OpenCV built-in face detector. Moreover, RetinaFace is the cutting-edge face detection technology. It can even detect faces in the crowd and it finds facial landmarks including eye coordinates. It is available as a backend in the DeepFace framework as well as SSD yet the real-time implementation code works with haar cascade (I guess this is because speed limitations). I include a proof that multiple face recognition works although not in 100% of the situations:

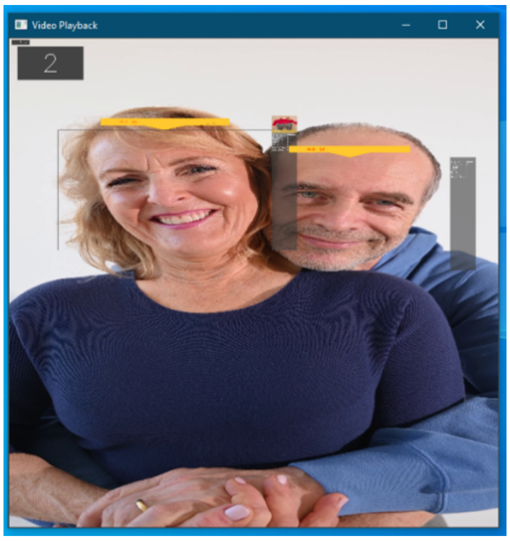
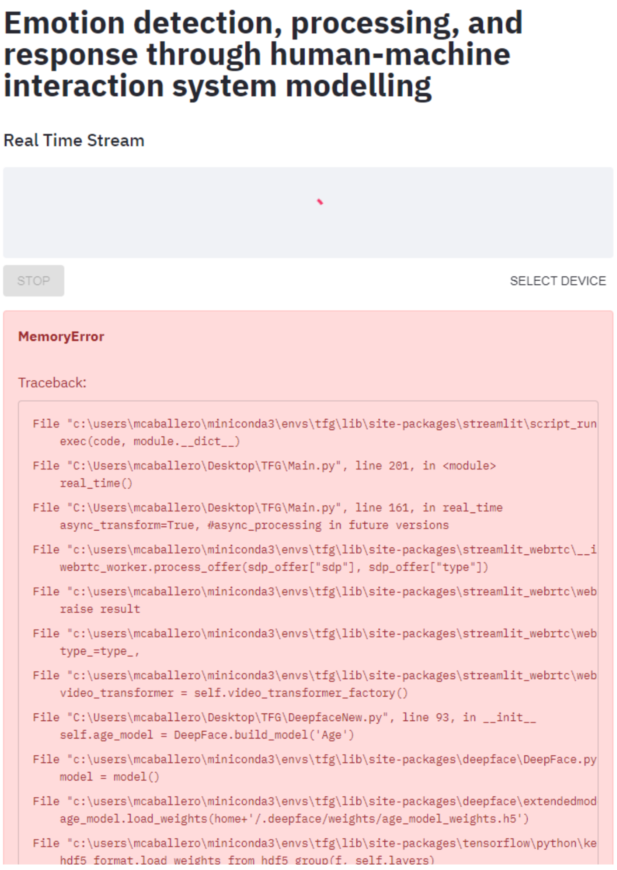


Figure 48 Multiple Face Detection Proof (Own Copyright)

- **How much performance does the application gain running on a GPU?**

The development stage has undergone in three different environments: my 2015 MacBook Pro laptop (slower than a snail), an HP general-purpose enterprise desktop (OK speed) and the GPU server (mind blowing speed). It took 60-90, 30-45 and 3-20 seconds to execute the project in each machine respectively (taking into account project complexity has increased with time). When I first tried executing the project with the GPU server just with the CPU (it runs an Intel Xeon 3.60 GHz 4 cores) it was already fast.


Emotion detection, processing, and response through human-machine interaction system modelling
Real Time Stream
STOP
SELECT DEVICE

MemoryError

Traceback:

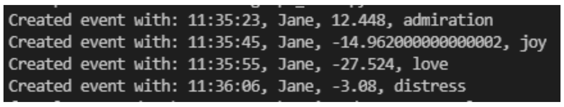
```
File "c:\users\mcaballero\miniconda3\envs\tfg\lib\site-packages\streamlit\script_run
    exec(code, module.__dict__)
File "C:\Users\mcaballero\Desktop\TFG\Main.py", line 201, in <module>
    real_time()
File "C:\Users\mcaballero\Desktop\TFG\Main.py", line 161, in real_time
    async_transform=True, #async_processing in future versions
File "c:\users\mcaballero\miniconda3\envs\tfg\lib\site-packages\streamlit_webrtc\__i
    webrtc_worker.process_offer(sdp_offer["sdp"], sdp_offer["type"])
File "c:\users\mcaballero\miniconda3\envs\tfg\lib\site-packages\streamlit_webrtc\web
    raise result
File "c:\users\mcaballero\miniconda3\envs\tfg\lib\site-packages\streamlit_webrtc\web
    type_=type_,
File "c:\users\mcaballero\miniconda3\envs\tfg\lib\site-packages\streamlit_webrtc\web
    video_transformer = self.video_transformer_factory()
File "C:\Users\mcaballero\Desktop\TFG\DeepfaceNew.py", line 93, in __init__
    self.age_model = DeepFace.build_model('Age')
File "c:\users\mcaballero\miniconda3\envs\tfg\lib\site-packages\deepface\DeepFace.py
    model = model()
File "c:\users\mcaballero\miniconda3\envs\tfg\lib\site-packages\deepface\extendedmod
    age_model.load_weights(home+'/.deepface/weights/age_model_weights.h5')
File "c:\users\mcaballero\miniconda3\envs\tfg\lib\site-packages\tensorflow\python\ke
    hdf5_format.load_weights_from_hdf5_group(f, self.layers)
```

Figure 49 Memory Outage Proof (Own Copyright)

Moving the inference tasks to the GPU has made a dramatical difference in speed and time terms (it can checked in the task administrator if effectively the GPU ramps up when executing the application). The major limitation of the system (aside it is becoming to age compared to last-gen GPU servers available nowadays) is the RAM memory. I sometimes ran into this error while running other applications which made the memory struggle. H5 models are weighty, specially when you have one for face recognition, age and gender.

