TRABAJO FIN DE GRADO

Título:	Desarrollo de una interfaz para análisis de sentimientos de fútbol en Twitter basada en Web Components y D3.js
Título (inglés):	Development of a Dashboard for Sentiment Analysis of Football in Twitter based on Web Components and D3.js
Autor:	Alberto Pascual Saavedra
Tutor:	Carlos A. Iglesias Fernández
Departamento:	Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	Mercedes Garijo Ayestarán
Vocal:	Álvaro Carrera Barroso
Secretario:	Juan Fernando Sánchez Rada
Suplente:	Tomás Robles Valladares

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

DEVELOPMENT OF A DASHBOARD FOR SENTIMENT ANALYSIS OF FOOTBALL IN TWITTER BASED ON WEB COMPONENTS AND D3.JS

Alberto Pascual Saavedra

Junio de 2016

Resumen

Esta memoria es el resultado de un proyecto cuyo objetivo ha sido desplegar y desarrollar una interfaz para análisis de sentimientos de fútbol en Twitter basada en Web Components y D3.js.

Para hacer esto se ha desarrollado un entorno de visualización para presentar los datos obtenidos de Twitter y analizados con Senpy. Este entorno de visualización se ha desarrollado con Polymer Web Components y D3.js

El minado de datos se ha realizado con un flujo entre Twitter, Senpy y ElasticSearch. Se ha usado Luigi en este proceso ya que ayuda a construir flujos complejos de tareas agrupadas, Luigi se ha encagado de realizar el análisis de los tweets y almacenarlos en ElasticSearch.

A continuación, se ha empleado D3.js para crear widgets interactivos que hacen que los datos sean fácilmente accesibles, estos widgets permitirán al usuario interactuar con ellos y filtrar los datos que le sean más interesantes. Para hacer la interfaz acorde al Material Design de Google y mostrar datos dinámicante en los widgets se ha usado la librería de Polymer Web Components.

Como resultado, este proyecto nos permitirá realizar un amplio análisis de la red social, haciendo hincapié en la influencia de los jugadores y de los equipos y de las emociones y sentimientos que generan en un momento determinado.

Palabras clave: Web Components, Fútbol, Sentimientos, Emociones, D3.js, Twitter, Análisis

Abstract

This thesis is the result of a project whose objective has been to develop and deploy a dashboard for sentiment analysis of football in Twitter based on web components and D3.js.

To do so, a visualisation server has been developed in order to present the data obtained from Twitter and analysed with Senpy. This visualisation server has been developed with Polymer web components and D3.js.

Data mining has been done with a pipeline between Twitter, Senpy and ElasticSearch. Luigi have been used in this process because helps building complex pipelines of batch jobs, so it has analysed all tweets and stored them in ElasticSearch.

To continue, D3.js has been used to create interactive widgets that make data easily accessible, this widgets will allow the user to interact with them and filter the most interesting data for him. Polymer web components have been used to make this dashboard according to Google's material design and be able to show dynamic data in widgets.

As a result, this project will allow an extensive analysis of the social network, pointing out the influence of players and teams and the emotions and sentiments that emerge in a lapse of time.

Keywords: Web Components, Football, Sentiments, Emotions, D3.js, Twitter, Analysis

Agradecimientos

Gracias a mi tutor Carlos, por darme la oportunidad de realizar este proyecto y por brindarme toda su ayuda y su paciencia.

A mis padres y mi hermana por todo el apoyo que he recibido durante esta carrera y por animarme siempre a terminarla.

A todos mis amigos de la escuela y de fuera de ella, han sido un apoyo incondicional durante estos años y me han ayudado en todo lo posible.

A Sandra por haber compartido horas de trabajo conmigo y apoyarme siempre que lo necesito.

Gracias.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

Nowadays, Twitter has become an interesting scenario for social analysis. We can find in this micro-blogging site millions of interactions between users. The main and most popular feature is its simplicity and synthesizing, that is because in most micro-blogging systems, user's messages are delimited by the number of characters. In those characters you can do many things such as talking about what you are doing, interact with other users, etc.

In this project we are focusing in one thing you can also do with these micro-blogging systems, give your own opinion. This means, Twitter is a source of many varied opinions about a topic.

The topic we are managing is football, because is known that sporting events evoke strong emotions amongst fans. The idea is to relate these sentiments generated in Twitter with these football events in order to analyse correlations between them. In particular, this final graduate work will be focused on the application of visualisation techniques for providing insight about these relationships.

To do so, we will use visualisation techniques based on interactive widgets to provide

easy access to data.

1.2 Project goals

In the long term, this project aims at showing to users sentiment analysis based on football. With this aim, the project will collect related tweets about football and create a pipeline for sentiment and emotion analysis. In addition, we will develop a dashboard with interactive widgets to understand the results of the analysis.

Among the main goals inside this project, we can find:

- Collect tweets focused in football events.
- Build a pipeline for sentiment and emotion analysis
- Design a dashboard based on widgets that allow users interact with them which will show the information collected .

1.3 Structure of this document

In this section we provide a brief overview of the chapters included in this document. The structure is the following:
Chapter 1 explains the context in which this project is developed. Moreover, it describes the main goals to achieve in this project.
Chapter 2 provides a description of the main technologies on which this project relies.
Chapter 3 describes the architecture of this project, including the design phase and implementation details.
Chapter 4 explains how the visualisation module is designed and the components it is made of.
Chapter 5 presents experimentation details.
Chapter 6 discusses the conclusions drawn from this project, problems faced and suggestions for a future work.

CHAPTER 2

Enabling Technologies

2.1 Introduction

In this chapter, we are going to give an insight into techniques used in this project. First of all, we are going to explain Polymer web components, the technology in charge of the visualisation's structure. Secondly, the technology that has made possible interactive widget making. Finally, the technology we have used to store all the data.

2.2 Polymer web components

Polymer is a software library used to define and style web components that was developed by Google [1]. Modern design principles are implemented as a separate project using Google's Material Design design principles [4].

Polymer makes easier to build your very own custom HTML elements. Creating reusable custom elements can make building complex web applications easier and more efficient. By being based on the Web Components API's built in the browser, Polymer elements

[1] https://www.polymer-project.org/1.0/

are interoperable at the browser level, and can be used with other frameworks or libraries that work with modern browsers.

These custom elements are particularly useful for building re-usable UI components. Instead of continually re-building a specific navigation bar or button in different frameworks and for different projects, you can define this element once using Polymer, and then reuse it throughout your project or in any future project.

Polymer uses a declarative syntax to make the creation of your own custom elements easier, they use all standard web technologies: HTML is used to define the structure of the element, CSS is for style personalization and you can use JavaScript to make these elements interactive.

In addition, Polymer has been designed to be flexible, fast and close. It uses the best specifications of the web platform in a direct way to simply custom elements creation.

2.2.1 Element Catalog

The Polymer project includes a collection of pre-built elements that you can drop on a page and use immediately, or use as starting points for your own custom elements. In figure (2.1) is shown the main Polymer element catalog categories. Now we are going to see what kind of elements are in each category.

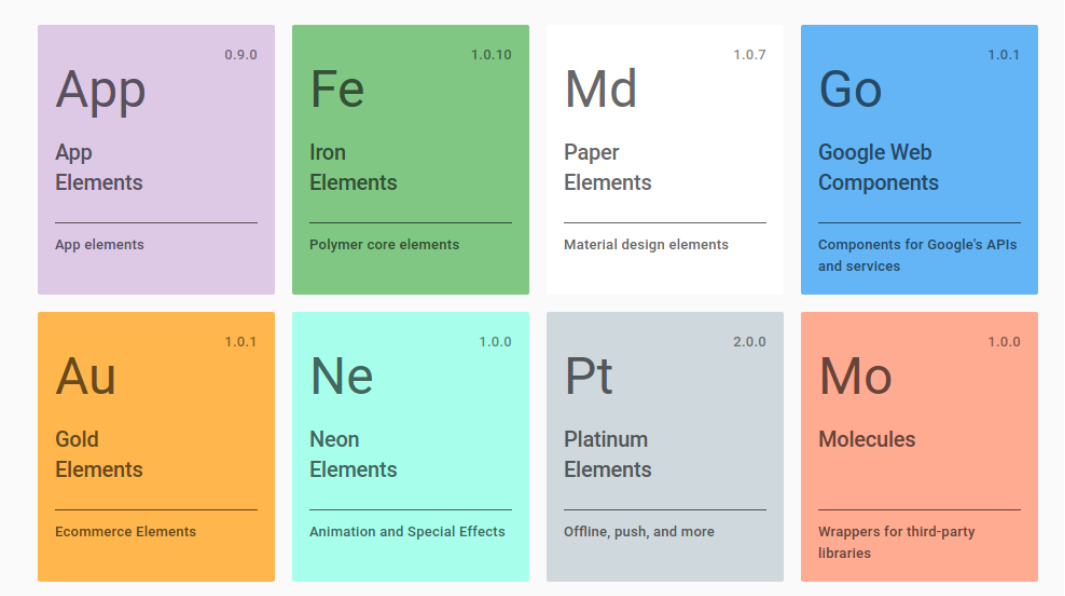

Figure 2.1: Polymer Catalog

- ***App Elements:*** These elements enable building full web applications out of modular custom elements. The main element is ***app-layout***, it gathers a set of layout elements for building responsive web applications.

