UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

ANALYSIS AND DEVELOPMENT OF A PERSONAL DASHBOARD FOR EMOTION TRACKING

CARLOS MARZAL ROMÓN

ENERO 2019

TRABAJO DE FIN DE GRADO

Título:	Análisis y Desarrollo de una Interfaz Personal para la Gestión de Emociones
Título (inglés):	Analysis and Development of a Personal Dashboard for Emotion Tracking
Autor:	Carlos Marzal Romón
Tutor:	Carlos A. Iglesias Fernández
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

ANALYSIS AND DEVELOPMENT OF A PERSONAL DASHBOARD FOR EMOTION TRACKING

Carlos Marzal Romón

Enero 2019

Resumen

En esta memoria se describirá mi proyecto al desarrollar una interfaz para el análisis de emociones personales mediante reconocimiento facial de expresiones y la cuenta personal de Twitter.

Para hacer esta interfaz personal se ha utilizado un componente llamado Clmtrackr para, utilizando la cámara principal del ordenador, analizar los gestos de la cara y de esa manera intentar aproximar las emociones sentidas en ese momento. La información se guarda usando Elasticsearch para luego ser interpretada en las gráficas de Google Charts con la visualización en Sefarad.

El otro método de análisis de emociones que usaremos será con los tweets particulares del usuario. Para esto hemos usado librerías de Python como Tweepy para recuperar los tweets y Senpy como analizador de sentimientos y emociones de dichos textos. Toda esta información será guardada en un índice de Elasticsearch y representada en gráficas y elementos visuales para el manejo de datos.

Al final queremos conseguir una aplicación que nos permita ver nuestros cambios en emoción usando distintos métodos. Estos datos pueden ser usados en conjunto con otras aplicaciones del manejo de emociones para conseguir un resultado más preciso.

Palabras clave: Emociones, Sentimientos, Cámara, Análisis, Video, Clmtrackr, Twitter, Personal, Interfaz.

Abstract

This document will describe my project which is developing an interface for the analysis of personal emotions through facial emotion recognition and our personal twitter feed.

To create the main aspect of this dashboard we have used a component called Clmtrackr which, using the main camera in our device, analyses our face gestures and tries to approximate it to an emotion. This information is then stored through Elasticsearch and interpreted in graphs and charts from Google Charts with visualization in Sefarad.

Our other method to emotion analysis will be using the tweets of the user. For this we have used Python libraries such as Tweepy to capture and store the tweets and Senpy to analyze the emotion and sentiment of said texts. All this information will be stored in and Elasticsearch index and represented in graphs and visual elements for data managing.

Our end product should be an application that lets us observe our change in emotion using different methods. This data could be then used in conjunction with other emotion software to get a more precise result.

Keywords: Emotions, Sentiments, Camera, Analysis, Video, Clmtrackr, Twitter, Personal, Dashboard.

Agradecimientos

Quería agradecer antes de nada a mi familia por apoyarme en todas mis decisiones y ayudarme siempre que lo necesito, en especial a mis padres Luis y Mercedes, a mis hermanos Daniel, Cristina y Álvaro y a mi abuela Marichu.

Muchas gracias a mi tutor Carlos por darme la oportunidad de hacer este proyecto y a los compañeros del GSI por orientarme y ayudarme con el desarrollo de mi TFG. También a todos los compañeros que me han ayudado a realizar las pruebas del trabajo.

Muchas gracias a todos.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

Emotion is something basic in everyone's live and trying to know how someone is feeling has been a great challenge for years since technology took a great part in our lives.

Facial expression is a great way to know how someone is feeling at a certain moment. Charles Darwin [1] reached the conclusion that expression and gestures were developed by humans as part of our development as a species, as it is shown in his research The Expression of the Emotions in Man and Animals, 1872 [2]. This evidence was reached after asking the same 16 questions to people living in 8 parts of the world: Africa, America, Australia, Borneo, China, India, Malaysia and New Zealand. From the answers they deduced that the facial expressions used by people in all these parts of the world where the same for each emotion; which shows how our gestures are something natural for human beings and not changed by cultures.

The fact that expressions are global, and everyone around the world will have the same face gestures when showing certain emotions make a facial recognition system a great way of figuring how any person is feeling at a given time [3]. This is great because it gives us valid results from people coming from any place on Earth. For this reason, video facial

recognition has been decided as the the central point of this project.

Of course, there has to be some contrast on information received from the video evidence by some other sort of emotion detection. That's why another source of detection is implemented: the written analysis, and since personal networks are such a big part of our lives nowadays, where everyone likes to share their thoughts and feelings [4], tweets from the user will be the way in which this is done. They are analyzed, and an emotion and sentiment is taken from them.

All the data gathered from both these methods will be interactively shown on screen using widgets and graphs based mostly on Google Charts adapted to be able to get the data from an Elasticsearch search query.

1.2 Project goals

The main goal of this project is to create a personal dashboard [5] that will track the change in emotion in an individual. Data will be shown through different forms, so it can be used for different purposes.

The dashboard will record your face using the computer's main camera and show it in screen, highlighting the main zones of your face. Using the JavaScript library Clmtrackr[1] with some modifications, your emotion will be calculated from your face and be shown on screen. This emotion analyzer will then store the information gathered in Elasticsearch to then be represented by graphs. This data can then be used to analyze the change in emotion in someone while they are watching or reading something.

The second section of the personal dashboard will run around someone's personal Twitter[2]. The intention here is to have a secondary source of emotion detection that centers around written text, so the idea is to analyze personal tweets to figure out the sentiments in that moment and track the change in emotion over time from what you write.

1.3 Project tasks

To achieve all these goals, we have a number of tasks to complete during the project. These tasks initially worked as a guide and a Trello board was be made to keep track of what is left to do.

[1] https://github.com/auduno/clmtrackr
[2] https://twitter.com/

The tasks are the following:

- Study the software Clmtrackr from Audino and think of possible improvements for adapting it to our dashboard.
- Create the main dashboard using Sefarad and have Elasticsearch run when starting the software.
- Create the Elasticsearch indexes that will be used.
- Create a Polymer element which contains the video tracking from Clmtrackr and have it working through the JavaScript in Polymer.
- Store the data from the Clmtrackr element into an Elasticsearch index.
- Represent all the data in various charts.
- Collection of tweets from a specific user.
- Classify all these tweets into various emotions and sentiments.
- Store all this information in Elasticsearch.
- Represent the Twitter data in various charts.

1.4 Structure of this document

This section provides a brief overview of the chapters included in this document. The structure is as follows:

Chapter 1 is a small resume of what and why we plan to do in this project.

Chapter 2 shows the main technologies hat where used in the development of the project.

Chapter 3 gives an insight of the architecture of the project and how the enabling technologies mentioned in the previous chapter where used to make everything work together.

Chapter 4 presents the experiment made to demonstrate the project has relevant information.

Chapter 5 discusses thee conclusions, with some insight in the problems faced and some possible improvements that could be made.

CHAPTER 2

Enabling Technologies

This chapter explains and gives an insight into the main techniques used in the project. The most important element of this personal dashboard resolves around polymer web components, which basically is the visual software library that helps us integrate all the elements together. The other main component of the project is Elasticsearch which is our main source of storing and accessing data. There are several other software components which were used in the creation of the dashboard which we'll obviously comment on.

2.1 Sefarad

Sefarad[1] is a software environment developed by the GSI at ETSIT-UPM which is used to visualize and update data from Elasticsearch.

For the **visualization** of the software, Sefarad uses **Polymer Web Components**[2]. This is an open source JavaScript library developed by Google which bases its functionality in the implementation of different elements in a main webpage.

[1] https://sefarad.readthedocs.io
[2] https://www.webcomponents.org/

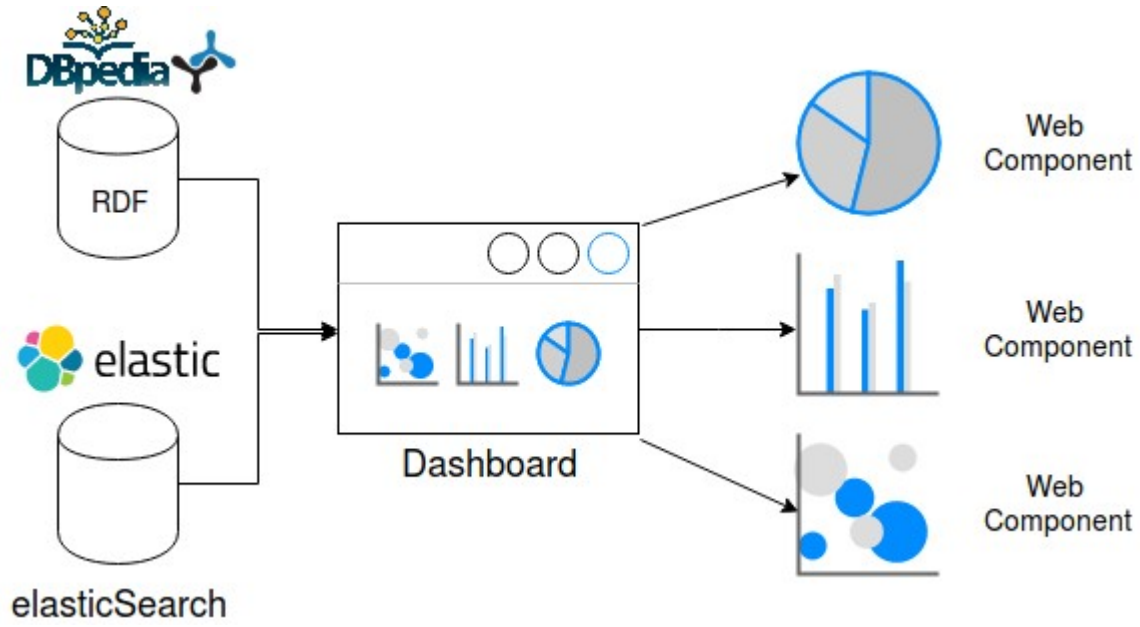

Figure 2.1: Sefarad architecture

You can use elements already created by other people or create elements yourself and implement them together. Creating elements where the code doesn't have to be in the main HTML/JavaScript document makes building complex applications much easier and efficient. Also, you can create elements which could be used in your future project without the need of having to find the exact part of the code that implements that element.