- **Aside from numbers, is DeepFace really accurate in real life?**

There is no true or false answer here: it depends. Lightning conditions, inclination of the faces, webcam resolution, face mask… and a huge list of elements can influence on the system´s behaviour. Sometimes under ideal conditions I´ve found 100% accuracy (based on my face and the emotions I intended to demonstrate as well as my real age and gender), other times specially when wearing face mask as DeepFace was not designed during a pandemic it always expressed more age and negative emotions and other times when conditions were weird it could mess up a little bit:



```
Created event with: 11:35:23, Jane, 12.448, admiration
Created event with: 11:35:45, Jane, -14.962000000000002, joy
Created event with: 11:35:55, Jane, -27.524, love
Created event with: 11:36:06, Jane, -3.08, distress
```

Figure 50 Emotion Detection Under Unpleasant Circumstances (Own Copyright)

This happened in a challenging demo video were lightning was awful and faces were inclined and overlapped by other elements yet expressing the same emotions throughout the video.

7.2. Interactive Dashboard (Streamlit)

This is the end result of the Streamlit dashboard: on the left side, parameters can be adjusted and loaded into a configuration file that will customize each execution.

- Emotion and mood decay rates can be adjusted from here to observe different behaviours.
- Debug mode skips the database insert to improve speed a little bit.
- Report gnerates a dynamic dataframe which grows in size as more detections are made and displays the data in 4 different kinds of charts.
- Real time or video playback modes.
- In case of selecting video playback, a demo will be selected from a folder containing mainly one kind of emotion.
- A faces database allows to detect a specific person in a real time or video playback.

On the left side, real time stream will be displayed using web-rtc component whereas video playback will be used opening an OpenCV window thread.

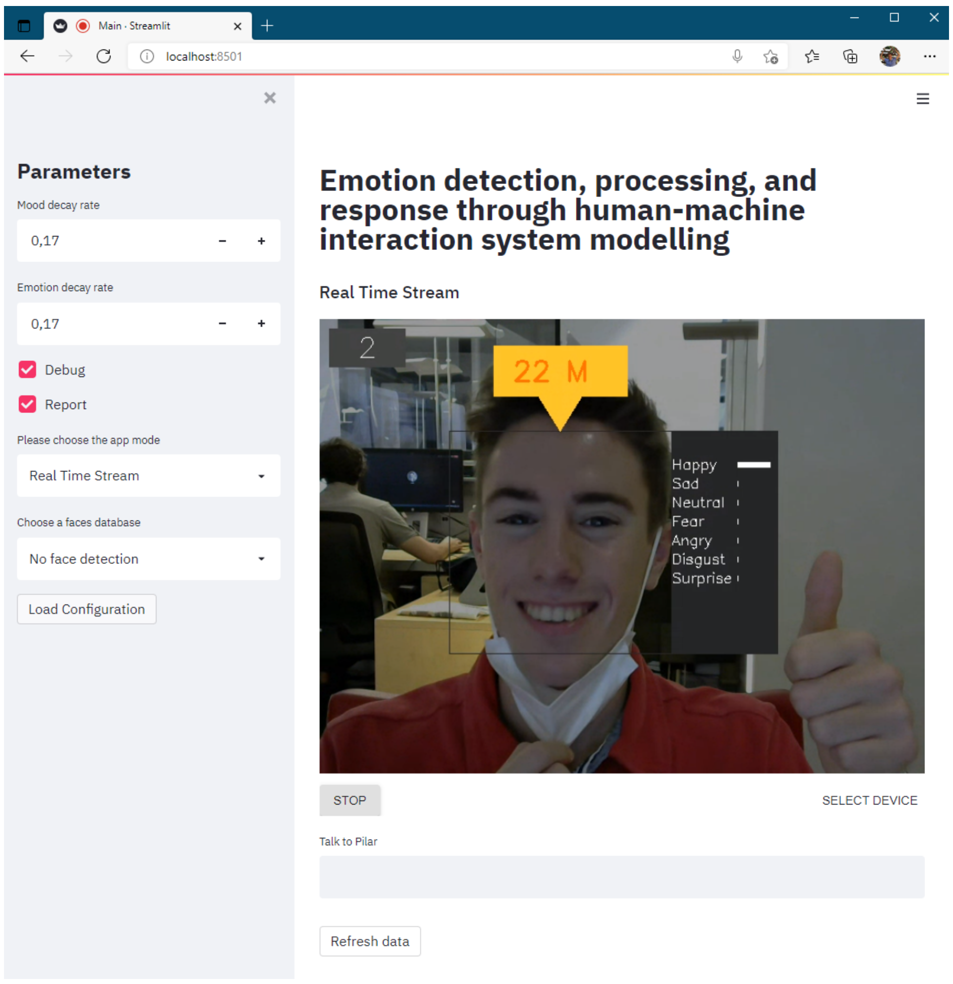
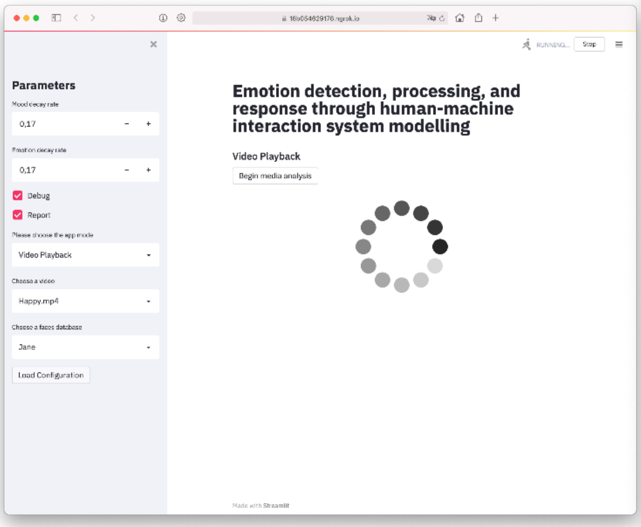