- ***Iron Elements:*** is a set of visual and non-visual utility elements. Includes elements for working with layout, user input, selection and scaffolding applications.

- ***Paper Elements:*** are a set of User Interface components designed to implement Google's Material design.

- ***Google Web Components:*** In this category we can find the Google's most popular services integrated in polymer, such as YouTube, Maps, Docs ...

- ***Gold Elements:*** These elements are built for e-commerce use-cases. Here we can find form credit card input to phone input validator.

- ***Neon Elements:*** is a suite of elements and behaviors to implement pluggable animated transitions for Polymer Elements using Web Animations

- ***Platinum Elements:*** these elements are used to turn your web page into a true web application, with push notification, off-line cache and Bluetooth support.

- ***Molecules:*** are elements that wrap other JavaScript libraries. By this time, Polymer only offer support for Markdown source and Prism.js.

2.3 D3.js

D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using standard web technologies HTML, SVG, and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation [9] [1].

D3 allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document. For example, you can use D3 to generate an HTML table from an array of numbers. Or, use the same data to create an interactive SVG bar chart with smooth transitions and interaction.

D3 is not a monolithic framework that seeks to provide every conceivable feature. Instead, D3 solves the crux of the problem: efficient manipulation of documents based on

data. This avoids proprietary representation and affords extraordinary flexibility, exposing the full capabilities of web standards such as HTML, SVG, and CSS. With minimal overhead, D3 is extremely fast, supporting large datasets and dynamic behaviors for interaction and animation. D3's functional style allows code reuse through a diverse collection of components and plugins.

2.3.1 Selecting Elements

Modifying documents using the W3C DOM API is tedious: the method names are verbose, and the imperative approach requires manual iteration and bookkeeping of temporary state. D3 employs a declarative approach, operating on arbitrary sets of nodes called selections. Selectors are defined by the W3C Selectors API and supported natively by modern browsers. Elements may be selected using a variety of predicates, including containment, attribute values, class and ID.

D3 provides numerous methods for mutating nodes: setting attributes or styles; registering event listeners; adding, removing or sorting nodes; and changing HTML or text content. These suffice for the vast majority of needs. Direct access to the underlying DOM is also possible, as each D3 selection is simply an array of nodes.

2.3.2 Manipulating Data

Using D3's enter and exit selections, you can create new nodes for incoming data and remove outgoing nodes that are no longer needed.

When data is bound to a selection, each element in the data array is paired with the corresponding node in the selection. If there are fewer nodes than data, the extra data elements form the enter selection, which you can instantiate by appending to the enter selection. A common pattern is to break the initial selection into three parts: the updating nodes to modify, the entering nodes to add, and the exiting nodes to remove.

By handling these three cases separately, you specify precisely which operations run on which nodes. This improves performance and offers greater control over transitions. For example, with a bar chart you might initialize entering bars using the old scale, and then transition entering bars to the new scale along with the updating and exiting bars.

D3 lets you transform documents based on data; this includes both creating and destroying elements. D3 allows you to change an existing document in response to user interaction, animation over time, or even asynchronous notification from a third-party. A hybrid ap-

proach is even possible, where the document is initially rendered on the server, and updated on the client via D3.

2.4 GeoJSON

GeoJSON is an open standard format designed for representing simple geographical features, along with their non-spatial attributes, based on JavaScript Object Notation [2].

The features include points (therefore addresses and locations), line strings (therefore streets, highways and boundaries), polygons (countries, provinces, tracts of land), and multi-part collections of these types. GeoJSON features need not represent entities of the physical world only; mobile routing and navigation apps, for example, might describe their service coverage using GeoJSON.

2.5 TopoJSON

TopoJSON is an extension of GeoJSON [3]. TopoJSON introduces a new type, "Topology", that contains GeoJSON objects. A topology has a map of objects which indexes geometry objects by name. These are standard GeoJSON objects, such as polygons, multi-polygons and geometry collections. However, the coordinates for these geometries are stored in the topology's arcs array, rather than on each object separately. An arc is a sequence of points, similar to a line string; the arcs are stitched together to form the geometry. Lastly, the topology has a transform which specifies how to convert delta-encoded integer coordinates to their native values (such as longitude and latitude).

2.6 ElasticSearch

Elasticsearch is a search server based on Lucene. It provides a distributed, full-text search engine with an HTTP web interface and schema-free JSON documents. Elasticsearch is distributed, which means that indices can be divided into shards and each shard can have zero or more replicas. Each node hosts one or more shards, and acts as a coordinator to delegate operations to the correct shard(s) [3].

[2] http://geojson.org/
[3] https://github.com/mbostock/topojson

2.6.1 Searching

The search API allows you to execute a search query and get back search hits that match the query. The query can either be provided using a simple query string as a parameter, or using a request body.

As a parameter:

```
http://localhost:9200/footballdata/tweet/_search?q=text:Griezmann
```

As a request body:

```
http://localhost:9200/footballdata/tweet/_search' -d '{
"query" : {
    "term" : { "text" : "Griezmann" }
  }
}
```

2.6.2 Adding data

There are two main ways for adding data to an Elasticsearch index, the first one is based on using Logstash to capture real-time data. In this case a configuration file is needed, in this file is described where data is going to be stored and the query terms that are going to be indexed.

Logstash example for catching real-time tweets:

```
input {
  twitter {
    consumer_key => "02fW91zy4FggYNZ9XtjJA1ZkQ"
    consumer_secret => "YY8XWQiYZyxg0BgRYJKe83RVGCkYnUCwGXKAQoEL7Du8IWCIB5"
    oauth_token => "377869454-qWPHBtpviVfiIV2dcbLukgLEGXlBWxq63J05vftd"
    oauth_token_secret => "7rottQoomqgZHChkkjntrzVBkuWdPGlrB7mIDT3GwzV7t"
    keywords => [ "#FCBAtleti" ]
    full_tweet => true
  }
}
filter {
}
output {
    stdout { codec => dots }
    elasticsearch {
    hosts => "http://localhost:9200"
    index => "twitter_football"
```

```
    document_type => "tweet"
    template => "../elasticsearch-config/twitter_template.json"
    template_name => "twitter"
  }
}
```

The other way is much easier, it consists in adding a whole index using the Elasticsearch's bulk API. This can greatly increase the indexing speed. In this case you need all your data stored in a JSON file. You manually add the index where data is going to be stored.

```
$ curl -s -XPOST localhost:9200/footballdata/tweet/_bulk --data-binary "
    @data.json"
```

2.7 SPARQL

SPARQL, acronym of SPARQL Protocol and RDF Query Language, is a RDF query language, this means is a semantic query language for databases, it is used to retrieve data stored in Resource Description Framework (RDF) format [5].

It was made an official W3C Recommendation in 15 January 2008, SPARQL queries allows conjunctions, disjunctions, and optional patterns.

The result can be represented in many formats, such as HTML, XML, JSON ...
SPARQL query example that retrieves all teams belonging to La Liga whose name is like 'Madrid':

```
PREFIX p: <http://dbpedia.org/property/>
PREFIX o: <http://dbpedia.org/ontology/>
PREFIX r: <http://dbpedia.org/resource/>
SELECT * WHERE {
  ?teams o:league  <http://dbpedia.org/resource/La_Liga> .
  FILTER regex(str(?teams), 'Madrid','i') .
}
```

CHAPTER 3

Architecture

3.1 Introduction

In this chapter, we will explain the architecture of this project, including the design phase and implementation details. First of all, in the overview we will present a global vision about the project architecture, identifying the visualisation server and other necessary modules. Secondly, we will focus on each module explaining its purpose in this project. Finally, we will present the pipeline in charge of data mining.