The programming languages used by Polymer web components are the standard web technologies: HTML, JavaScript and CSS for the style, with some modifications to the JavaScript used to define an element.

We used elements that where already created by modifying some thing and created a new one called Clmtrackr that is mentioned in further sections. The already created elements used in this project are:

2.1.1 Google Chart

This is an element created to represent different Google charts[3] from aggregations coming from data in an Elasticsearch index.

The elements have various inputs to select the name of the aggregation and some specific data for Google charts. The inputs that the element allow are the following:

[3]https://developers.google.com/chart/

- **Data:** A JSON with the data received from Elasticsearch. This in our case will be the variable in which the results from the search queries where stored in.
- **Field:** The field that is displayed in the charts. In our case it's the name used in the aggregation to be represented.
- **Type:** The Google chart type we want to represent.
- **Filters:** The variable that contains the filters for the query, this will be changed by the elements when you click on certain parts of the Google chart.
- **Cols:** Array with the labeling of each column.
- **Options:** An array with a list of the options of the element such as title.
- **Optionsbi**: An array with the options of Google chart, such as height, width, stacked...

2.1.2 Tweet Chart

Polymer element that receives an Elasticsearch index with the tweets stored in them and represents the tweets on a list with different characteristics depending on their analyzed sentiment and/or emotion.

It also has some inputs to know what data it would want to retrieve:

- **Data:** JSON with the analyzed data from Elasticsearch query.
- **Title:** title of the element that appears on screen.
- **Icon:** name of iron icon that appears at the left of the title.

2.2 Elasticsearch

Elasticsearch[4] is an open source, real time, RESTful, distributed search and analytics engine built on Apache Lucene [6]. It´s developed in Java and right now it's one of the most popular search engines with various uses, especially in terms on analytics [7]. In this project it's used for the persistence layer, as seen in figure 2.1.

[4]https://www.elastic.co/

It uses JSON documents without a schema to store data and it has numerous APIs to interact with all the information stored in it, such as Python API, native Java API, JavaScript API... In this project we'll be using both the Python API and the JavaScript API.

Elasticsearch is a really good option for the data storage since its reliable and has long term persistence. It's also distributed, making it easy to scale and use in both big and small projects and organizations. It has five main parts which compose the main storage aspect [6]:

- **Document:** A collection of fields defined in JSON format. Every document resides inside an index and has a specific type.
- **Index:** It's a collection of different documents. It uses shards to improve the performance
- **Node:** This is a single running instance of Elasticsearch. This can be a physical or a virtual server.
- **Shard:** Indexes are divided into different shards which work as an independent node. Each shard contains information on the property of the document, but doesn't contain all the JSON objects of the index,
- **Replicas:** A user can create a replica of an index or a shard, which will improve the performance by carrying out operations parallelly.

This software contains a number of different API to interact with it in different programming languages. For the development of this personal dashboard two of this APIs are used:

- **Python API:** This one is used to interact with Elasticsearch in the programming language of Python. It's used to store all the tweets after their analysis through Python.
- **JavaScript API:** This one is used to retrieve the data stored from the tweets from Sefarad using JavaScript. More importantly, through this we stored all the data coming from our facial analyzer and retrieved the same data to be used in the visualization system.

2.3 Clmtrackr

Clmtrackr is a JavaScript library created by Auduno (Audun Mathias Øygard) which is used for fitting facial models to faces in videos or images [8].

Originally, the software was made to track down essential parts of a face and set coordinates to each section, as seen in figure 2.2. With help from people and some developing from Auduno the software has developed and now has some features such as: Face substitution, Face masking, face deformation, caricature and the one that we'll be using in this project: emotion detection.

How that emotion detection feature works is by getting the coordinates mentioned before and depending on the difference in distance between the points and their position it calculates which emotion you have at a certain moment. Clmtrackr will distinguish 6 different feeling: angry, sad, surprised, happy, disgusted and fear, but the two latter are subjective and don't always work, so the emotions that are detected in this project are only the first four mentioned.

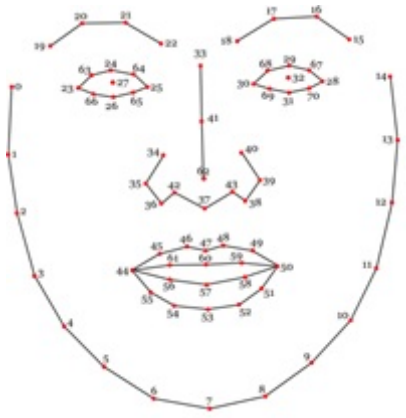

Figure 2.2: Clmtrackr face model

One of the great advantages of this library is it only needs JavaScript to work, with no external plugins or execution of other programs. The main Git Hub page for this software comes with examples of how to use all the features available as well as the code. To use a certain part, you need to copy the necessary JavaScript files and paste them in your project. Using these libraries is very intuitive and since you have examples it makes it even easier.

Probably the main disadvantage of Clmtrackr is how much computer capacity it takes. If you want your face outline appear on screen with the emotion detection this will put a heavy load on your computer system and lower end computers struggle to keep the pace. Also, the emotion detection tends to not find the angry emotion unless you exaggerate it, so

some changes have been done to the code to solve this. Another small thing (although not really important) is that this software only worked when the input video had a width/height proportion of 1.5/1.

2.4 Senpy

Senpy[5] is a framework developed by the GSI at ETSIT-UPM which analyzes sentiment and emotion in texts. One of the main advantages of Senpy is it turn the sentiment and emotion analysis into semantic web services so data can be automatically stored with the analysis and it allow developers to focus on the use of this analysis independently.

All the different analysis and emotion detection that Senpy provides share the same API, meaning it can be implemented easily and in an interchangeable way. In this project, the used Senpy services are sentiment-vader, which analyzes the sentiment of a text(the outputs are positive, negative and neutral) with different options, such as language (in this case the selected language is Spanish since it's the language in which our tweets are focused); and emotion-anew, which works like the previous but in this case It gives us the emotion of the text (the possible outputs of this analysis are emotions such as happy, sad, angry, negative-fear...).

The different widgets that Senpy provides come with some options such as language or if you want to know any extra parameters. There're also widgets that calculate the sentiment and emotion using different algorithms in case you find you are having to many of one, you can change to another method.

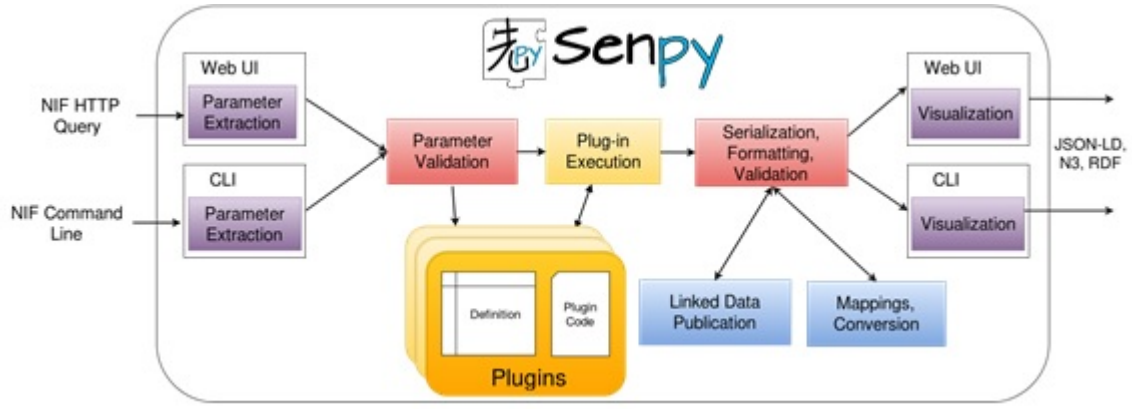

Figure 2.3: Senpy's architecture

[5]https://senpy.readthedocs.io/

2.5 Luigi

Luigi[6] is a Python module that helps you build complex pipelines of batch jobs. It handles dependency resolution, workflow management, visualization... It was initially developed by Spotify for their use and is now widely used to run different tasks in Python in a specific order.

It checks if there have been any errors in the execution of one of the jobs and of so, it won't execute the posterior ones until there is a solution to the failed on. On short, if two tasks are dependent, one won't run until the other one has finished, therefore you make sure nothing will be executed out of order in case of a problem. This can be very useful for pipelines with a lot of tasks that would be tedious to run individually and cause trouble.

In this project we used Luigi to retrieve the tweets and analyze their sentiment. It first executed the tasks that get the tweets using Tweepy and store them in a file. Then it runs functions that analyze the sentiments and emotions using Senpy and put them in JSON format with the tweets; finally, it stores all this information in Elasticsearch using the Elasticsearch python API in an index called Twitter.

Luigi is optimal for this work because it works like a pipeline, meaning the output of the first task will be the input of the second... Which in this case comes in handy since each task we are performing utilizes the output from the previous task and adds something to it. It also has a feature which allows you to visualize the status of the workflow using its visualizer. This can be very useful to keep track of the processes in real time and know what has executed correctly and has yet to be executed.

2.6 Twitter API

To collect all the tweets from a particular user we have used the API[7] that Twitter provides for this kind of tasks.

The API allows us to read and write data related to Twitter. Its main use is to read and write tweets, as well as read the profile of a particular user and their followers and other relevant information. To use this API, you need to sign up to the Twitter developer application and register an application to the service. Once you've done this you are provided a consumer key and an access token which can be used to utilize the Twitter different APIs.

[6] https://github.com/spotify/luigi
[7] https://developer.twitter.com/en/docs

There's two main Twitter APIs for retrieving tweets:

- **REST API:** This one is mainly used to collect and store all the tweets from particular users. It gives you an X number of tweets from the selected user/users a store then in JSON format.

- **Streaming API:** This is uses if you want to get tweets with a particular characteristic in it, such as tweets containing a certain word since a certain date, or tweets that contain a specific location...