Figure 51 Streamlit Real-Time Stream Sample (Own Copyright)



18b054629178.ngrok.io

RUNNING... Stop

Parameters

Mood decay rate

0,17

Emotion decay rate

0,17

✓ Debug

✓ Report

Please choose the app mode

Video Playback

Choose a video

Happy.mp4

Choose a faces database

Jane

Load Configuration

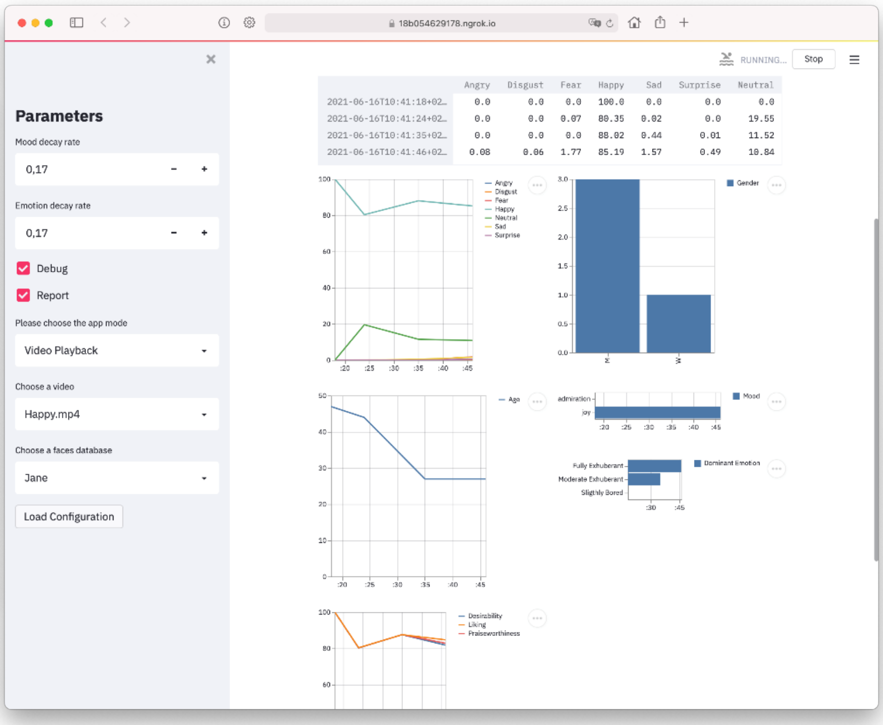
Emotion detection, processing, and response through human-machine interaction system modelling

Video Playback

Begin media analysis

Made with Streamlit

Figure 52 Streamlit Video Playback Sample (Own Copyright)



18b054629178.ngrok.io

RUNNING... Stop

Parameters

Mood decay rate

0,17

Emotion decay rate

0,17

✓ Debug

✓ Report

Please choose the app mode

Video Playback

Choose a video

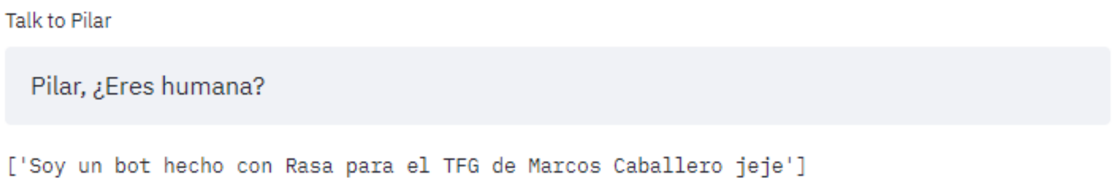
Happy.mp4

Choose a faces database

Jane

Load Configuration

	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
2021-06-16T10:41:18+02…	0.0	0.0	0.0	100.0	0.0	0.0	0.0
2021-06-16T10:41:24+02…	0.0	0.0	0.07	80.35	0.02	0.0	19.55
2021-06-16T10:41:35+02…	0.0	0.0	0.0	88.02	0.44	0.01	11.52
2021-06-16T10:41:46+02…	0.08	0.06	1.77	85.19	1.57	0.49	10.84



Talk to Pilar

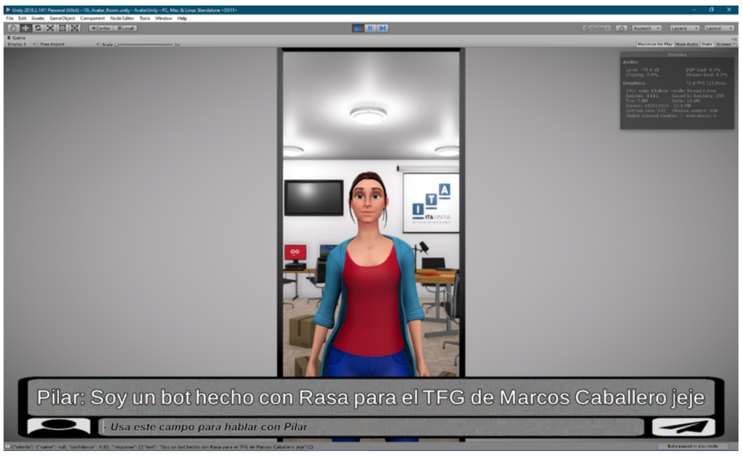
Pilar, ¿Eres humana?

```
['Soy un bot hecho con Rasa para el TFG de Marcos Caballero jeje']
```