3.2 Overview

In this section we will present the global architecture of the project, defining the different subsystems that participates in the project. We can identify the following subsystems:

- ***Visualisation system:*** This is the main part of the project, it will be able to process data and show it in different views. This visualisation server is developed from Sefarad 3.0.
- ***Search and indexing system:*** This system is composed of ***Elasticsearch*** [3] a

searching server in charge of providing necessary data to the visualisation server. Also is composed of a tool for real-time tweet collection called ***Logstash*** [1]. Finally, in this system we can include ***Twitter*** [2] which provides data to be analysed and represented.

- ***Orchestrator:*** We use ***Luigi*** [7] as orchestrator to build pipelines through our search and indexing system and the analytic services, in order to facilitate sentiment analysis.

- ***Analytic services:*** In this project we have used two analytic services. On one hand we have ***Senpy*** [2] used to analyse sentiments and ***Nerdy*** [6] necessary to recognise entities in different tweets.

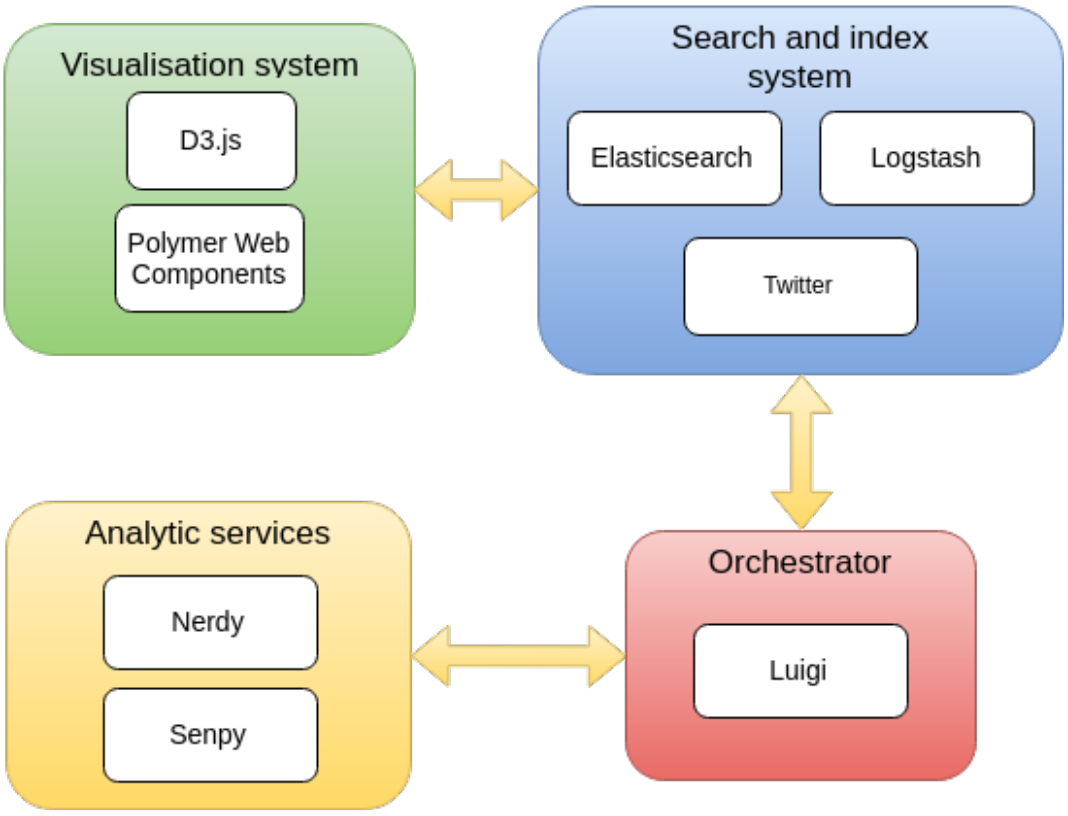

Figure 3.1: Architecture

In the following sections we are going to describe deeply subsystems involved in the project.

[1] https://www.elastic.co/products/logstash
[2] http://twitter.com

3.3 Searching and indexing system

Logstash service catches every data from Twitter and stores it in an Elasticsearch index. There are some configurable parameters:

- ***Input:*** Here we configure Twitter app keys and the keyword or hashtag(#) we want to get data from.
- ***Output:*** In this parameter we configure where to store the data collected: The host, the index and the template used for data storage.

3.4 Orchestrator

Luigi is a Python module that helps you build complex pipelines of batch jobs [7]. It handles dependency resolution, workflow management, visualization etc. This module needs a script describing the pipeline to follow.

3.5 Analytic services

Senpy is sentiment and emotion analysis server in Python developed by the Intelligent Systems Group, for this task we have used two plug-ins developed by Ignacio Corcuera [2]:

- ***sentiText plug-in:*** This one is used for sentiment analysis. It distinguishes between positive, neutral or negative sentiment.
- ***EmoTextANEW plug-in:*** This other one is used for emotion analysis. Emotions available are anger, disgust, fear, joy, neutral and sadness.

NERDY is an acronym for Named Entity Recognition Server in Python, this analyser has been developed by Constantino Román [6]. We have used it to get the entities that are mentioned in a tweet to make the analysis and filtering easier. The library used for this task was polyglot-es.

We are going to describe below the pipeline created for data mining, that is the existing relations between subsystems.

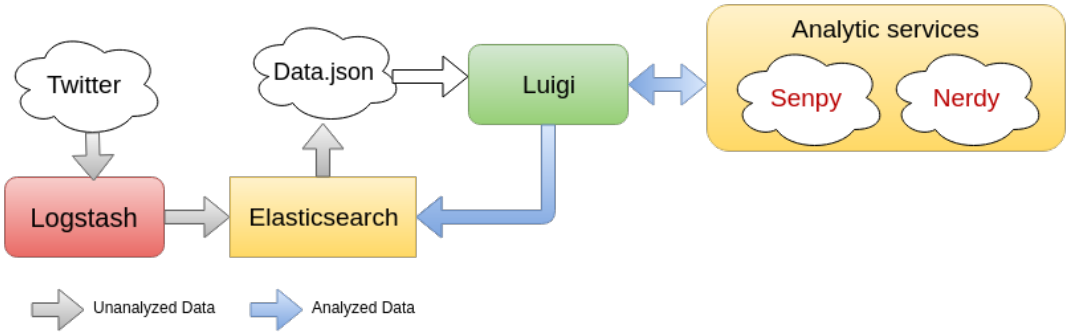

Figure 3.2: Pipeline Phases

3.6 Pipeline

Nowadays, Twitter has become an interesting scenario for social analysis. We can find in this micro-blogging site millions of interactions between users. As Twitter posts (tweets) are short and are being generated constantly, they are well-suited sources of streaming data for opinion mining and sentiment polarity detection. Also Twitter provides news about the person who is posting them: commentary on links, directed discussion, location information, status or any other content. Moreover, Twitter does not distinguish between celebrities and other people, even if users belong to different ideology groups. Twitter has representation in many countries so it provides a global view about the topic. This is why we have used Twitter as source of information. We are going to present the phases and technologies involved during data mining process below. In figure 3.2 we can see the phases of the whole process.

3.6.1 Logstash

This is the first phase of the pipeline. In this step, is necessary to define a strategy to follow. We need tweets related to football, so the strategy is to run Logstash during important match events. For this project we have chosen the FC Barcelona - Atlético de Madrid match, which was the quarter final of the UEFA Champions League. Logstash was configured to catch every tweet belonging to the hashtag (#)FCBAtleti and store them in an Elasticsearch index.

3.6.2 Elasticdump

After the previous phase has been completed data is stored in an Elasticsearch index. Logstash stores many fields that are not necessary for this project so we are going to filter the

index. Fields required for this project are “user.location”, “text”, “id” and “@timestamp”.

This task is done with elasticdump [8], this tool allows us to export data from Elasticsearch to a JSON file. The command used was:

```
elasticdump \
--input=http://localhost:9200/twitter_football \
--output=Data.json \
--searchBody '{fields:["text", "user.location", "@timestamp", "id"]}'
```

The result was a file called Data.json with only necessary fields.

3.6.3 Sentiment and emotion analysis

This phase has been automatized using the orchestrator, we have described tasks using a script called sefarad.py for sentiment analysis that has been developed by Enrique Conde. Tasks described in this script are: ***FetchDataTask***, ***SenpyTask***, ***NerdyTask*** and ***ElasticsearchTask***

We had to adapt this script to be able to also analyse emotions creating another one called sefarad-emo.py that has the same tasks as the described above. Now we are going to deeply describe each task.