In this project, the REST API was used to collect the tweets from the user using the application, although we have the code available if we wanted to find out the emotion and sentiments of tweets with a certain word in it, or even tweets that mention our account. This could be useful as a possible addition to our dashboard.

2.7 Tweepy

Since Python is our programming language used to collect and analyze the tweets, we need a Python library which can interact with the Twitter API in an easy and intuitive way; that's why this tasks are performed using the Tweepy[8] open source Python library.

Tweepy [9] takes care of all the authentication and connection with the Twitter API. It also usually responds to any error given by Twitter and tries to solve it. All Around it a really intuitive library that facilitates enormously out interaction with Twitter to collect all the tweets, and all it needs is for you to provide the consumer key and access tokens as well as their respective secrets. It also gives you options of what kind of tweets you want to retrieve, how many, if you include retweets...

It's considered the best Python library to communicate with the Twitter API and overall you can see how easy it is for the user to program a petition to the API and work with the data received.

[8] http://www.tweepy.org/

2.8 Python

As mentioned, Python[9] is the programming language chosen to be used to collect and analyze all the tweets, as well as storing the information. To do this, various libraries that perform specific tasks are used, the main ones been the Elasticsearch on python, Tweepy and Luigi, already explained before.

Some other libraries that had a use in the project where:

- **re** to compile text for it to be in Unicode to it can be analyzed in Senpy.
- **json** to create JSON documents from the data collected.
- **Pandas**[10], which is an open source library used mostly for grouping, merging and querying data structures. It's use here was to group the tweets that where collected to then make the JSON format out of it.

[9] https://www.python.org/
[10] https://pandas.pydata.org/

CHAPTER 3

Architecture

3.1 Introduction

This section details the architecture of the project. Since the personal dashboard contains completely different and independent elements, the first review will be the general overview of the architecture to show how it stuck all the elements together; then, a bigger insight is given to see a more specific overview of how each different component works.

The global architecture of the system divided into different subsystems, each of them having a specific characteristic to complete:

- **Visualization system:** As it was mentioned in the last section, our visualization system is done using Sefarad, which allows to see the data from Elasticsearch with various different widgets and is based in Polymer Web Components.

- **Persistence system:** For our data storage and persistence layer the chosen component was Elasticsearch, mainly because it's what works better with Sefarad and has all the utility we need in our project. It's responsible for storing both the data obtained from Clmtrackr and to store the tweets. The search engine it has permits us retrieve this data to be used in the visualization system.

- **Tweet-capturing and analyzing system:** This is the subsystem responsible of capturing and storing personal tweets. This is be done in Python using some external libraries mentioned further down. It also assigns an emotion and a sentiment to every received tweet.

Overall the architecture functions together using Luigi as an orchestrator on the first part of the project and Elasticsearch is who gives all the data to Sefarad, while Clmtrackr works both with Sefarad for having the video appear on screen and Elasticsearch to save the data in it.

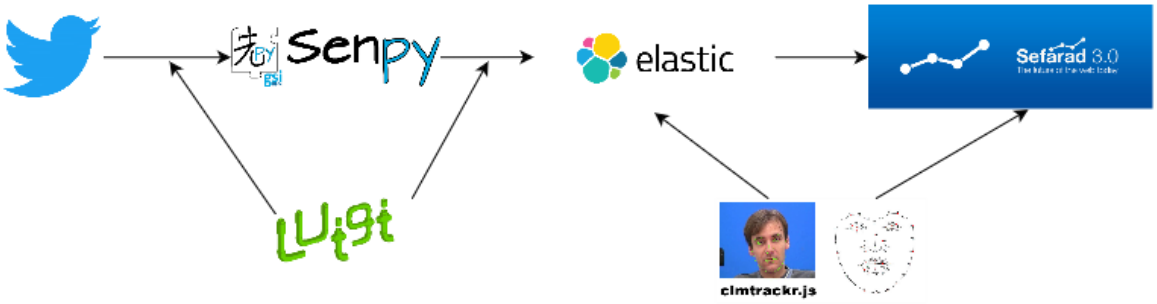

Figure 3.1: Complete architecture of the system

The following sections describe deeply the three main systems involved:

3.2 Visualization System

The main objective of this system is to visualize the data coming from Elasticsearch and to show the video coming from the camera with the emotions detected live. It will also provide some functionality so the user is able to interact and control the data that is shown on screen.

Our dashboard made in Sefarad uses several widgets that are already made as well as some made by ourselves or slightly modified by us. These widgets are mentioned in the following sections.

The goal of our visualization system is to have a visually pleasant, organized and intuitive dashboard which the user can interact with. The main aspect of the dashboard is the Clmtrackr video tracking emotion detection, it has to be placed where it gets the attention of the user. On the side of the camera recording are the graphs that correspond to the facial emotion detection. Bellow everything related to Clmtrackr is the analysis from the tweets and it's graphs.

We created some mockups before starting the project to have a main idea of how the end product should look like. Of course, in the end it would look somewhat differently as things were taken away or added, but this was the main idea (the main difference was the addition of 3 more charts and a change in the position of the tweet chart):

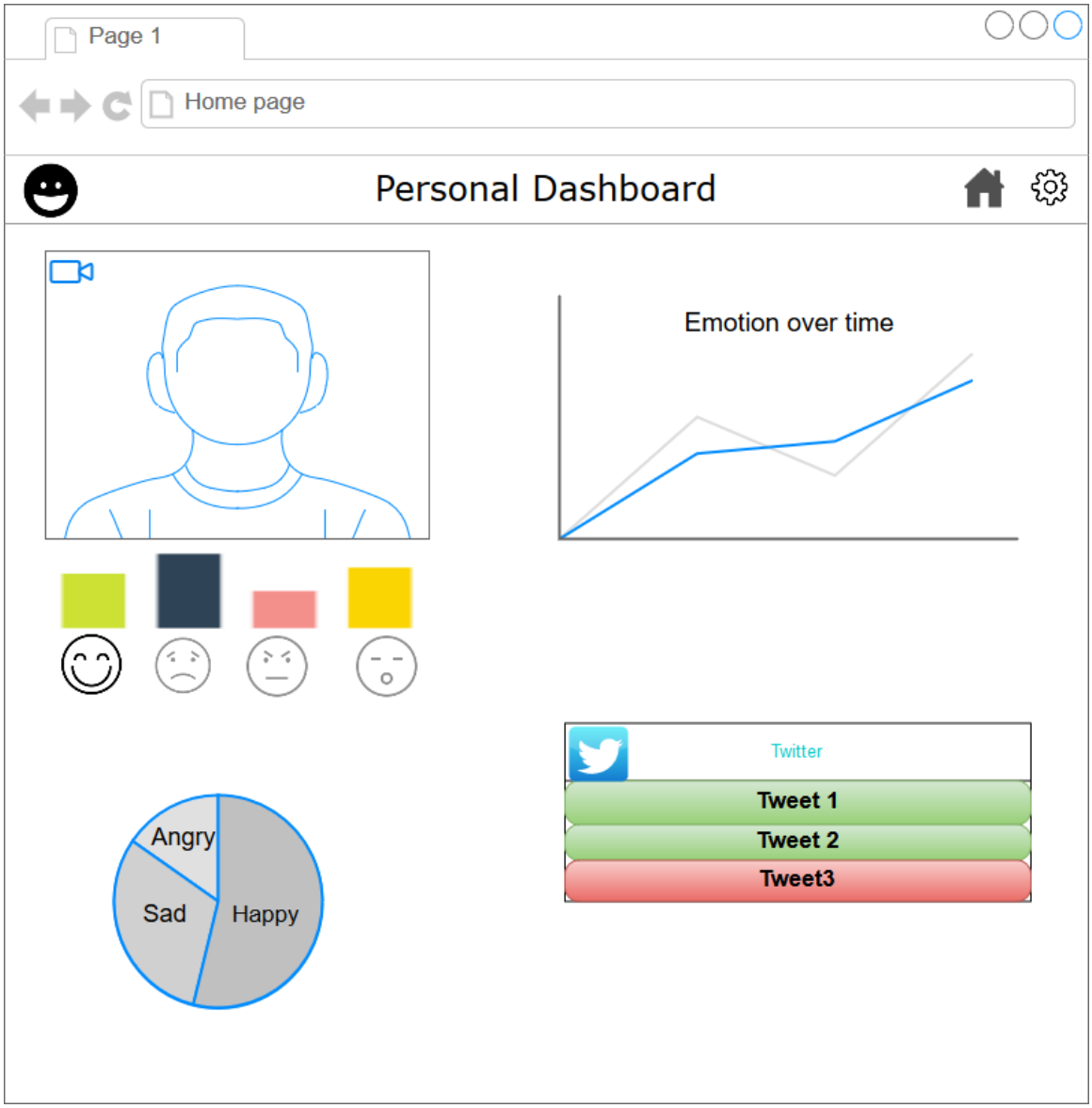

Figure 3.2: First mockup of dashboard

As previously mentioned, Sefarad bases it´s visualization server in Polymer Web Components, which, in summary, works by joining different complex elements into a single page. In this case, a new element was created called "Clmtrackr" which has the video that makes a petition to access your camera and all the code to make the emotion tracking work as well as storing all this information into an Elasticsearch index. Polymer elements has made it really simple to differentiate each section of the dashboard into a different element, keeping all the code much cleaner.

The other great factor into using Sefarad is how well it worked with Elasticsearch, since each widget could filter the data to be used separately from different queries. This becomes very useful especially in this project where there are two forms of data coming from the facial emotion tracker and the Twitter.

Figure 3.3: Final aspect of complete dashboard

These are the widgets that were used in our project:

3.2.1 Clmtrackr

This is the main widget of out project and it has been done by us using the Clmtrackr JavaScript library with some modifications to the code to adapt it to Polymer. The main objectives of this widget are:

- To get the computer's camera input on screen.
- To start the face tracker on that video.
- To get the emotion analyzer working from the face tracking system.
- To store data gathered from the emotion tracking every 10 seconds in an Elasticsearch index.