Figure 53 Streamlit REST Rasa Chatbot Channel Sample (Own Copyright)

7.3. Pilar (Unity)

This is a sample Unity scene taken from the already made project from which interactions can be made with Pilar. Depending on her current emotion (which depends on who she has seen and what where that person´s emotions), Pilar will make 5 different actions which include: joy, sad, anger, disgust and fear.



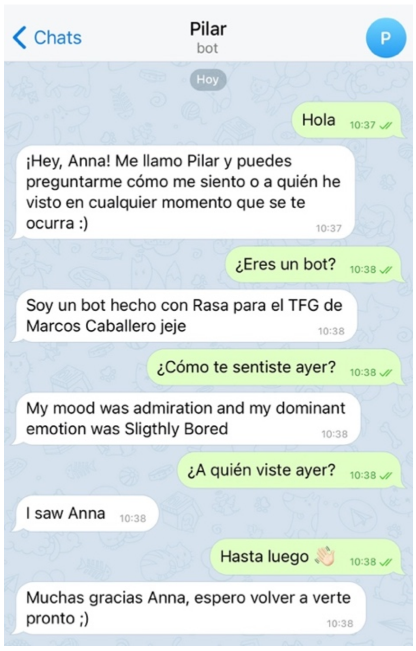
Pilar: Soy un bot hecho con Rasa para el TFG de Marcos Caballero jeje

Usa este campo para hablar con Pilar

Figure 54 Pilar Avatar Unity Sample (Own Copyright)

7.4. Telegram bot

This is a sample conversation in which Pilar sees the previous day a known face in the dataset (with a demo video), captures her emotions and the next day she is asked how she felt and who did she saw the previous day.



Chats
Pilar
bot
Hoy

Hola 10:37

¡Hey, Anna! Me llamo Pilar y puedes preguntarme cómo me siento o a quién he visto en cualquier momento que se te ocurra :) 10:37

¿Eres un bot? 10:38

Soy un bot hecho con Rasa para el TFG de Marcos Caballero jeje 10:38

¿Cómo te sentiste ayer? 10:38

My mood was admiration and my dominant emotion was Sligthly Bored 10:38

¿A quién viste ayer? 10:38

I saw Anna 10:38

Hasta luego 👋 10:38

Muchas gracias Anna, espero volver a verte pronto ;) 10:38

Figure 55 Telegram REST Rasa Chatbot Channel Sample (Own Copyright)

8. Conclusion

Based on all of the above, in accordance with the results obtained through the development of this project, it seems feasible to say that the resulting application meets (and for the Rasa chatbot and Streamlit dashboard exceeds) all the expectations except one (object detection module) and presents a performance and design in accordance defined in the initial requirements.

8.1. Future

8.1.1. Technological point of view

Technical work can continue in the following paths:

- There exist today an ever increasing trend around ontologies and knowledge graphs, becoming the backbone of different systems including semantic search engines, recommendation systems and conversational bots. Inference could be done in the database so Pilar can really understand the concept of a person or certain objects and their relations simulating a real brain.
- Following the last approach, one of the downsides of the Neo4j database is that every time a person is recognized, to Pilar´s eyes it is a different person no matter that it matches a previously recognized person name in the database. To stablish a relation, unsupervised learning and clustering can be applied within the database as done in this notebook. The result would look something similar like this:

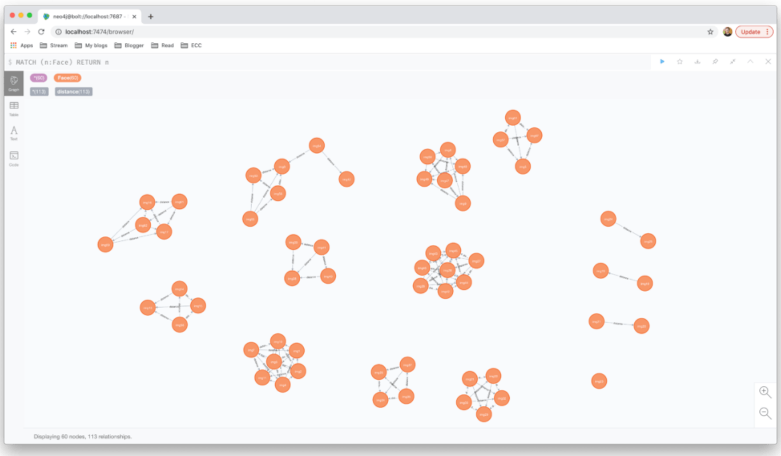


Figure 56 People Relationship Graph Inference (Copyright Sefik Ilkin Serengil)

- Time is already modelled in this graph database schema. However, LiDAR technology has proven to be very evolved and 3D mapping a space is something not as expensive as it was a few years ago with some projects like Intel RealSense. This technology could be exploited to more precisely understand the outside world as stated in these (one and two) Neo4j website articles.
- It would have been nice that Sefik Ilkin Serengil would have implemented ArcFace and RetineFace for facial detection and recognition tasks in real time. It would dramatically improve results in scenes with a lot of faces.
- RasaLit is a collection of helpful viewers that help with understand Rasa NLU components. Some of these views are made using streamlit, hence the wink in the name. It would be of of great help to implement it in case of having a bigger chatbot with more user stories, NLU data and functionalities in general.
- The UI for the Rasa chatbot implemented in Streamlit is, whichever way you look at it, very simplistic. Dialogflow provides a component for Streamlit that looks really pleasing to the eye. I tested it out and tried to make it work with Rasa but didn´t succeed. It would be interesting to see something similar made for Rasa.