3.6.3.1 FetchDataTask

The main goal of this task is to read the JSON file. This is the Python code describing this task:

```
class FetchDataTask(luigi.Task):
"""
Generates a local file containing 5 elements of data in JSON format.
"""

#: the date parameter.
date = luigi.DateParameter(default=datetime.date.today())
field = str(random.randint(0,10000)) + datetime.datetime.now().strftime("%Y
    -%m-%d-%H-%M-%S")

def run(self):
"""
Writes data in JSON format into the task's output target.
The data objects have the following attributes:
* `_id` is the default Elasticsearch id field,
```

```
* 'text': the text,
* 'date': the day when the data was created.
"""
today = datetime.date.today()
with open('Data.json') as f: #Here enter the path to your JSON file
j = json.load(f)
for i in j:
i["_id"] = i["id"]

with self.output().open('w') as output:
json.dump(j, output)
output.write('\n')

def output(self):
"""
Returns the target output for this task.
In this case, a successful execution of this task will create a file on the
    local filesystem.
:return: the target output for this task.
:rtype: object (:py:class:'luigi.target.Target')
"""
return luigi.LocalTarget(path='/tmp/_docs-%s.json' % self.field)
```

3.6.3.2 SenpyTask

This task loads data fetched with previous task and send it to Senpy tool in order to analyse data retrieved and check sentiments expressed. In this process we can see the main difference between both scripts: Sefarad.py makes requests to Senpy's sentiText plug-in and the other one to Senpy's EmoTextANEW plug-in. Response is filtered and we get the sentiment or emotion and store it in a JSON file.

3.6.3.3 NerdyTask

This task loads data fetched with previous task and send it to Nerdy tool in order to analyse data retrieved and recognise names and entities. This is the Python code describing this task:

```
class NerdyTask(luigi.Task):

#: date task parameter (default = today)
date = luigi.DateParameter(default=datetime.date.today())
```

```
file = str(random.randint(0,10000)) + datetime.datetime.now().strftime("%Y
    -%m-%d-%H-%M-%S")

def requires(self):
"""
This task's dependencies:
* :py:class:`~.FetchDataTask`
:return: object (:py:class:`luigi.task.Task`)
"""
return FetchDataTask()

def output(self):
"""
Returns the target output for this task.
In this case, a successful execution of this task will create a file on the
     local filesystem.
:return: the target output for this task.
:rtype: object (:py:class:`luigi.target.Target`)
"""
return luigi.LocalTarget(path='/tmp/analyzed-%s.jsonld' % self.file)

def run(self):
"""
Send data to Wrapper tool and retrieve it analyzed. Store data in a json
    file.
"""
with self.output().open('w') as output:
with self.input().open('r') as infile:
j = json.load(infile)
for i in j:
r = wrapper.service(u"%s" % i["text"], "polyglot-es")
i["_id"] = i["id"]
i["entities"] = r[0]
output.write(json.dumps(i))
output.write('\n')
```

3.6.3.4 ElasticsearchTask

This task loads JSON data contained in the file produced in the previous step into an Elasticsearch index. You set the index and doc-type data when you call function Luigi, we have used footballdata index for sentiment analysed tweets:

```
luigi --module sefarad Elasticsearch --index footballdata
--doc-type tweet
```

And we have used footballemotion index to store emotion analysed tweets:

```
luigi --module sefarad-emo Elasticsearch --index footballemotion
--doc-type tweet
```

CHAPTER 4

Visualisation Server

4.1 Introduction

The main goal of this module is to show football sentiment analysis to users. This visualisation server is based on Polymer Web Components Library [1]. This library is based on Google's material design and allows the creation of reusable widgets or components in web documents and web applications [4].

We have also used D3.js library to design some new Web Components that provide interactive, dynamic data visualizations in web browsers [1].

4.2 Analysis

There is much information in the Internet about football. Although many websites show statistics or results of matches, there is not yet any available resource that publishes information regarding their sentiment or emotion. We decided to make an interactive dashboard to present data we have collected from Twitter. The main steps to get to this dashboard

[1] https://www.polymer-project.org/1.0/

where research other sites and make a mock up.

4.2.1 Review of online visualisation dashboards for sentiment analysis

In this step we have focused on one hand, in many websites that show sentiment analysis such as iRedes [1], also we had a look at "la lista wip" to see a managed ranking of celebrities and we discover an interactive tool from PubNub x Twitter [2] that shows North America's emotions. On the other hand, we looked for dashboards, in websites like Templateflip [3] to make us an idea of what widgets we wanted to create, and see structured data. Also in this step we made a research across D3.js library [9] to discover new ideas and transform some of them to our own widgets.

4.2.2 Mock-up

After doing all the research we decided to use the on-line tool draw.io [4] to make some mock-ups of the dashboard with the ideas we had collected. We decided not to have a sidebar because it hides many information and we wanted to make it easily accessible, for this reason we decided to make a tab division in the top bar. In figure 4.1 we can see a mock-up of the first tab, in this one we are going to display information about football events.

In the second tab we are going to show information relating to teams, so we are going to replace the sentiment chart with the most influential teams as we can see in the mock-up, figure 4.2.

In figure 4.3 is shown the mock-up of players tab, in this tab we display the most influential players instead of teams.

[1] http://iredes.paradigmatecnologico.com/
[2] http://pubnub.github.io/tweet-emotion/
[3] https://templateflip.com/bootstrap-admin-templates/
[4] http://draw.io