So, our main idea was to have a separate Polymer element that contains the video screen with the emotions and their values bellow, with an option to start and stop the tracking with buttons. Also, a face with the most predominant emotion should appear on top of the bar of said emotion for you to recognize which emotion will be the one stored in Elasticsearch as main one in that exact instance.

First of all, to have the library working the element had to import the necessary JavaScript files that are provided by Clmtrackr and adapt it from raw JavaScript to Polymer. To do this we had to change the way the code mentioned the different objects in the element. There're some basic JavaScript functions that don't work inside Polymer elements due to the complexity of the latter. For example, this,getDocumentById() changes in Polymer to this.id and other small details.

Also, the values for the emotions that appear on screen are stored separately and where inaccessible from our main Polymer code, so we had to be creative and store the values through some hidden inputs in HTML that then could be accessed by the Polymer element by reading these values.

Since the idea was to store the data every 10 seconds, a way to trigger a function every said time had to be found and implemented. For this the code used the setInterval function from JavaScript [10] to trigger a function which did the following:

- Capture the 4 values for the given emotion at that time using the hidden inputs mentioned above.
- Find out which one it the main emotion at that moment and store it in a variable.

- Find out today's date using JavaScript's date functionalities.
- Store these 6 fields in the Elasticsearch index.

After this was all completed, I realized that if your face went of screen, the last data sets would stay in the hidden input and they would be stored constantly even when there was no real data, just the last one registered repeated. So, avoid this I simply added a new functionality in this function which checked if the new input is exactly the same as the last one registered, in which case it wouldn't register.

Other aspects of how Clmtrackr worked were changed to make it work better in this environment:

1. Make the "angry" emotion easier to get, since before you would really have to exaggerate it for it to appear as the main emotion.
2. Add a Stop button that would replace the start button once the tracker had started. This was in case the used wanted to have a look at the data without the need of recording the emotions in that moment.

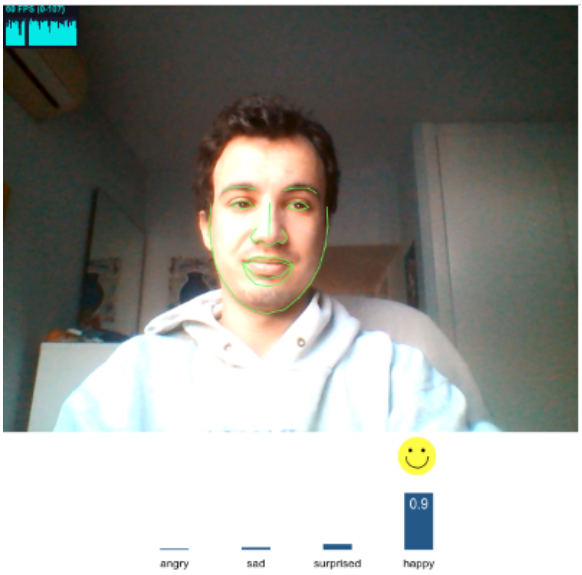

Figure 3.4: Final look of clmtrackr element

3.2.2 Google charts

For the representation of the data in the visualization system the project used various types of Google charts with some modification to get the data from an Elasticsearch aggregation.

To be more specific, the Polymer element Google-chart-Elasticsearch was used to represent the data. Lots of modifications had to be done to this element in order to represent the data in a multiple-line line graph and in an accumulated bar graph.

A great feature of this elements is it can know what section you are pressing in the charts and transfer that into filter. For example, if you want to only get the emotions from Mondays, you can click on the Monday bar chart and get on the other charts only the information coming from that day in particular. This also works with particular emotions.

There is a total of 5 Google charts that are used in this project, three related to Clmtrackr and 2 related to Twitter:

3.2.2.1 Clmtrackr pie chart

This chart represents the emotions tracked from the camera in a pie chart where you can see the percentage of each emotion you've had. This chart will help you see what your main emotion has been over a long period of time, but its main use comes when used with the filters or the slider.

For example, of you have watched a movie for 2 hours and you want to know what emotions were captured during that 2 hours you can set the slider to only show the last 720 values (which is 7200 seconds that represents two hours) and watch what percentage of each emotion the analyzer has captured.

It also has the ability to filter all the data depending on what emotion you click in. For example, if you click on sad, only the sad emotion appears on the other graphs.

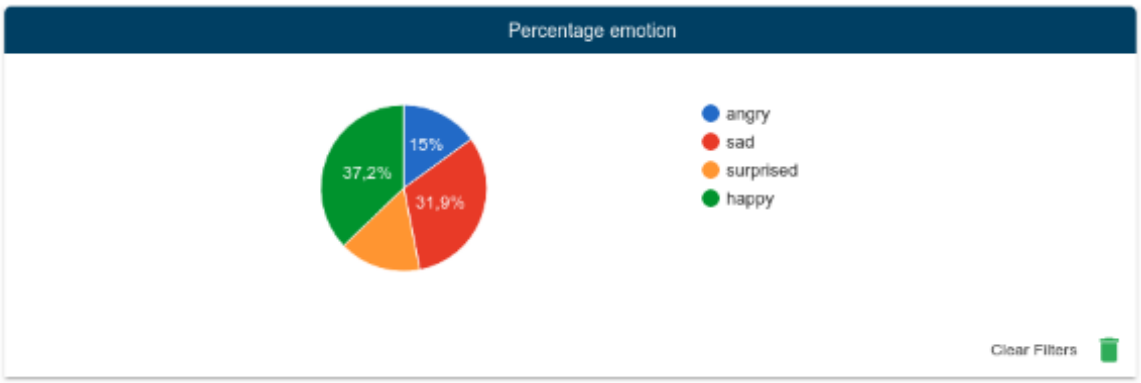

Figure 3.5: Final look of Clmtrackr pie chart

3.2.2.2 Clmtrackr line graph

In this graph shows the exact emotion the Clmtrackr element saved (as mentioned before, it's a double from 0 to 1) for each emotion. This way you can see a more precise representation of your emotions. For example, if your happy value was 0.9 but your surprise was something like 0.8, only the "happy" will appear as the main emotion, but it could also be interesting to note surprise was also a strong emotion as that point.

A big downside of this graph is that it's hard to see when there's lots of data results. For instance, if you have something like 1000 results, the graph is too small to be able to analyze it correctly; that's why is works really well using the slider or the date filters to watch a more concrete space of time, where you can appreciate the emotion values more specifically.

This graph also contains two extra options to make it easier to observe. The lines are curved by using *curvetype: function* and you can zoom in and out and move around using when giving it the option *explorer:* .

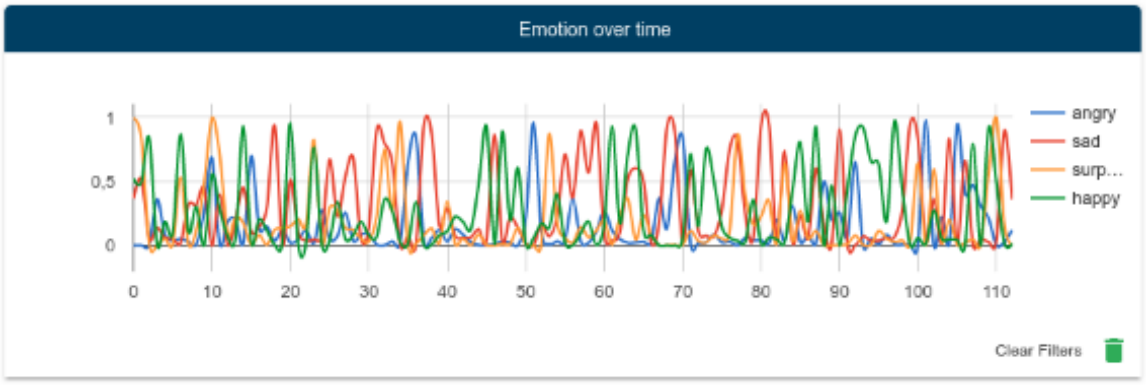

Figure 3.6: Final look of Clmtrackr line graph

3.2.2.3 Clmtrackr bar chart

This graph is designed to observe your difference in emotion for every day of the week. The idea is that after a while you can see how you have trends in how you feel different days of the week. For example, you might feel happier on Sundays than you might be on Mondays. This could work especially well if used in a normal work weekday.

The bar graph is a stacked bar graph that stacks the different emotions for every different day of the week. Also, it works by percentages of emotion, so the bars will complete to the end regardless on the amount of data they have every day. This is a way to one bar been

very big and the others small if you have a lot more values for one particular day.

Just like the pie chart, it creates different filters depending on where you click on the graph. The most relevant filter you can create is for the different days of the week. This one could be very useful when viewed from the other graphs.

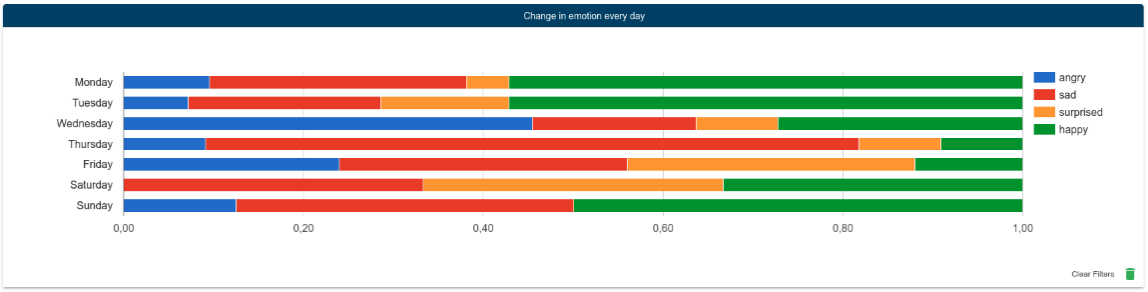

Figure 3.7: Final look of Clmtrackr bar chart

3.2.2.4 Twitter pie chart

Now the explained graphs are the ones representing the Twitter data. This first one is a bar graph containing the emotions of the tweets. These emotions come from the analysis done using Senpy and have a large variety of possibilities, such as "neutral", "negative-fear", "joy", "disgust", "sadness", "fear" or "angry", there's also other values but they are not used in our example.