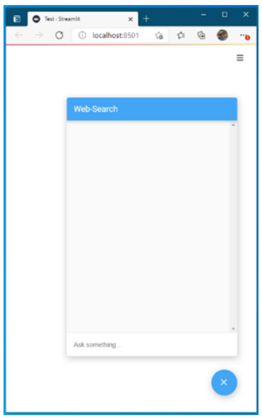


Figure 57 Dialogflow Streamlit Component (Own Copyright)

- Concurrency can be implemented in order to optimize the existing code in such tasks that can be developed in an asynchronous way, such as inserting events into the database while real-time face attributes or object detection is being performed.
- More modules can be added to the Event "box" such as video-speech recognition. This is actually a very difficult task that has been covered in the State of Art Section, resulting in very few papers being able to achieve it providing the code without having to retrain the whole CNN by yourself or having to translate it from another framework, requiring a big amount of work. Work that Sefik Ilkin Serengil has done to develop his Deepface framework for the vision-only side, which is highly admirable.
- Other computational emotion modelling papers could be used to improve existing ALMA such as "Adding the emotional dimension to scripting character dialogues" [4] from the same author or even changing to a more modern more complex paper. See also FATIMA toolkit [76].
- Biometric Mirror [77] is project developed by the University of Melbourne which aims to prepare for an AI future from an ethical point of view. It can detect a range of data from people's faces, including demographic, cultural and psychological attributes. The downside about AI algorithms is that they can be flawed or biased and they need to be used in an ethical way. See also a micro-documentary about this.

8.1.2. Strategical/Business point of view

Given the final results of the project, multiple expansion options are available:

- Sell this project as an only asset hiring Unity, DevOps, Customer Relationships and Software Engineer experts to send this to companies as a traditional receptionist replacement or information point.
- Expand Rasa module and sell it as a customer service chatbot embedded as a web or Telegram experience maybe implementing some NLP sentiment analysis techniques.
- Use the emotion, gender, ethnicity and age predictions to generate new assets. For instance, calculating a life insurance price estimation based on these and other indicators (e.g. smoker, mental health, etc.). Another example, make a human resources tool aiding online job interviews or automating them without the need for a real person to be handling the interview.
- Use the Streamlit dashboard to monitor down-syndrome patients reactions to certain mental stimulus.

8.2. Personal Opinion

Honestly, my personal and academic learning outcomes of this project have been more than expected. In the academic side I have enjoyed reading tons of Data Science and AI literature, which has provided me with a very interesting general culture (adding to my previous background) about the field. As well as that, developing a solution is not the same as imagining how it could be developed: a myriad of problems arises and you have to deal with each of them. The most exciting part for me has been feeling autonomous in overcoming difficulties and the self-learning ability I have developed along the degree and more sharply along the final degree project.

As for the personal side, I like challenges in my life and this has been a full-fledged challenge ☺. I have discovered my best side, getting to know new capabilities I didn´t know I had regarding hard work and focus. However, "all that glitters is not gold". I have had my ups and downs: from my experience and point of view, it is hard to argue that the pandemic has made the world a colder place and has objectively done damage to society in general. This fact sometimes led me to demotivate and give less of myself. Yet on balance, bad things always have a silver lining and on account of this circumstances and I can say in the end I have eagerly tackled these conflicts to make a better version of myself.

To sum up, the technologies developed here are cutting-edge. This project has widely contributed to my ML career path and I would love to continue developing AI, Data Science, Business Intelligence, Computer Vision, NLP, etc. applications for ITAINNOVA.

9. Annexes

9.1. Pilar (Avatar Unity)

Unity is a is a cross-platform game engine released in 2005. It supports thousands of developers to develop videogames, simulation AR & VR scenes and avatars, among hundreds of possibilities more. Pilar project arises from a collaboration between ITAINNOVA and Imascono three years ago in which they intended to investigate real life applications of avatars using different technologies. They built Pilar just for that purpose, providing "her" with lots of capabilities. Now Eva has replaced Pilar and it is an evolution over her being just an Imascono asset.

9.2. Logbook

The project can be divided into three stages match the summer holiday, academic year and internship periods. It is worth mentioning what facts make one phase different from another for the reader to be more aware of the circumstances in each moment and therefore better understand of the project in general.

9.2.1. Summer – October 2020

The initial project scope stated before was thought when the Summer ended. I wanted to work in a company to fulfil my curricular practice subject for the next academic year but one of the negative effects of the pandemics was that companies were not hiring any interns. In consequence I contacted mostly all my contacts and asked if I could use some help to stay busy this summer. My tutor Rafa proposed me to start working in the final degree project doing some research work and documentation to take advantage of the early start and decide in which computer science knowledge areas I would like to put more focus on. He gave me 5 different alternatives: prediction of plagues in fields with satellite images, predictive maintenance in Industry 4.0, improvement of the Pilar assistant so that she could feel by continuing the work of Martín and a European project on cancer prediction with images.

Everything sounded marvellous yet there was a big BUT: all these projects depended somehow on ITAINNOVA and I was not guaranteed a final degree project internship in October because of the pandemics. The only project that could be finished in case I couldn´t work in ITAINNOVA was the improvement of as instead of her Unity Project I could simply use a bunch of emojis to represent the emotions model output. I was then suggested by Rafa that until October we could work together figuring out the best way to focus the final degree project scope. The project began to take shape in my mind by getting some culture analysing a lot of interesting papers and by looking into how AI could integrate into a system´s architecture looking the code some papers came available with.