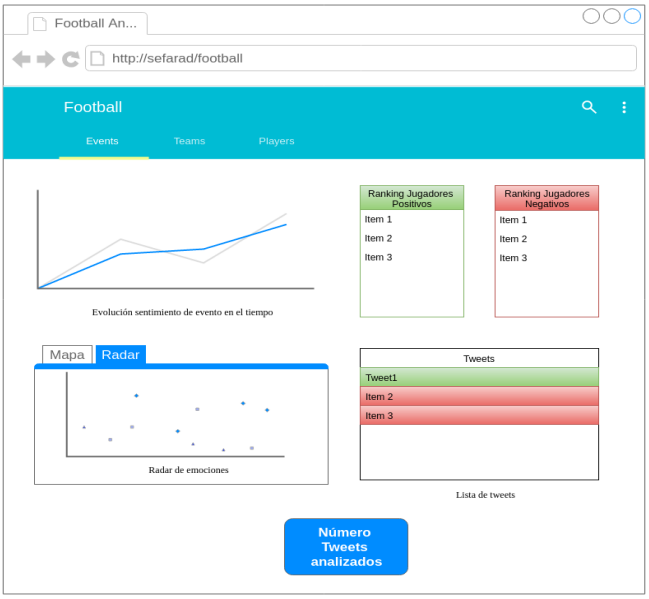

Figure 4.1: Events tab mock-up

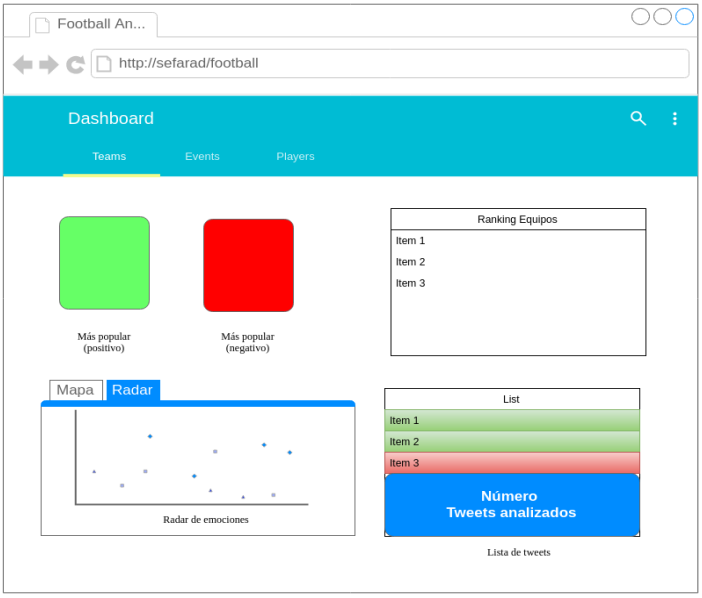

Figure 4.2: Teams tab mock-up

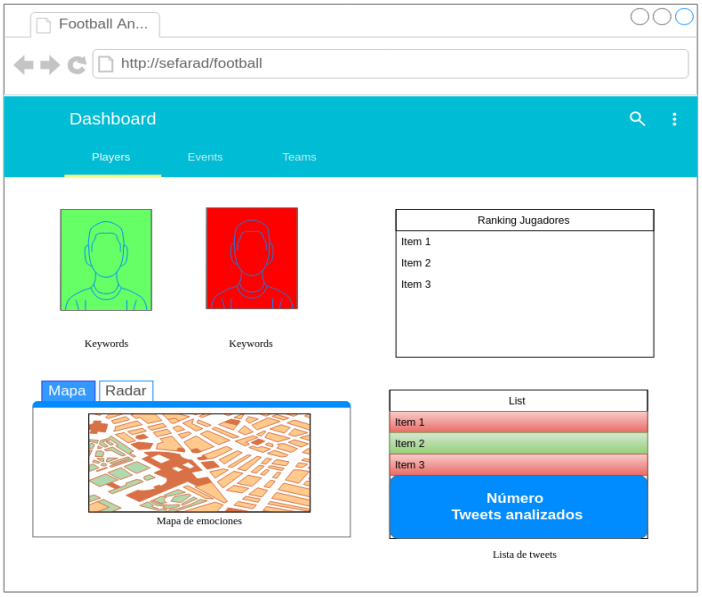

Figure 4.3: Players tab mock-up

4.3 Widgets

Many of these Web Components created are widgets for data visualisation, we have used D3.js to make them.

Each widget provides us different information based on tweet's data. Some of these widgets are described below.

4.3.1 Player Ranking

This widget is used for getting the most influential players, it can show the most positive or the most negative players. The parameters accepted are:

- Index and Subindex (required): Indicates in which Elasticsearch's database are you querying.
- Title (required): This title will be displayed in the top widget bar. It can be positive or negative and select which ranking you want.
- Color: It is for personalise bubble colour.
- Query: This parameter is auto filled with search box.

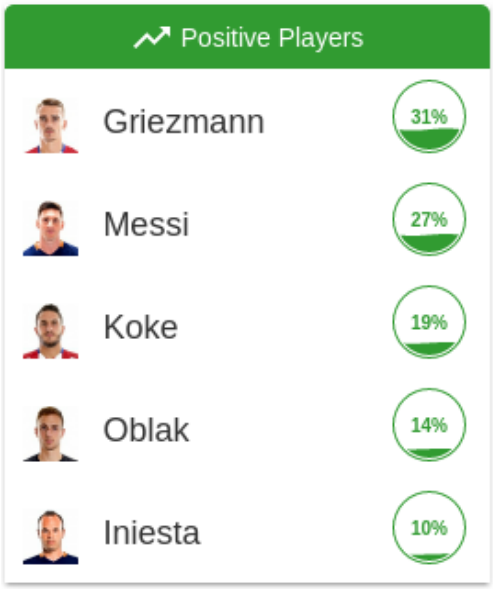

Figure 4.4: Player ranking widget

After you have completed all required parameters it makes a query to Elasticsearch database. Data received is filtered and only the top ones are shown as we can see in figure 4.4.

4.3.2 Team Ranking

This widget is quite similar to the player ranking one. It shows the most influential teams instead of showing players. The parameters accepted are the same as player ranking widget (4.3.1).

4.3.3 Sentiment Map

Many of these tweets we have collected and analyse can be located thanks to user's location. We have used this location to place tweets in a map of Spain. This map has been designed with D3.js and using a Spain coordinates' TopoJSON file. The parameters accepted are:

- Index and Subindex (required): Indicates in which Elasticsearch's database are you querying.
- Query: This parameter is auto filled with search box.
- Type: It is used for selecting which type of entities we want to display, players or teams.

After querying to database data received are grouped by city and predominant sentiment present in each city is calculated. Tweets are represented with emojis and also each province is coloured according to sentiment.

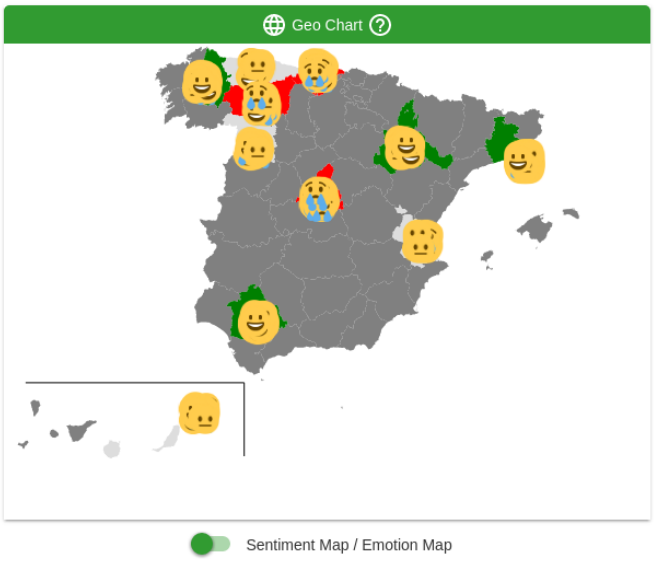

Figure 4.5: Map widget

4.3.4 Emotion Map

It's quite similar to the sentiment map (4.3.3), instead of showing sentiments in the map this one is designed to show emotions. The parameters accepted are:

- Index and Subindex (required): Indicates in which Elasticsearch's database are you querying.
- Query: This parameter is auto filled with search box.
- Type: It is used for selecting which type of entities we want to display, be players or teams.

After querying to database data received are grouped by city and predominant emotion present in each city is calculated. Tweets are represented with emojis and also each province is coloured according to its emotion.

4.3.5 Emotion Radar

This widget is designed to study the emotion present in tweets. The axis in radar describes each analysed emotion. The coloured area shows the amount of tweets pertaining to different emotions. The parameters accepted are:

- Index and Subindex (required): Indicate in which Elasticsearch's database are you querying.
- Query: This parameter is auto filled with search box.
- Type: It is used for selecting which type of entities we want to display, players or teams.

4.3.6 Tweet Chart

This widget is used for showing the latest tweets available. The parameters accepted are:

- Index and Subindex (required): Indicate in which Elasticsearch's database are you querying.
- Query: This parameter is auto filled with search box.

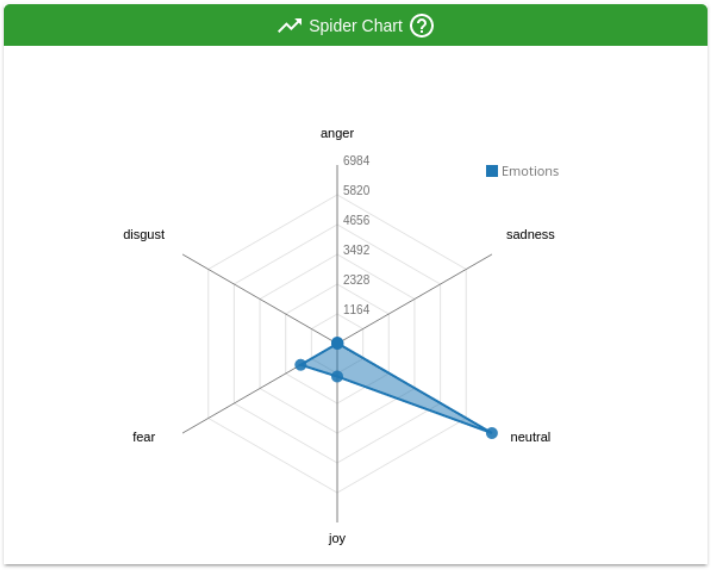

Figure 4.6: Emotion Radar widget

- Type: It is used for selecting which type of entities we want to display, players or teams.

Data received is presented on a list, tweet's background is coloured according to each tweet sentiment green colour portrays a positive tweet, red colour represents a negative tweet and grey colour means a neutral tweet.

Figure 4.7: Tweet chart widget

4.3.7 Number Widget

We have created this widget to represent the number of tweets (corpus) studied. This data is real time updated and is located in the main bar. The parameters accepted by this widget are:

- Index and Subindex: This is required to get the total number of tweets in each index.
- Query: This parameter is auto filled with search box to determine the number of tweets being analysed and studied.