The pie chart shows the percentage of each emotion in all the recorded tweets. If you click on one of the emotions, you will filter the tweets so that only the tweets that have that emotion appear on the tweet-chart element. Also, the other tweet graph that we'll mention now will filter its sentiment to the tweets with the selected emotion.

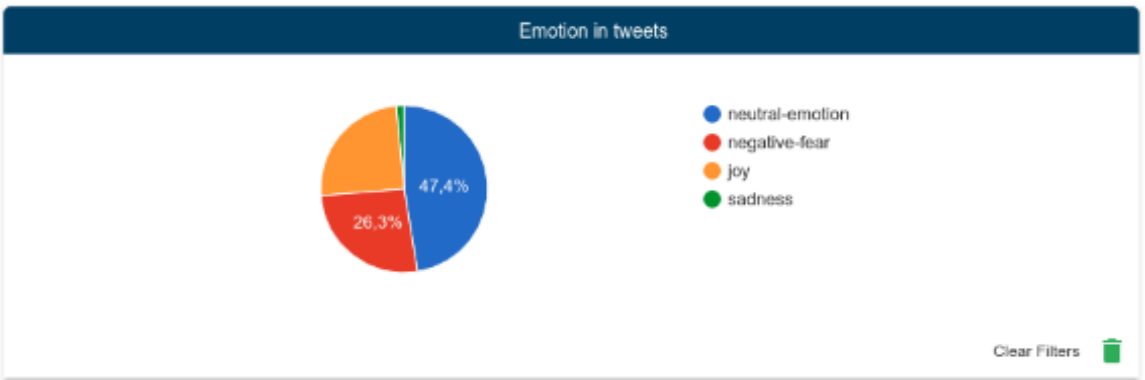

Figure 3.8: Final look of Twitter pie chart

3.2.2.5 Twitter bar chart

This bar chart does something similar to the pie chart above, but it shows the sentiment rather than the emotion. The sentiment can be either positive, negative or neutral. Each bar represents the number of tweets with that sentiment.

The filtering works exactly the same as the pie chart but with sentiments. Note that, since the sentiment and emotion are evaluated using different methods, it is possible to get something like a joy emotion and a negative sentiment, but that is rarely the case when observed filtering tweets.

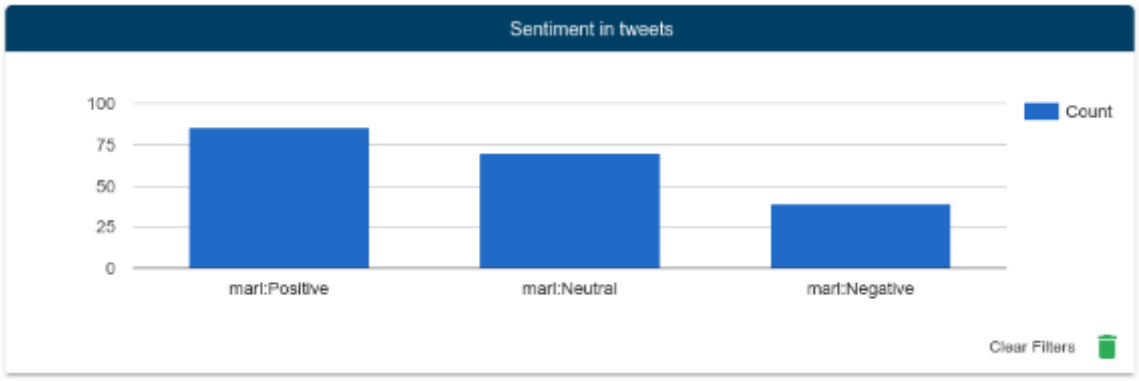

Figure 3.9: Final look of Twitter bar chart

3.2.3 Tweet chart

This is an element created to represent in a polished list all the tweets that you pass as a parameter. As it's mentioned in previous segments, these tweets are analyzed to divide them into emotions and sentiments, so out intent was to represent these things in the tweet chart in a way that you could see which one of these two divisions were assigned to each tweet.

First, a variable is sent to the element called data that is the result of the select query from Elasticsearch, that data is then treated inside the tweet-chart element to show the texts on screen as well as different CSS styles for the different emotions and sentiments.

In resume, the element had three main purposes:

1. Show the texts on a scrollable list that looks good.
2. Change the background of the text depending on the sentiment (grey for neutral, green for positive, red for negative).

3. Show a face next to the text showing what emotion the tweet has (for example an emotion of joy would have a happy face).

The first two where already done and only some slight changes had to be done in order for it to look more visually pleasing (for example changing the height of the cells containing the text from the tweets or changing the name of the sentiments for the program to get them correctly), but the last one was done completely by us. Basically, the element is calling a function from the HTML part of the element to check the emotion of the tweet that it's treating, then, depending on the emotion we send a different image link of a face showing emotion which is then represented on the left of the text. There are considered 6 different emotions outputs ("neutral", "negative-fear", "joy", "disgust", "sadness", "fear" or "angry") since they are the ones gotten after trying over 100 different tweets (there are more, but they are so rare, in case they appeared they would simply have a neutral face by them).

Overall the element ended up working pretty well and it gave a nice look to the whole dashboard. The colors blend in well with the rest of the charts and have a similar style to them.

Figure 3.10: Final look of Tweet chart

3.3 Persistence system

This is the system responsible for the storage and managing of the data used in our project. As explained before, it has been done using Elasticsearch to index and consult out documents.

To make this work, there has to be two indexes to store the two different sources of information needed: the dataset from Clmtrackr and the analyzed tweets, for which two different index were used: indexes named "trackrdata" and "Twitter" respectively. Both are stored separately and have nothing to do with each other.

3.3.1 Trackrdata Index

This first index contains the data form the camera emotion tracking device. The way the data is stored is, every 10 seconds it stores the emotion value for the four registered feelings (it's a value from 0 to 1 with 5 decimal points) as well as the day of the week it is and the most prevalent emotion in that 10 second frame.

This time between data sets can be easily changed, but after some trial and error I felt like it was the right amount that didn't cluster the graphs too much but on the same time had enough values to have a nice representation over a short period of time.

The data is inserted into the index using the Elasticsearch JavaScript API, but the index itself had to be created outside of JavaScript because of some slight problems: In you index some data into an index that doesn't exist, the index will be created automatically, but when the dashboard created the index this way through the HTML site, the mainemot and the day fields were created all text instead of keyword, therefore the query couldn't make aggregations of these keys (it's a characteristic of Elasticsearch to make queries faster). Therefore, the index was created making a direct request to Elasticsearch using Curl.

To access the data from this index the main element of the dashboard contains a search query directly in the Polymer JavaScript. It contains various aggregations to be used in the graphs. For example, and agg that counts the number of main emotions for each of the four emotions to plot them in a pie chart showing which percentage each emotion has.

Table with *trackrdata* fields:

Field	Use
Date	This field contains the date and hour at which the emotion was captured. This is used mainly in sorting the results in time, but it's also here because it could provide some really useful information for a future improvement.
Mainemot	This has stored the emotion with a higher coefficient at the moment of capturing.
Day	Contains the day of the week in which the emotion was captured. Used to create the bar chart that divides by day.
Angry	Contains the coefficient, from 0 to 1 in 5 decimal places of the Angry emotion.
Sad	Contains the coefficient, from 0 to 1 in 5 decimal places of the sad emotion.
Surprised	Contains the coefficient, from 0 to 1 in 5 decimal places of the surprised emotion.
Happy	Contains the coefficient, from 0 to 1 in 5 decimal places of the happy emotion.

Table 3.1: Fields of the trackrdata index

Since it's known the time interval between every data record is 10 seconds, the end user might want to only watch on screen the dataset from the last minute/minutes or so, that's why we added the option, using a slider, to only show on screen the desired results. We'll give an insight on how the slider and visualization works further down, but for returning only an X amount of data values, the query uses an option given by top aggregations on Elasticsearch: from and size. Using these two we got only the values we needed doing some simple calculations.

Another ability we wanted to give the user was to delete some specific data in case it was not desired or for some reason was misinterpretable data. To do this a button calls the delete function from the JavaScript Elasticsearch API, which works by deleting the introduced IDs from a specific index. The IDs used to then be deleted come from the results from the last query executed and store the desired IDs in a Polymer variable called

MyIds which contains an array with all the IDs.

The user also has the option to delete all the values from a specific day or emotion using the filters proportioned by the graphs.

3.3.2 Twitter Index

This index contains the information from all the tweets and the emotion and sentiment analysis of them. The whole process on the analysis and recovering of the tweets is explained in the last system mentioned, here we'll focus on the index itself and how it was created; and all the information stored.

The index contains a loss of information, but the main ones that the project utilizes are the tweets text, it's emotion and its sentiment. As mentioned in previous sections, this index is created and stored using the Python API.

Table with *Twitter* fields:

Field	Use
created at	This field indicates the date and time the tweet was published.
id	Unique identifier of the tweet.
user.id	Unique identifier of the user who wrote the tweet.
text	This field is the tweet text in UTF-8 format.
sentiment	This field contains the result of sentiment analysis plug-in
emotion	This field contains the result of emotion analysis plug-in

Table 3.2: Fields of the Twitter index

To access this data, the same procedure as the data from the other index was used: using the JavaScript API and including some aggregations to be used by the graphs. Basically, there are two aggregations, one for the emotion and one for the sentiment and they are used in separate graphs. There's also a filter option that filters based on what you select from the graphs.

3.4 Tweet Capturing and Analyzing System

This system is responsible for the capturing of an individual's personal Twitter and the analysis of emotion and sentiment of those tweets. This is all done in Python using multiple different libraries. This whole system is made in 4 steps:

1. Capture the tweets of a tweet account using Tweepy.
2. Analyze the emotion of the tweet using Senpy emotion-anew.
3. Analyze the sentiment of the tweet using Senpy sentiment-meaningcloud.
4. Store all this information into an Elasticsearch index.

To facilitate the whole process, the use of an external library is need, so it uses Luigi to execute the functions for every tweet. Luigi works great in this system because every individual step uses the output from the previous step like a pipe.