When the Summer ended and the Autumn started along with the new academic year, somehow everything made sense and was tied up in a way that it could be presented to a tribunal as a final degree project targeting a high mark. The first checklist of objectives was presented here. As so, I was assigned an academic (Jesús Carro) and a company tutor (Rafa) so I was covered in all the different scenarios that could result if I could manage to work in ITAINNOVA or not.

9.2.2. October 2020 – April 2021

Rafa made a big effort with ITAINNOVA Human Resources department to hire me because there I would be provided with a very powerful computer. However, the sanitary situation and the resulting human resources policy did not allow for the incorporation of new personnel for internships nor PFGs. In May, the first call for FDP internships was published and I also had the subsequent job interview.

9.2.3. April – June 2021

Fortunately, I was hilarious to hear the news that I was finally joining ITAINNOVA in April. The greatest effort was put into this project stage (yet not undermining the rest) because I immediately began to notice how my work capacity and motivation were taking off disposing of a specific workplace for my FDP, cooler tools and an economic incentive.

References

[1] B. O. Ahmed Banafa, "What is affective computing?," [Online]. Available: https://www.bbvaopenmind.com/en/technology/digital-world/what-is-affective-computing/. [Accessed 14 June 2021].

[2] M. Cativiela, Caso de uso (from his Final Degree Project).

[3] P. Gebhard, ALMA: A Layered Model of Affect, Conference Paper, Deutsches Forschungszentrum für Künstliche Intelligenz, 2005.

[4] P. Gebhard, M. Kipp, M. Rumpler and T. Rist, Adding the Emotional Dimension to Scripting Character Dialogues, Conference Paper in Lecturer Notes in Computer Scienc, 2003.

[5] Neo4j, "Neo4j Graph Database Use Case: Knowledge Graph," [Online]. Available: https://neo4j.com/use-cases/knowledge-graph/. [Accessed 14 June 2021].

[6] Y. Zhang, Z.-R. Wang and J. Du, Deep Fusion: An Attention Guided Factorized Bilinear Pooling for Audio-Video Emotion Recognition, Hefei: National Engineering Laboratory for Speech and Language Information Processing, University of Science and Technology of China, 2019.

[7] F. Schroff, D. Kalenichenko and J. Philbin, FaceNet: A Unified Embedding for Face Recognition and Clustering, Google Inc., 2015.

[8] Y. Taigman, M. Yang, M. A. Ranzato and L. Wolf, DeepFace: Closing the Gap to Human-Level Performance in Face Verification, Facebook AI Research Menlo Park, Tel Aviv University, CA, Tel Aviv, 2014.

[9] T. Baltrusaitis, P. Robinson and L.-P. Morency, OpenFace: an open source facial behavior analysis toolkit, 2015.

[10] HuffPost, "Facebook's New 'DeepFace' Program Is Just As Creepy As It Sounds," [Online]. Available: https://www.huffpost.com/entry/facebook-deepface-facial-recognition_n_4985925. [Accessed 14 June 2021].

[11] O. M. Parkhi, A. Vedaldi and A. Zisserman, Deep Face Recognition, Oxford: Visual Geometry Group, Department of Engineering Science, University of Oxford, 2015.

[12] Y. Sun, X. Wang and X. Tang, Deep Learning Face Representation by Joint Identification-Verification, Hong Kong: Department of Information Engineering, The Chinese University of Hong Kong, 2014.

[13] J. Deng, J. Guo and N. Xue, ArcFace: Additive Angular Margin Loss for Deep Face Recognition, London: Imperial College London, 2019.

[14] M. Grootendorst, "9 Distance Measures in Data Science | Towards Data Science," [Online]. Available: https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa. [Accessed 14 June 2021].

[15] S. I. Serengil, "deepface/Fine-Tuning-Threshold.ipynb at master | Github," [Online]. Available: https://github.com/serengil/deepface/blob/master/tests/Fine-Tuning-Threshold.ipynb. [Accessed 14 June 2021].

[16] S. I. Serengil, "Facial Expression Recognition with Keras," [Online]. Available: https://sefiks.com/2018/01/01/facial-expression-recognition-with-keras/. [Accessed 14 June 2021].

[17] O. A, C. G. L. and C. A, The Cognitive Structure of Emotions, Cambridge, MA: Cambridge University Press, 1988.

[18] Neo4j, "What is a Graph Database? - Developer Guides," [Online]. Available: https://neo4j.com/developer/graph-database/. [Accessed 14 June 2021].

[19] Infoq, "Data Modeling in Graph Databases: Interview with Jim Webber and Ian Robinson," [Online]. Available: https://www.infoq.com/articles/data-modeling-graph-databases/. [Accessed 14 June 2021].

[20] Neo4j, "Graph Modeling Guidelines - Developer Guides," [Online]. Available: https://neo4j.com/developer/guide-data-modeling/. [Accessed 14 June 2021].

[21] Slideshare, "Modelling Event Data As A Graph," [Online]. Available: https://es.slideshare.net/dilyand/modelling-event-data-as-a-graph. [Accessed 14 June 2021].

[22] M. Needham, "Neo4j: Cypher - Creating a time tree down to the day," [Online]. Available: https://www.markhneedham.com/blog/2014/04/19/neo4j-cypher-creating-a-time-tree-down-to-the-day/. [Accessed 14 June 2021].

[23] Rasa, "Open source natural language processing (NLP)," [Online]. Available: https://rasa.com/solutions/open-source-nlu-nlp/. [Accessed 14 June 2021].