4.3.8 Sentiment Chart

This widget is used to study sentiment produced during events, x axis describes the time and in y axis we can see the sentiment polarity for each lapse of time. The parameters accepted by this widget are:

- Index and Subindex: This is required to get the total number of tweets in each index.
- Query: This parameter is auto filled with search box to determine the number of tweets being analysed and studied.
- Title: This title will be displayed in the top widget bar.

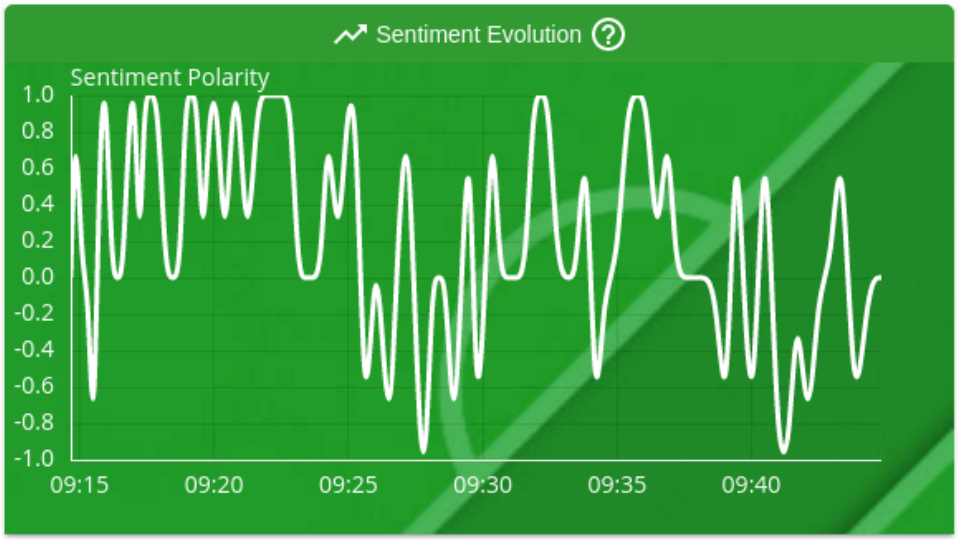

Figure 4.8: Sentiment chart widget

4.3.9 Top Team

This widget is used to show which is the most influential teams, it gets the positive and the negative one. For each one is displayed the amount of positive, negative and emotions tweets. Team's badge is also shown. The parameters accepted are:

- Index and Subindex: This is required to get the total number of tweets in each index.
- Query: This parameter is auto filled with search box to determine the number of tweets being analysed and studied.
- Title: This title will be displayed in the top widget bar. This parameter is required, it can be positive or negative and select which one you want.

Figure 4.9: Top team widget

4.3.10 Top Player

It is the same widget as top team one (4.3.9) but in this case we substitute the most popular positive and negative team, for the most popular players.

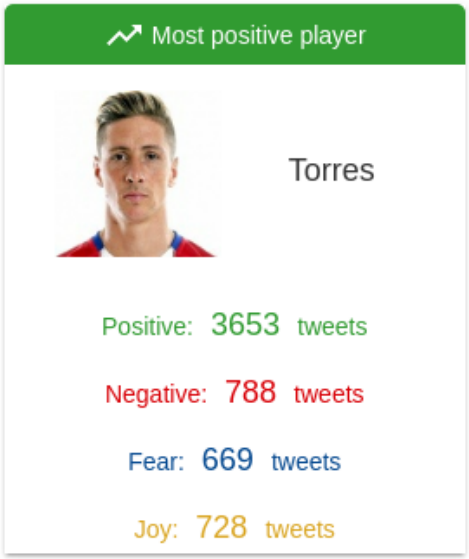

Figure 4.10: Top player widget

4.4 Dashboard Tabs

This visualisation server uses a tab division, each tab is designed for showing different type of data. Tabs available are in visualisation server are ***events tab***, ***teams tab***, ***players tab***, ***about tab*** and ***SPARQL tab.*** Now we are going to focus in defining them, describing each tab's components and purpose.

4.4.1 Events tab

This tab is designed for showing to the user data produced by football events. We have made a selection of components to make this data easy to understand. Those we have selected for this view are:

- Sentiment chart, which is used to see sentiment evolution during the match.
- Positive player ranking, where top 5 players with more positive tweets are shown, is also shown the percentage each player gets of the total amount of tweets.
- Negative player ranking, shows the same as positive one but this time are shown top 5 players with more negative tweets.
- Toggle window, we have used this toggle to make data more accessible and to avoid scroll. We can find two widgets here:

  - Sentiment map, for monitoring where the event have more impact and show each province sentiments.
  - Emotion map, for monitoring where the event have more impact and show each province emotions.

- Emotion radar, help us to have a quick view of the whole event present emotions.
- Tweet chart, gives us a detailed view of the most recent tweets.

4.4.2 Teams tab

This view is focused in data based on teams. It is quite similar to events tab but data showed is totally different. The components selected for this view are:

- Top positive team, in this widget it is displayed the most popular team's badge and some sentiment and emotion statistics.
- Top negative team, is the same as the above one but displaying the most negative team data.
- Team ranking, where top 5 teams with more positive tweets are shown, is also shown the percentage each team gets of the total amount of tweets.
- Toggle window, we have used this toggle to make data more accessible and to avoid scroll. We can find two widgets here:
  - Sentiment map, for monitoring where the event have more impact and show each province sentiments.
  - Emotion map, for monitoring where the event have more impact and show each province emotions.
- Emotion radar, help us to have a quick view of the whole teams emotions.
- Tweet chart, gives us a detailed view of the most recent tweets.

4.4.3 Players tab

In this tab we are going to show data about players, to present this data we are going to use the same widgets as in teams tab but changing top teams widget for top players widget:

- Top positive player, in this widget is displayed the most popular player's photograph and some sentiment and emotion statistics.
- Top negative player, is the same as the above one but displaying the most negative player data.
- Player ranking, where top 5 players with more positive tweets are shown, is also shown the percentage each player gets of the total amount of tweets.
- Toggle window, we have used this toggle to make data more accessible and to avoid scroll. We can find two widgets here:
  - Sentiment map, for monitoring where the event have more impact and show each province sentiments.
  - Emotion map, for monitoring where the event have more impact and show each province emotions.
- Emotion radar, help us to have a quick view of the whole players emotions.
- Tweet chart, gives us a detailed view of the most recent tweets talking about players.

4.4.4 SPARQL editor tab

In this view we have added a SPARQL editor, we have used YASGUI which is a web-based editor. This editor is used to load existing queries or create new ones. We have used it to explore football data stored in the dbpedia [2] because is one of the largest endpoints.

[2] http://dbpedia.org/sparql

CHAPTER 5

Case study

5.1 Introduction

This chapter describes the process we have follow for data mining, the results we have obtained of data analysis, the visualisation implementation and how does the system work. We will also present the main conclusion of this thesis.

5.2 Collecting data

We have used Twitter as source of information for this project. We have distributed data collection in different phases, first of all we decided which event will our subject of study. The event selected was the quarter finals of the UEFA Champions League 2015-2016, this match brings Fútbol Club Barcelona and Atlético de Madrid together. It was played on Tuesday, april 5^{th} at 8:45pm and the match was held in Camp Nou stadium, where Fútbol Club Barcelona plays as home team.

Secondly, we configured Logstash service to catch all the tweets containing the hashtag (#)FCBAtleti, which was the official Spanish hashtag for this match on Twitter. In our strategy was to analyse between one hour before and half one hour after the match, so

Logstash was working from 7:45pm to 11pm. In this lapse of time we obtained 99336 tweets.

These tweets has been analysed using Senpy [2]. On one hand sentiment analysis results were 65693 neutral, 7058 negative and 26615 positive tweets. On the other hand emotion analysis results were 16332 negative-fear, 12709 joy, 109 disgust, 395 sadness, 468 anger and 69353 neutral emotion tweets. We also analysed data with Nerdy [6], we obtained 31105 tweets talking about Fútbol Club Barcelona and 18334 about Atlético de Madrid. We also used Nerdy to obtain the most influential players, the results were 15944 tweets talking about Torres and 10728 about Luis Suárez, far from this two is Messi the third most influential player with 3853 tweets. This data has been stored in an Elasticsearch index, analysing and storing data has been done by Luigi tasks.

5.3 Displaying data

The data is displayed using a dashboard based on tabs and widgets, the main tab showed when a user opens the site is the events tab, in this tab we have focused in showing sentiment and emotion analysis for a specific event. In figure 5.1 we can see the events tab screen, and the widgets it is composed of.