The following subsections explain how every individual task was done in Python, joining the second and third step together since it has similar procedures:

3.4.1 Capturing tweets

For this task we used the Python library Tweepy, that, as explained in the section 3, works with the Twitter API and tries to facilitate the tasks of working with tweets.

For Tweepy to communicate with the API you need to provide the consumer key and access tokens as well as their secrets. This is provided by Twitter if you register for a program.

After this, we set the code to retrieve the tweets from a particular Twitter name using their user_timeline function. The variable containing the Twitter user is provided by Luigi from the command that calls it.

Finally, all the gathered results are stored in a file in JSON format. This JSON contains all characteristics of the tweet such a who wrote it, when, where, and other information provided by Twitter. This file is used in the next steps, so it's saved in the same directory as the code is in but in a directory called timeline.

3.4.2 Analyzing sentiment and emotion

This step was done majorly using Senpy as our analyzer. From the JSON file mentioned before we got the text from the tweets and send them as a parameter to a function that receives a text as parameter and returns an emotion/sentiment as an output. The procedures of both where done the same way, obviously both with their own different petition.

The analysis works by doing a request to a URL that accesses a software to analyze the text. A problem of this request is that the text you send to analyze couldn't contain some of the letters used in the Spanish language and also the emojis which are widely used in tweets. To solve this we had to substitute these with other Unicode characters that are accepted in the petition (for example changing á for a or a happy face emoji for a :)). Also, the returned value wasn't always the intended value, so they were split by # to get the results that were returned after the #character, which was the emotion.

After this, a new JSON file was created which, for every tweet, containing its id, user, text, date, emotion and sentiment (this way we stop it from being all clustered with extra unneeded information). This file is created using the json.dumps function that dumps all this information from all the recorded tweets in a file.

3.4.3 Inserting the data into Elasticsearch

Elasticsearch has a Python API to save data into an index through this programming language. It's used in this project to create and save all the data into an index called Twitter.

The function that creates this index is called Elasticsearch and it basically gets the JSON data stored from the first steps and stores it in the Twitter index. Since this storing method uses JSON to store the data the procedure is smooth and doesn't need much programming to be set (you could say the hardest part is storing all the data in a JSON rather than transferring it to Elasticsearch).

Also note that, since the Index won't have a massive amount of data and it will be used only in out dashboard it only has to have one shard and one replica.

3.4.4 Using Luigi

As mentioned previously, Luigi is how functions pass on parameters and create a pipeline to reach the end result.

First of all, our whole Twitter code is divided into two Python files, one called Twitter that is in charge of the first step of getting the tweets. The other file is called Twittertask as it's our main code where we are running everything together. This works using Luigi to set a pipe to join all the tasks and run it once for each tweet.

Also, the script passes parameters to the document by using Luigi parameters from the command, in this case such as Twitter name, number of tweets to retrieve, name of Elasticsearch index... There really isn't much complex code in this other than the understanding of how Luigi works. Our final command to run the code was the following, showing the set parameters for our project:

```
python -m luigi --module Twittertask Elasticsearch --index Twitter
--doc-type tweets --num {number of tweets to retrieve} --user-Twitter
{Twitter username} --local-scheduler
```

Once called it will do every part individually and if something went wrong it says what failed. After some trial and error and some modifications of thing that went wrong, we finally managed to get everything working.

CHAPTER 4

Case study

4.1 Introduction

To try our product and prove the results that are presented aren't random and follow a somewhat logical pattern, this section explains experiments followed and analyzes the obtained results. Since the main project is divided in two sections, we followed two unrelated experiments on both Clmtrackr and the Twitter results. For the Clmtrackr there is also and explanation of how it works and a small demonstration of how to use it appropriately.

4.2 Clmtrackr

To try this element of the dashboard we recorded our data while doing/watching things that might state a certain emotion. This could be things like listening to different types of music, reading something, watching a video... Now, an explanation is given on how the whole system works and what you are able to do with the information.

Using the video emotion tracking is really simple and intuitive. Once you load the dashboard, the tracking won't be active until you hit the "start" button bellow the camera.

Once this is done it will start and a green outline appears where it considers your face is at. If it doesn't get your face correctly, move around and open and close your mouth until it does (moving the eyebrows also helps). Below the video you will see the emotion been captured in bars that go up the higher the emotion it found. Once you reach 0.6 in an emotion a face appears on top of the bar to represent it. To stop the tracking, you can either reload the page or simply hit the stop button that now replaces the start.

Once data is been collected, you can hit the "update" button at the right to update the data into all the graphs. If you hit the "delete data" button, it will delete **only the selected data** at that moment (so if you have a filter or a number of data sets it will only delete the selected ones). As mentioned in previous sections, the slider on top of the buttons is there if you want to center your data around the last results (each result corresponds to 10 seconds).

If you want to filter your data to a certain day of the week you can do this by clicking on the desired day in the bar chart. The same can be done if you click an emotion from any chart- it filters data to only show that emotion (the line graph will show the data when said emotion is the main one, this is useful if you want to know what your secondary feeling normally is when the primary one is the selected).

You'll realize each person has a different emotion when having a "normal" face, for example, for me it would default to sad, while for my brother it would appear happy (eyes take a big part in this), so you must adapt how you interpret your data to what you normally get.

After trying the software for days, I say a clear trend in my change in emotion while I was watching something that made me feel a certain way. For example, when watching a comedy show with my family, I would leave the computer in front of me recording and I would have a much higher percentage of happy and surprised than I did while doing other things such as writing this project.

Overall, I'm really happy with the final result, and the slight changes in the code I made to make sad not be as normal and angry to be a bit more normal where definitely appropriate and made emotions a bit more scattered.

4.3 Twitter

The best way to demonstrate Twitter results correlate to what they should is to try it on different users and correlate the results to their type of Twitter account. This section

also mentions how to analyze the obtained data from the graphs to use them on your own Twitter account.

This experiment was composed of different Twitter accounts that have a really defined theme to it, such as a political, sport, humor or motivational account and compare said data to what was expected and determine if the emotion analysis is somewhat precise. For example, a political Twitter should have a higher percentage of negative sentiments than those from a motivational or humor account (although the latter might be tricky, since wordplay takes a big part in misleading analysis).

Through this thought process, there can be an observation of the different results we obtain and determine whether they where as expected, or our classifiers didn't do the expected job. To have a decent data set I'll collect 250 tweets from each account without including retweets (basically because we want to observe the emotion through the text someone writes). Also, our analysis is based in Spanish text, so the selected accounts have to be in Spanish and have a decent number of tweets.

The selected Twitter accounts from which data will be gathered are the following:

- **@MarioAlonsoPuig:** he is a doctor claiming to be an expert in motivation, creativity and communication. His tweets are mostly positive and try to have a motivational vibe to them. That's why this is our **motivational** account.
- **@2010MisterChip:** This Twitter account is run by an expert in data analytic related to sports, and tweets thing mainly in curious data that occurs every week in sports events. He tries to be as unbiased as possible, so that's why it is the chosen **sports** account.
- **@elpais:** this is the official Twitter of the biggest newspaper in Spain. They tweet both serious news that are happening worldwide and some curious news about mundane things. This will be our account related to **news**.
- **@protestona1:** this Twitter account is a **political** based account which basically likes to protest about everything and give negative emotion to its tweets. This is a great example of an account that could give relevant information.

Now that the accounts are selected, we'll try to retrieve the desired number of tweets and see if the data corresponds to what it should. A small negative thing of the Twitter API is that you can only get the last 300 or so tweets from an account, and this includes retweets, so we've tried to pick people that write themselves tweets more than retweet others to have at least 100 results from everyone.

The results are the following:

4.3.1 Motivational Twitter

This should probably be the account with more polarizing results in terms of emotion and sentiment, since the texts that come from here are mostly positive and try to have a good vibe to them. Obviously, there will be tweets talking about negative things and how to overcome them, and I expect most of the negative sentiment of fear emotion to come from them.

Results:

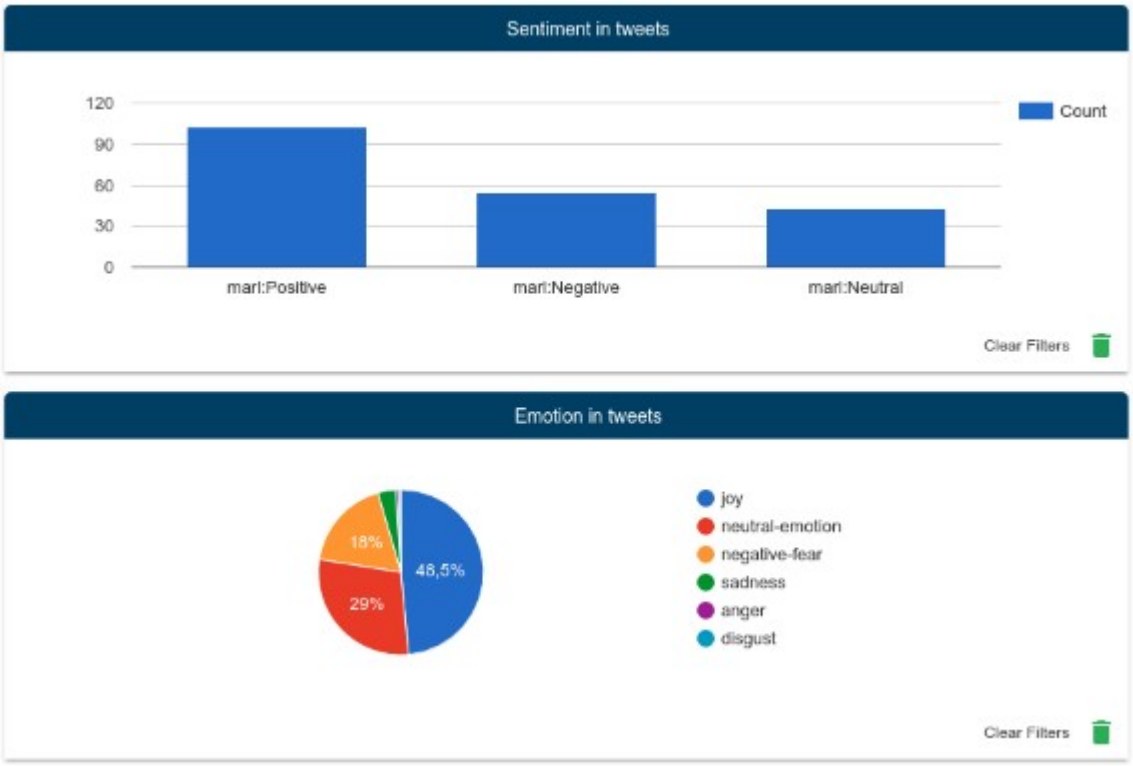

Figure 4.1: @MarioAlonsoPuig analysis results

Our total result set was composed of 200 tweets.