[24] Rasa, "Tuning Your NLU Model," [Online]. Available: https://rasa.com/docs/rasa/tuning-your-model%22%20%5Cl%20%22how-to-choose-a-pipeline. [Accessed 14 June 2021].

[25] S. Poria, D. Hazarika, N. Majumder and M. Rada, Beneath the Tip of the Iceberg: Current Challenges and New Directions in Sentiment Analysis Research, Singapore, Michigan: Information systems Technology and Design, Singapore University of Technology and Design; School of Computing, National University of Singapore; Electrical engineering and Computer Science, University of Michigan, 2020.

[26] S. Poria, N. Majumder, R. Mihalcea and E. Hovy, Emotion Recognition in Conversation: Research Challenges, Datasets, and Recent Advances, Singapore University of Technology and Design; Instituto Tecnológico Nacional, Mexico City; Computer Science and Engineering, University of Michigan; Language Technologies Institute, Carnegie Mellon University.

[27] V. Paradigm, "What is Agile Software Development? [Quick Guide]," [Online]. Available: https://www.visual-paradigm.com/guide/agile-software-development/what-is-agile-software-development/. [Accessed 14 June 2021].

[28] Infoq, "Agile Development Applied to Machine Learning Projects," [Online]. Available: https://www.infoq.com/articles/machine-learning-agile/. [Accessed 14 June 2021].

[29] V. Paradigm, "Comprehensive Scrum Guide," [Online]. Available: https://www.visual-paradigm.com/scrum/what-is-scrum/. [Accessed 14 June 2021].

[30] Indeed, "¿Cuánto se gana en Zaragoza, Zaragoza provincia de Programador/a junior?," [Online]. Available: https://es.indeed.com/career/programador-junior/salaries/Zaragoza--Zaragoza-provincia?from=top_sb. [Accessed 14 June 2021].

[31] Indeed, "¿Cuánto se gana en Zaragoza, Zaragoza provincia de Data scientist?," [Online]. Available: https://es.indeed.com/career/data-scientist/salaries/Zaragoza--Zaragoza-provincia. [Accessed 14 June 2021].

[32] V. Driessen, "A successful Git branching model | nvie," [Online]. Available: https://nvie.com/posts/a-successful-git-branching-model/. [Accessed 14 June 2021].

[33] S. U. o. T. a. Design, "Emotion Recognition in Conversations," [Online]. Available: https://github.com/declare-lab/conv-emotion. [Accessed 14 June 2021].

[34] S. U. o. T. a. Design, "Awesome Emotion Recognition in Conversations," [Online]. Available: https://github.com/declare-lab/awesome-emotion-recognition-in-conversations. [Accessed 14 June 2021].

[35] S. Shahriar, "Emotion Recognition," [Online]. Available: https://github.com/sadat1971/Emotion_Recognition. [Accessed 14 June 2021].

[36] P. w. Code, "Computer Vision Emotion Recognition," [Online]. Available: https://paperswithcode.com/task/emotion-recognition. [Accessed 14 June 2021].

[37] P. w. Code, "Audio + Video Emotion Recognition," [Online]. Available: https://paperswithcode.com/search?q_meta=&q=audio+video+emotion+recognition. [Accessed 14 June 2021].

[38] S. I. Serengil, "Github DeepFace: A Lightweight Face Recognition and Facial Attribute Analysis Framework for Python," [Online]. Available: https://github.com/serengil/deepface. [Accessed 14 June 2021].

[39] S. I. Serengil, "YouTube DeepFace: A Lightweight Face Recognition and Facial Attribute Analysis Framework for Python," [Online]. Available: https://www.youtube.com/watch?v=WnUVYQP4h44&list=PLsS_1RYmYQQFdWqxQggXHyn P1rqaYXv_E&index=2.

[40] NVIDIA, "What Is CUDA | NVIDIA Official Blog," [Online]. Available: https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/. [Accessed 14 June 2021].

[41] A. Vidhya, "Solution to TensorFlow 2 not using GPU," [Online]. Available: https://medium.com/analytics-vidhya/solution-to-tensorflow-2-not-using-gpu-119fb3e04daa. [Accessed 14 June 2021].

[42] Google, "Tensorflow: Configuraciones de compilación probadas, GPU," [Online]. Available: https://www.tensorflow.org/install/source_windows#gpu. [Accessed 14 June 2021].

[43] A. Sehgal, "Kaggle Forum: GPU not detected when upgrading to tensorflow 2.4.0," [Online]. Available: https://www.kaggle.com/discussion/208757. [Accessed 14 June 2021].

[44] Eda, "experimental_list_devices attribute missing in tensorflow_core._api.v2.config," [Online]. Available: https://stackoverflow.com/questions/60581677/experimental-list-devices-attribute-missing-in-tensorflow-core-api-v2-config. [Accessed 14 June 2021].

[45] S. I. Serengil, "DeepFace Github GPU Issues Forum," [Online]. Available: https://github.com/serengil/deepface/issues?q=gpu. [Accessed 14 June 2021].

[46] A. Tinkerpop, "Getting Started," [Online]. Available: https://tinkerpop.apache.org/docs/current/tutorials/getting-started/. [Accessed 14 June 2021].

[47] Neo4j, "Neo4j Python Driver 4.2," [Online]. Available: https://neo4j.com/docs/api/python-driver/current/. [Accessed 14 June 2021].

[48] N. Training, "Graph Data Modeling for Neo4j," [Online]. Available: https://neo4j.com/graphacademy/training-gdm-40/enrollment/. [Accessed 14 June 2021].