The ***sentiment evolution widget***, shows the sentiment polarity evolution during part of a match. In the development of this widget we have used all tweets we have collected and we have filtered them by time. The result has been the sentiment polarity in 30 minutes, the user can check the average sentiment polarity in each minute.

In sport events is also important which are the most influential players, for doing so we have created two widgets, the ***positive ranking*** and the ***negative one***. The development consisted in getting all tweets referring to players, and checking each tweet sentiment. After we have checked all of them, we created a list with all players and the amount of positive and negative tweets referring to each player. Finally, we ordered the list according to the amount of positive tweets if we wanted the positive ranking, or according to the amount of negative tweets. We only show the top five of each list. The results we obtained were that the positive ranking was formed by: Fernando Torres, Luis Suárez, Claudio Bravo, Lionel Messi and Neymar Jr. On the other hand, the negative one was formed by: Lionel Messi, Fernando Torres, Sergio Busquets, Luis Suárez and Neymar Jr.

Moreover, we considered that was important to geolocalize the users that were posting tweets. We decided to design a map of Spain with two variations: the ***sentiment map***

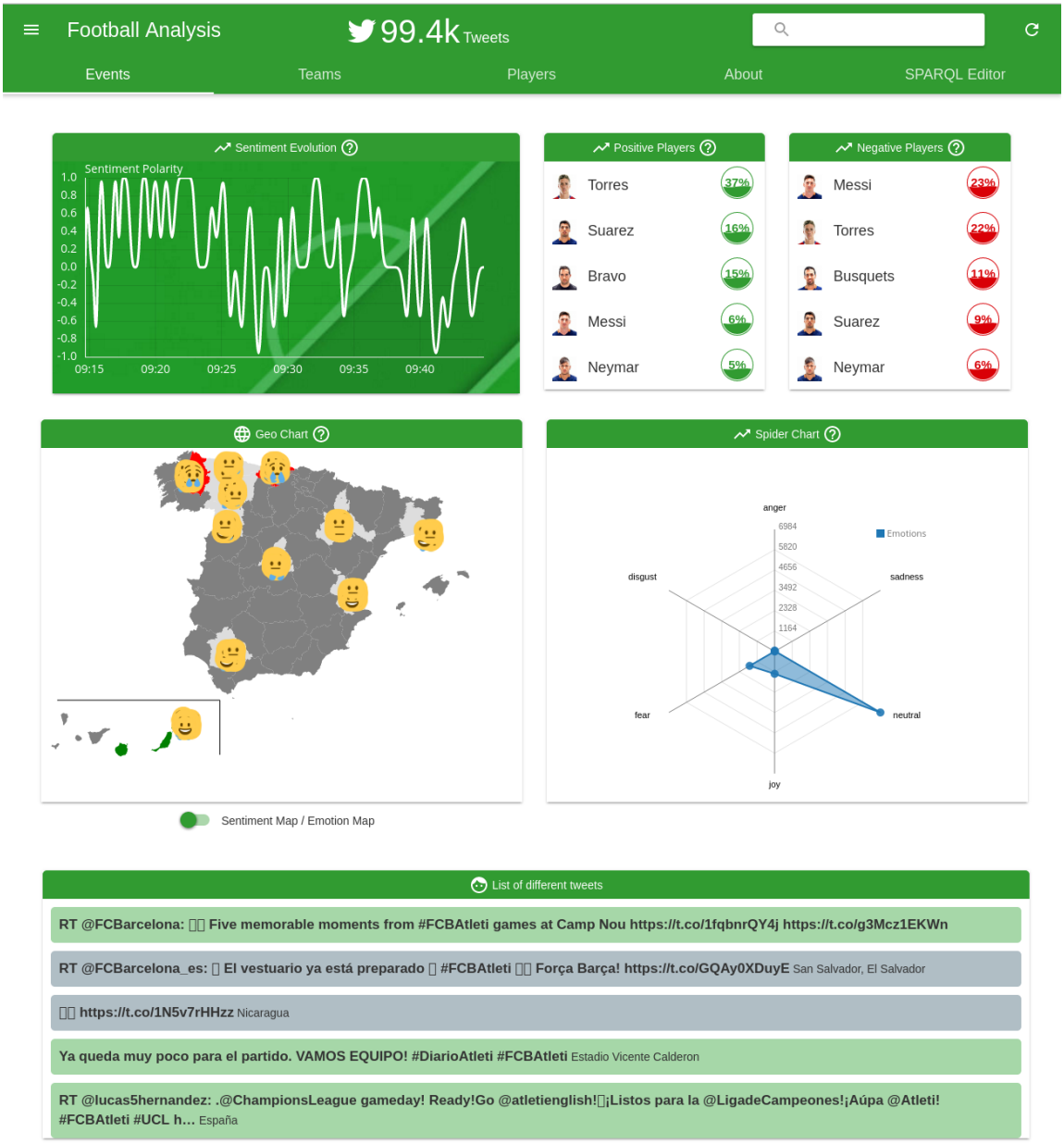

Figure 5.1: Events tab

and the ***emotion map***. The development was nearly the same for both maps and it had two phases, firstly we designed a map of Spain using TopoJSON. Once we had the map drawn, we collected all tweets and grouped them by city name. Finally, we drew emoji faces according to each tweet sentiment or emotion, and coloured the province according to the predominant sentiment or emotion.

We have also added an ***emotion radar***, we thought it was interesting to show each emotion amount of tweets. In this widget the area coloured represents the number of tweets. In the developing we have used all tweets, we processed them and create a number variable for each emotion. The result was an array of numbers, this array is used to draw the radar axis representing an emotion.

Finally, we decided it was relevant to show the tweets for this purpose we created the ***tweet chart***. The development consisted in process the 5 latest tweets stored in Elasticsearch, this was done adding sort filtering in the query. Finally, we create a method for checking each tweet sentiment. This method is also responsible of colouring the tweet background according to tweet's sentiment.

In figure 5.2 is shown the teams tab, there are common widgets with the events tab, the development was the same, but instead of collecting all tweets available we have focused in those referring to teams.

In this view sentiment chart has been replaced by the ***most positive*** and the ***most negative team***, the development of this widget was quite similar as player rankings. We collected all tweets referring to teams and we created a list with all teams and the amount of positive and negative tweets referring to each team. After we have sorted them by positive or negative number of tweets, the name of the team is printed and the amount of positive or negative tweets. We considered also important to show the predominant emotion so we also printed predominant emotion's number of tweets. The results obtained were 8060 positive, 1550 negative, 823 negative-fear and 75 joy tweets for Fútbol Club Barcelona, in the other side is the Atlético de Madrid with 7926 positive, 1746 negative, 2175 negative-fear and 1584 joy tweets.

We have also replaced the player ranking by the ***team ranking***, the development was the same as the player ranking. We obtained similar results for both teams so the ranking reflect a fifty percent for each team.

In figure 5.3 is presented the players tab, as is said before there are some common widgets between this and the events tab, in this case we have only selected tweets referring to players.

As with the teams tab, we have replaced the sentiment chart by ***most positive*** and the ***most negative player*** widget. This widget is an extension of player ranking widget, the development was the same as the most positive and most negative teams. In this case the results were: the most positive player was Fernando Torres with 3653 positive, 788 negative, 669 negative-fear and 728 joy tweets. The negative one was Lionel Messi with 612 positive,

Figure 5.2: Teams tab

823 negative, 177 negative-feat and 454 joy tweets.

It is common in every tab the search box in the top bar, this tool allow us to focus in an specific aspect: yellow cards, red cars, referee, etc. Also is present the real time updated number of tweets analysed in this bar.