Our sentiment bar chart shows how a big percentage of the tweets are positive, having 102, for the 55 negative and only 43 neutral. This completely corresponds to what was expected (maybe even a bigger difference, but there's a few tweets talking about how to overcome negative things and some are marked as negative due to word-play). I think it's also noteworthy that there are fewer neutrals than negatives.

In terms of emotion the trend is similar: joy is the main one taking nearly 50% of the samples. Also, the sadness is really low at 2,6%, showing that our initial predicaments where right.

This result set is really good to start with our experimentation and shows the results follow a pattern that match the expected.

4.3.2 Sports Twitter

This sports account tries to be as unbiased as possible and is mostly about data facts, so emotion here should tend more to the neutral side. On the other hand, he also likes to praise players and encourage people to follow someone or even talk about how great a match was... Overall, I expect the results to be more on the positive side, but it wouldn't be surprising if neutral mas the main emotion.

Results:

Figure 4.2: @2010MisterChip analysis results

In total there are 193 data samples for this account.

In this case, the sentiments went a lot more to the positive side than expected, with

also a really high percentage of neutral. There here 85 positives, 70 neutrals and only 39 negatives. After reading through the last couple of tweets, you can see that he tries to give positive facts, and very rarely does he give negative records. Also, he uses a lot the words "record", "positive", "win" and exclamation marks, which seems to trigger the positive sentiment; therefore, these results seem more according than initially thought.

In terms of emotion the result was more expected, with nearly halve being neutral and joy having half of the remaining. Also, there's only one sadness result and no anger or disgust, which has complete sense in a fact and statistical account.

Overall results are close to expected. It seems like emotion and sentiment don't always "agree" with each other, but that gives a nice contrast of ideas.

4.3.3 News Twitter

News can be a little tricky when analyzing its emotion since there are positive and negative news. Overall, I'd say most of the serious news that are written nowadays are about negative things that has happened in the world, so our expectations are that the results would tend more to negative and fear, but not as much as our political example.

Results:

First of all, there are 118 results which I consider enough to be evaluated.

Of these 118 tweets, the grouping in sentiment where: 43 set as negative, 41 as neutral and 34 as positive. This corresponds to what was expected from this account. More negative than positive, but not by a huge amount. In my opinion the number of neutral tweets is surprising since news tend to show a strong emotion to them, so that's a small detail to be considered

In terms of emotion, there was a smaller number of neutral tweets than the rest of the examples, which correlates to what was expected. It also got a similar number of joy and negative-fear and finally the sadness percentage (8.6%) was bigger than in the rest of examples, which also makes sense.

As a whole the results showed to be scattered but a bit more on the negative side, which falls into our logical assumptions. This consolidates the analysis as valid as a whole, with some minor exceptions.

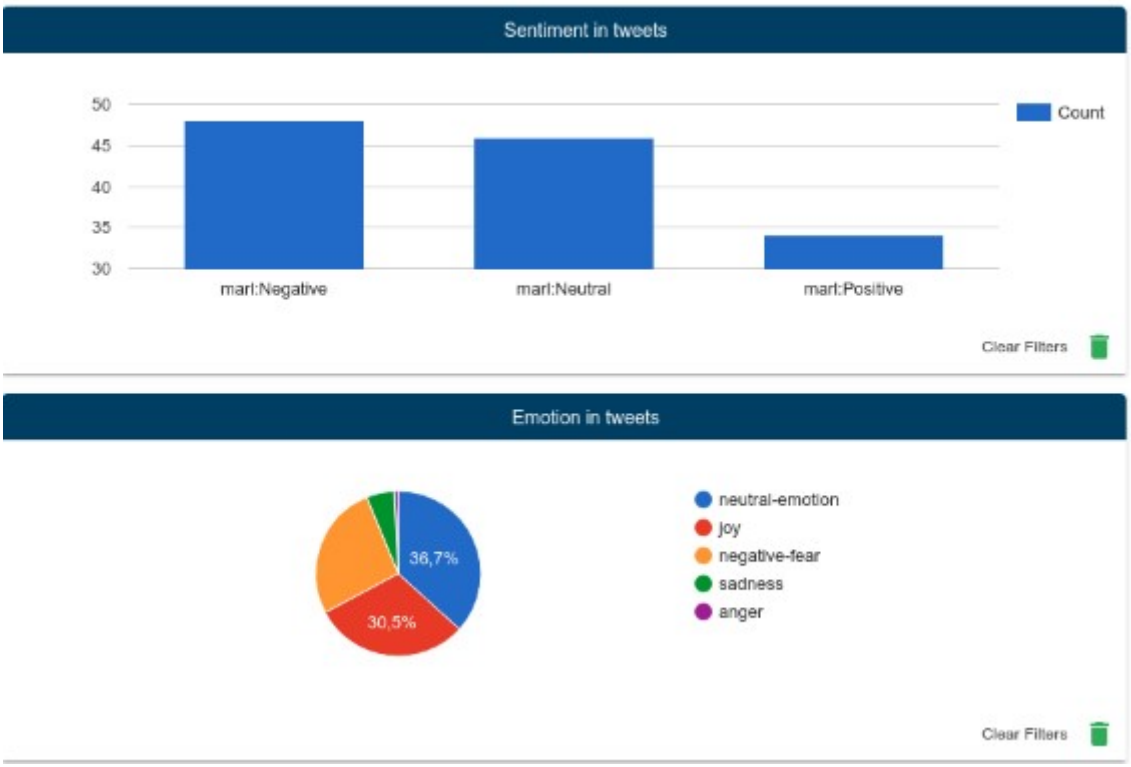

Figure 4.3: @elpais analysis results

4.3.4 Political Twitter

Political tweets tend to be more negative than the normal ones, normally complaining about something or attacking the opposing side. There should also be a few that writes something positive about their side, but our idea was to have this Twitter account to be on the negative spectrum in terms of sentiment and emotion. The selected account has mostly negative tweets and the results should show that.

Results:

In this data sample a total of 151 tweets where collected.

This one probably shows the most expected results of all the examples in terms of sentiment, having nearly a 50% (69) of negative and less than that of positives (29). This corresponds to what was expected before. Also, after watching some of the positive ones, some where due to the use of sarcasm and others where simply nice texts.

In terms of emotion there more of an unexpected result. There as a majority of neutral texts and more joy than negative-fear, which is totally unexpected. After analyzing the tweets and their corresponding emotions the reason I found for this where mainly that there

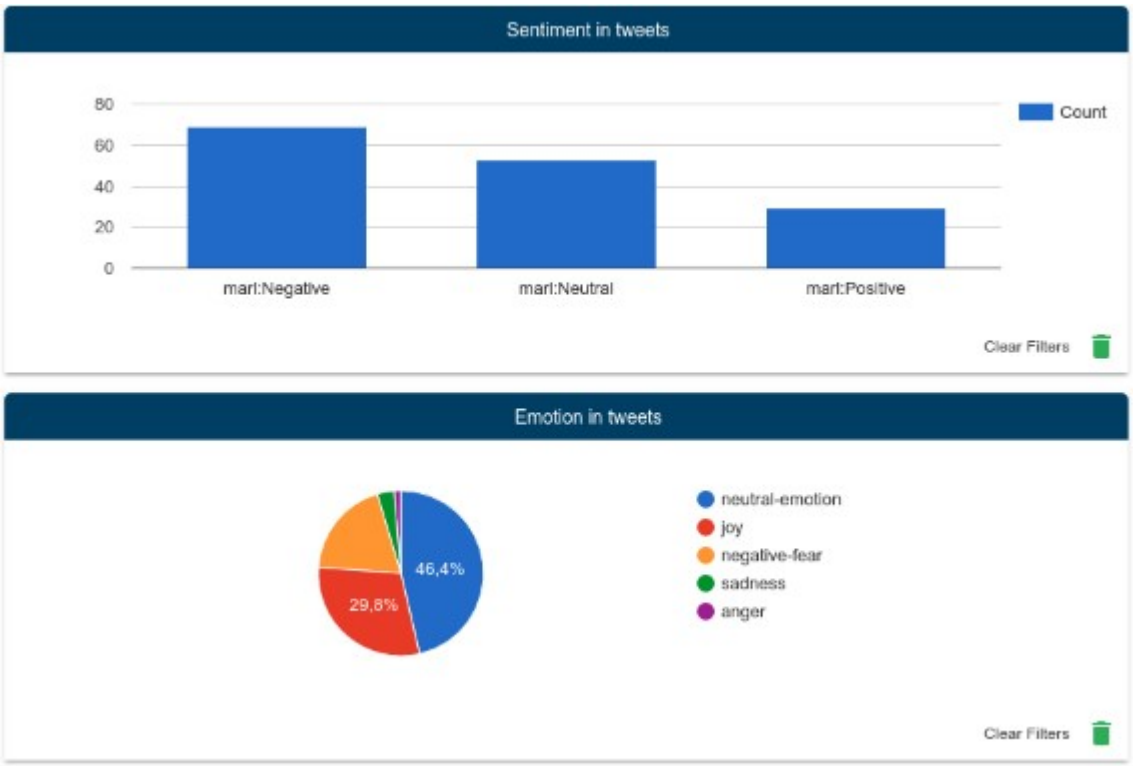

Figure 4.4: @protestona1 analysis results

are a lot of short texts in this account which are marked as neutral emotion and also, the use of sarcasm and wordplay seems to make the tweets go neutral emotion (probably because it checks for individual words and normally the beginning of a text would be "positive" due to sarcasm and the ending negative, which canceled out).