[49] N. D. Guides, "Graph Modeling Guidelines," [Online]. Available: https://neo4j.com/developer/guide-data-modeling/. [Accessed 14 June 2021].

[50] M. Needham, "Neo4j: Cypher - Creating a time tree down to the day," [Online]. Available: https://www.markhneedham.com/blog/2014/04/19/neo4j-cypher-creating-a-time-tree-down-to-the-day/. [Accessed 14 June 2021].

[51] N. C. Manual, "Clauses," [Online]. Available: https://neo4j.com/docs/cypher-manual/current/clauses/. [Accessed 14 June 2021].

[52] N. D. Guides, "Dates, datetimes, and durations," [Online]. Available: https://neo4j.com/developer/cypher/dates-datetimes-durations/. [Accessed 14 June 2021].

[53] N. C. Manual, "Temporal (Date/Time) values, Accessing components of temporal instants," [Online]. Available: https://neo4j.com/docs/cypher-manual/current/syntax/temporal/#cypher-temporal-accessing-components-temporal-instants. [Accessed 14 June 2021].

[54] Neo4j, "Neo4j Cypher Refcard 4.2," [Online]. Available: https://neo4j.com/docs/cypher-refcard/current/. [Accessed 14 June 2021].

[55] M. S. Madsen, "Awesome Streamlit," [Online]. Available: https://github.com/MarcSkovMadsen/awesome-streamlit. [Accessed 14 June 2021].

[56] zhaoooyue, "Streamlit Forum: How to save all widget state?," [Online]. Available: https://discuss.streamlit.io/t/how-to-save-all-widget-state/1764. [Accessed 14 June 2021].

[57] M. Winter, "Streamlit Forum: Vertically center the content of columns," [Online]. Available: https://discuss.streamlit.io/t/vertically-center-the-content-of-columns/6163. [Accessed 14 June 2021].

[58] T. Teixeira, "Github Gist: SessionState.py," [Online]. Available: https://gist.github.com/tvst/036da038ab3e999a64497f42de966a92. [Accessed 14 June 2021].

[59] A. Streamlit, "How to use Streamlit with VS Code," [Online]. Available: https://awesome-streamlit.readthedocs.io/en/latest/vscode.html. [Accessed 14 June 2021].

[60] Lottie, "LottieFiles: Loading Animations," [Online]. Available: https://lottiefiles.com/animation/loading. [Accessed 14 June 2021].

[61] Y. T. (Tsuchiya), "New Component: streamlit-webrtc, a new way to deal with real-time media streams," [Online]. Available: https://discuss.streamlit.io/t/new-component-streamlit-webrtc-a-new-way-to-deal-with-real-time-media-streams/8669. [Accessed 14 June 2021].

[62] Y. T. (Tsuchiya), "Building a Web-Based Real-Time Computer Vision App with Streamlit," [Online]. Available: https://dev.to/whitphx/build-a-web-based-real-time-computer-vision-app-with-streamlit-57l2. [Accessed 14 June 2021].

[63] Y. T. (Tsuchiya), "WebRTC demo," [Online]. Available: https://share.streamlit.io/whitphx/streamlit-webrtc-example/main/app.py. [Accessed 14 June 2021].

[64] Y. T. (Tsuchiya), "Github streamlit-webrtc/app.py," [Online]. Available: https://github.com/whitphx/streamlit-webrtc/blob/main/app.py. [Accessed 14 June 2021].

[65] Y. T. (Tsuchiya), "Developing a streamlit-webrtc component for real-time video processing," [Online]. Available: https://towardsdatascience.com/developing-a-streamlit-webrtc-component-for-real-time-video-processing-5d4c07405c4d. [Accessed 14 June 2021].

[66] Rasa, "Rasa Masterclass: Developing Contextual AI assistants with Rasa tools," [Online]. Available: https://www.youtube.com/watch?v=rlAQWbhwqLA&list=PL75e0qA87dlHQny7z43NduZHPo6qd-cRc.

[67] V. Warmerdam, "PyData Hamburg May '20 - Vincent Warmerdam: An Open-Source Chatbot made with Rasa," [Online]. Available: https://www.youtube.com/watch?v=6D8NvhzNMkQ. [Accessed 14 June 2021].

[68] Rasa, "Tuning Your NLU Model," [Online]. Available: https://rasa.com/docs/rasa/tuning-your-model. [Accessed 14 June 2021].

[69] Rasa, "Domain," [Online]. Available: https://rasa.com/docs/rasa/domain/. [Accessed 14 June 2021].

[70] Rasa, "NLU Training Data," [Online]. Available: https://rasa.com/docs/rasa/nlu-training-data/. [Accessed 14 June 2021].

[71] Rasa, "Rules," [Online]. Available: https://rasa.com/docs/rasa/rules/. [Accessed 14 June 2021].

[72] Rasa, "Stories," [Online]. Available: https://rasa.com/docs/rasa/stories/. [Accessed 14 June 2021].

[73] M. Caballero, Servidor de Acciones Rasa.

[74] Rasa, "Using Duckling for Entity Extraction with Rasa | Rasa Tutorial," [Online]. Available: https://www.youtube.com/watch?v=MDF-4LDvn0M. [Accessed 14 June 2021].

[75] Rasa, "Connecting to Messaging and Voice Channels," [Online]. Available: https://rasa.com/docs/rasa/messaging-and-voice-channels/. [Accessed 14 June 2021].

[76] F. Toolkit, "Fatima Toolkit," [Online]. Available: https://fatima-toolkit.eu/. [Accessed 14 June 2021].

[77] N. Wouters, "Biometric Mirror," [Online]. Available: https://socialnui.unimelb.edu.au/research/biometric-mirror/campus-displays/. [Accessed 14 June 2021].