Overall results kind of match what was expected, and we realized that sentiment seems better at figuring out sarcasm than emotion.

4.3.5 Twitter analysis results

At a whole, the results obtained correspond to what was expected, and honestly gave results even better than I thought. There are some minor exceptions, but given the data sets this falls into normal behaviour. There are also some other conclusions to be obtained from the data sets:

- Emotion and sentiment don't always get the exact same results.
- Sarcasm takes a great part in misleading data, but the sentiment analyzer seems to do a better job at detecting it.

- Emotions seems to focus more on individual words and cancels out when positive and negative ones are used, while sentiment looks more at the text at a whole.

CHAPTER 5

Conclusions and future work

This chapter details the conclusions that have been reached after the creation of the dashboard and the experiments set on it. Also, it discusses some problems that were faced while doing the project at a whole and how they were solved.

Finally, it mentions some possible improvements to the personal emotion dashboard that couldn't be implemented either for lack of time or ability.

5.1 Conclusions

In this project we have created a personal dashboard for emotion tracking using video and text evidence, also, all the data was made accessible and interactive through different means. The dashboard is structured in an intuitive and colorful way, so the user knows what to do at every moment and feel like they want to use the product.

The dashboard is composed by two main systems that are independent to each other, the video tracking by Clmtrackr and the text analysis by Twitter. The both have their respecting graphs to show data in a way the user can understand it and use it for their own purpose. Overall, the graphs and the filters they provide make a great use of the

information collected, trying to get all the useful information.

After trying both components, the results are very pleasing and demonstrate that they do their job correctly most of the time: know your emotions. This was the primary subject of the project, and I think it has been achieved in both ways that were intended.

Now in the following sections I will describe in depth the achieved goals, the problems faced and some suggestions for a future work:

5.2 Achieved goals

The following sections explain the achieved goals that this project has accomplished at a whole:

- **Use the Clmtracker video tracking to get your emotions in real time:**

  This was probably the main objective of the project. We had to adapt the library to Polymer and create an element from it. Also, some changes where made to make emotions more precise.

- **Collect your personal tweets and analyze their sentiment and emotion:**

  This goal was achieved completely by Python using different libraries. Tweepy was used to collect the tweets from the entered user and Senpy to group them by said fields. Then, Luigi made the process more automatic like a pipeline and stored everything in a file.

- **Store all the data in Elasticsearch indexes:**

  For the Twitter index the project simply used the Elasticsearch Python API to store the information which was previously saved in a JSON format, which wasn't hard. The hardest part was probably to collect the data received from Clmtrackr, adapt it to Polymer and through various methods insert it in Elasticsearch, using its JavaScript API.

- **Represent all the data in a dashboard with various charts:**

  The dashboard and everything it in was created using Sefarad which is based in Polymer Web Components; and various elements to represent the data. The charts where represented using Google-charts and the tweets using tweet-chart. Both with their own modifications to adapt them to our project and show the data that was thought to be more relevant.

5.3 Problems faced

During the development of this project a few problems were faced. These problems are listed below:

- **Adapting Clmtrackr to Polymer:**

  This was probably the biggest issue faced in the project. We had to change part of the code to initialize this component and do things such a creating the video interface by JavaScript instead of plain HTML, creating hidden inputs to store data, create interval functions to update and store the data... In the end everything was solved and got working, but it did take a longer time than expected.

- **Special characters in tweets:**

  The special characters in the Spanish alphabet and the emojis that people use in tweets gave us errors when their sentiment was tried to be analyzed. The code had to include a compiler in Python that took away all these characters and replace some manually using the ".replace" function in Python. This was really frustrating because sometimes an emoji or character that the compiler didn't consider would appear and the whole tweet process had to restart from the beginning after solving that particular case.

- **Adapting Google-Charts:**

  The element Google-Chart-Elasticsearch was already created to access data from Elasticsearch aggregations, but there were two problems: since more than one field was wanted in most cases, the query for collecting data had to use top aggregation, which made accessing the data different and so code in said element had to be changed. But most importantly, the stacked bar chart and the line graph weren't really provided, so we had to code the way to collect and use the required data from zero.

5.4 Future work

This next sections explain the possible new features or improvements that could be done to the project.

- **Let the user pick their Twitter account from the dashboard:**

  Right now, you have to collect the tweets from python using the provided Luigi command, but the initial idea was to do this from HTML in a text input field. The

problem was mainly that the Senpy analysis couldn't be done through JavaScript as of right now (at least not to my understanding), so that part was scratched.

- **Relate the tweets time to the video emotions:**

  It would be nice if you could relate the time at which a tweet was written to the video emotion tracking in order to check if both analyzed emotions match. To do this we already decided to store the date with time of both things, in case in the future something like this wanted to be done.

- **Add other elements to the dashboard:**

  On the end of the day this is a personal dashboard, so it would be nice it other things were added, such as emotion of friends, email sentiments, change in emotion through months... This would give the image of a more complete personal dashboard that felt more of your own.

- **Add another source of emotion tracking:**

  The idea that was in the first draft of the project was to track and analyze music we listened to. Using Spotify it is possible to collect the music you listened to, the problem came mainly when analyzing said music, I didn't find any reliable source.

APPENDIX A

Impact of this project

A.1 Introduction

Emotion has always been a wide area of study in which lots of professionals dedicate their lives to analyze. In this project we intended to give people a small help to track their emotion through different means.

Two great ways to determine this is by the look of your face and by what you write, and that's what this dashboard uses to save your emotion.

This appendix covers the impacts that this project accomplishes from a social, economic, environmental, ethical and political point of view.

A.2 Social impact

As it has been mentioned, the idea of this personal dashboard it to provide the used a set of data in which they can rely to keep track in changes in emotion, and what affects them.

The main use of this dashboard would probably come from individual clients who simply want to track their emotions and are finding a tool to do so. There's other applications that

are widely used right now which consists in the used introducing their emotion of the day. Sometimes this can be hard to know by yourself, so this is could be a complimentary tool to get a more precise data. It also can be useful to find changes in your normal emotion, this could be used in prevention of depression and treating it quicker,

This can also be used by a psychology company to make a customer use it and know their emotions a bit more in detail.

A.3 Economic impact

In this section we'll assess the possible economic impacts this project provides to the user of company using it.

From a normal user, we could consider this as substitution to other sorts of emotion detectors, and in some cases simply as an addition to their software. Some benefits it could bring include the detection of depression and therefore drop in money for treatment in it.

From the point of view of a company it could be a tool that reduces the time needed with each patient.

A.4 Environmental impact

This section s the main environmental impact of the development this personal dashboard

Computers and other information technology infrastructures consume great amounts of electricity, adding a huge charge on our electricity networks and contributing to negative effects getting this electricity produces. This is probably the only form of environmental impact of this project, since no paper of physical materials were used.

A.5 Ethical Implications

This section evaluates the ethical implication of this project.

The main ethical problem of this project comes from the idea of collecting tweets from someone in particular. One could argue that this is unethical, and our dashboard shouldn't contain this information. But, first of all, this should be the personal twitter of the user, so I doubt the user would not want this displayed, especially since this is completely private

and no one else should see it. Also, the privacy policy used by Twitter indicates that users consent to the collection, transfer, and storage of data that is public.

APPENDIX B

Economic budget

B.1 Introduction

In this appendix we are going to make an adequate economic budget for the realization of this project. The main parts of this budget will be explained in the following sections.

B.2 Physical resources

The main physical cost of this project consist in the computer needed for it to work. We could also include a server if the application was to be uploaded in a server but it can also be installed locally, this is the chosen path.

The computer should have some minimal characteristics, specially since Clmtrackr does need some decent power to work smoothly:

- **CPU:** Intel core I7.
- **RAM:** 8 GB
- **Disk:** 500GB

- **Graphing:** Any Nvidia GT or GTX 700 or higher

A computer with these features should cost around **900€.**

B.3 Human resources

Considering a user with no previous experience in the sector, it is estimated that 10€/hour should be the salary. The number of hours dedicated to the project, including the written report, rounds to 330h, so the net salary of the worker it around **3300€.**

B.4 Taxes

Taxes should be taken into consideration if we consider this as job. In which case, given our age and state, it will be 18% in taxes, which with a salary of 3300€, we are having to pay 594€ in takes.

This leaves us a salary for the project of **2706€.**

Bibliography

[1] Paul Ekman. *Darwin and facial expression: A century of research in review.* Ishk, 2006.

[2] Charles Darwin and Phillip Prodger. *The expression of the emotions in man and animals.* Oxford University Press, USA, 1998.

[3] Byoung-Moo Kwon and Kang-Hee Lee. An introduction to face-recognition methods and its implementation in software applications. *International Journal of Information Technology and Management*, 17(1-2):33–43, 2018.

[4] Mark E Larsen, Tjeerd W Boonstra, Philip J Batterham, Bridianne O'Dea, Cecile Paris, and Helen Christensen. We feel: mapping emotion on twitter. *IEEE journal of biomedical and health informatics*, 19(4):1246–1252, 2015.

[5] Joao Aires and Daniel Gonçalves. Personal information dashboard-me, at a glance. In *PIM 2012 Workshop*, pages 1–8, 2012.

[6] Michael McCandless, Erik Hatcher, and Otis Gospodnetic. *Lucene in action: covers Apache Lucene 3.0.* Manning Publications Co., 2010.

[7] Clinton Gormley and Zachary Tong. *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine.* " O'Reilly Media, Inc.", 2015.

[8] AM ØYGARD. Emotion detection example. clmtrackr. *URL: http://auduno. github. io/clmtrackr/examples/clm_em otiondetection. html. Vigente al*, 2, 2015.

[9] Joshua Roesslein. tweepy documentation. *http://tweepy. readthedocs. io/en/v3*, 5, 2009.

[10] Michael Mikowski and Josh Powell. *Single page web applications: JavaScript end-to-end.* Manning Publications Co., 2013.

[11] Daniel Suárez. Design and development of a system for sleep disorder characterization using Social Media Mining. Tfg, ETSIT, Madrid, June 2018.