The last tab in this project is the SPARQL editor tab, this view is shown in the figure 5.4. Here we have added a SPARQL editor to query the dbpedia, there are some preconfigured

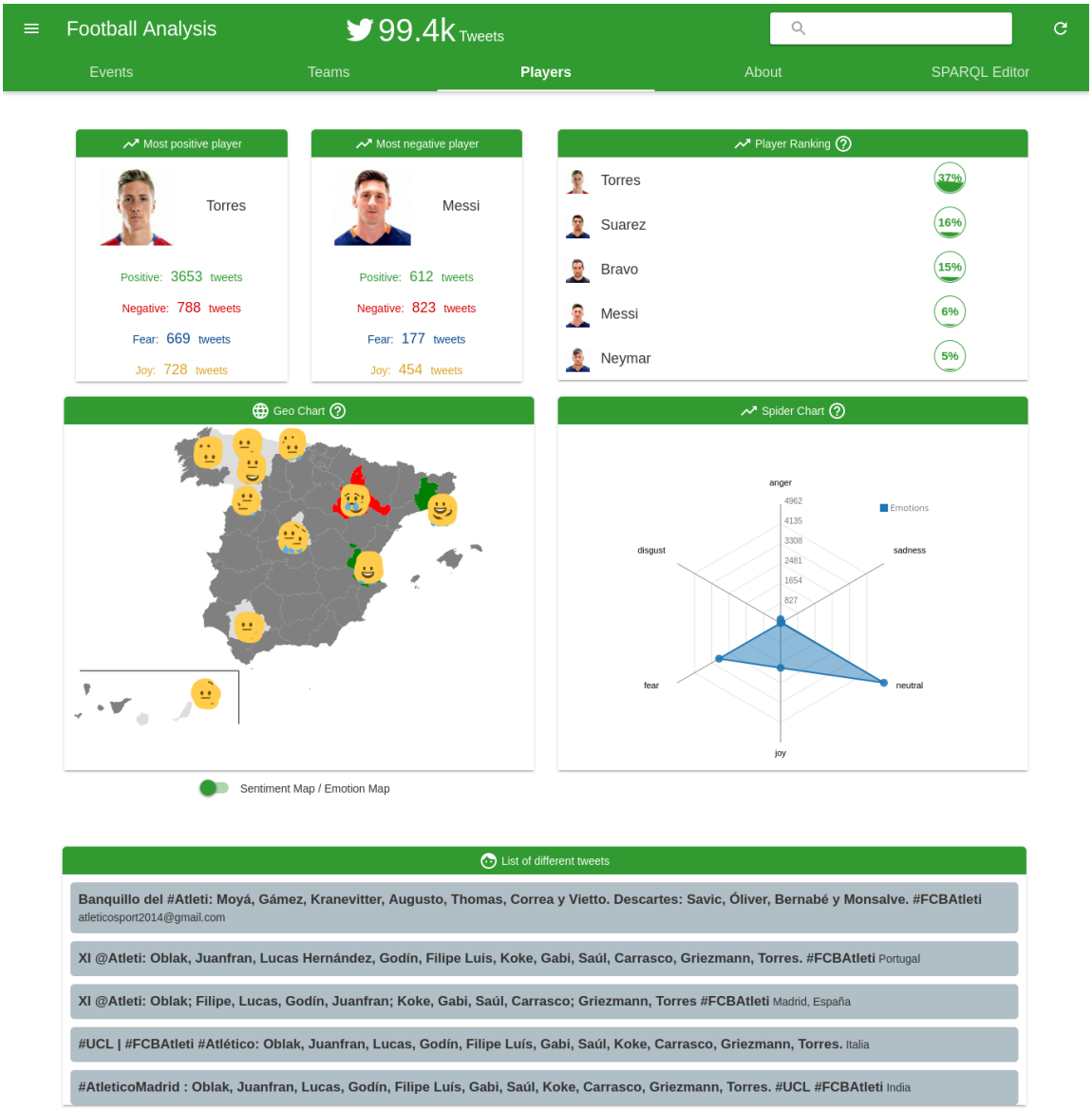

Figure 5.3: Players tab

queries such as birthplace, number of teams, club number and position of Real Madrid players ordered by club number. The result can be displayed in a table, in a Google Chart or in RAW format.

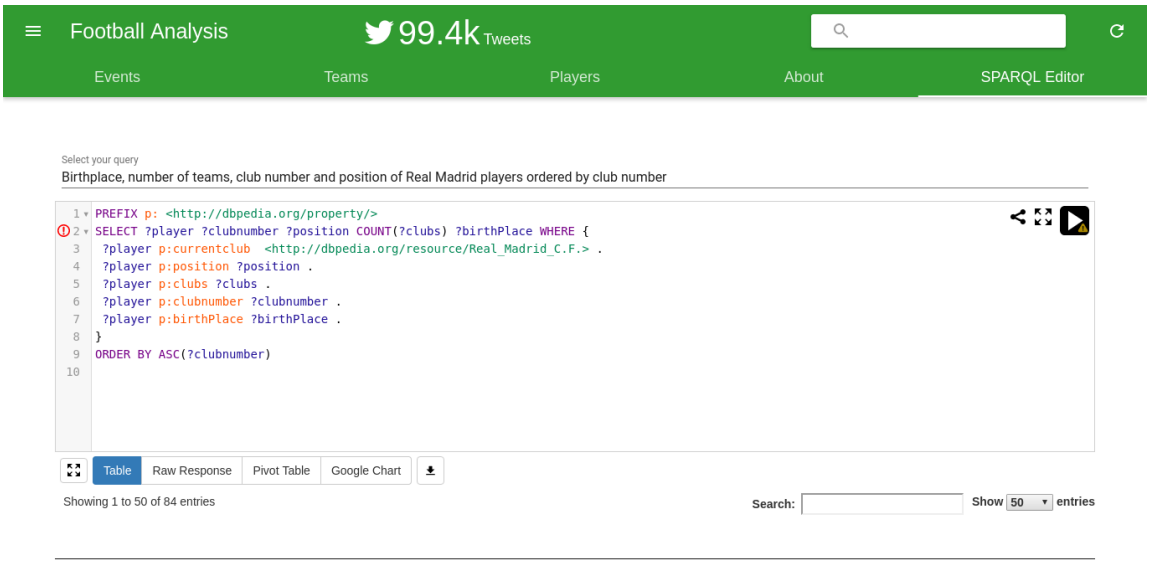

Figure 5.4: SPARQL tab

5.4 Conclusions

In this chapter we have presented the steps we have followed for data mining and analysing. We have explained how were the widgets and the different views developed, and also how the website could be used.

To sum up, we have presented the analysis' results, on one hand we have checked that the Fútbol Club Barcelona is still the most popular amongst teams, with 63 percent of the tweets referring to teams talking about it. In the other hand, we have verified that Fernando Torres generated most of tweets referring to players due to his red card in this match. In addition, filtering results by the role played by the referee in this match: in Barcelona it produced a positive sentiment, but in Madrid was the opposite.

CHAPTER 6

Conclusions and future work

6.1 Introduction

In this chapter we will describe the conclusions extracted from this thesis, problems, achievements and suggestions about future work.

6.2 Conclusions

In this project we have created a dashboard for sentiment and emotion analysis of football in Twitter based on Web Components y D3.js to make data accessible and interactive. This dashboard is based on widgets and provides a lot of information at a glance.

This project is formed by 4 subsystems, the main one is the visualisation system described above. We have also used an orchestrator, which is a module for sequencing and executing tasks and dependencies. This orchestrator is in charge of the indexing data service and the analysing services.

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

6.3 Achieved goals

In the following section I will explain the achieved features and goals that are available in this project.

Collect tweets focused in football events We needed to obtain tweets referring to football matches, this goal has been achieved thanks to Logstash. This service provided us this real time data during matches. This feature was crucial because it supplies data that has been used by the other systems in this project.

Build a pipeline for sentiment and emotion analysis All data collected was unanalysed, so we needed a service that provides sentiment and emotion analysis. Luigi helped us with this goal: we used it as an orchestrator. It provided a pipeline between Senpy service and Elasticsearch, so analysed data was stored in an Elasticsearch index.

Named Entity Recognition (NER) In our project was important to know what were tweets talking about. Nerdy provided the entities recognition, so we used the orchestrator to provide a pipeline between Nerdy service and Elasticsearch. Analysed data updated the Elasticsearch index.

Design a dashboard based on widgets This was the main goal of the project, we had to develop a dashboard. This dashboard is based on new technologies like Polymer Web components. Despite this, this goal has been accomplished and data visualisation is interactive.

6.4 Problems faced

During the development of this project we had to face some problems. These problems are listed below:

- Map development: We needed a map of Spain for positioning tweets, it was quite hard to obtain a map with Canary province on it. These islands have different coordinates because they are too south from the mainland, so we needed to transform the polygon coordinates to be able to see them in the map. We also had some problems with updating map's data, it drew a new map each time we change the query. Finally, we create a new method for colouring each province instead of updating the map.

- Tab design: Polymer Web components includes an element called paper-tabs. This element allow tab menu creation but it only allow one tab in each page. We created a new custom-tab element but the search box did not work properly with this new element, so we had to update the mock-up we have created and use a paper toggle button to switch between maps instead of tabs.

6.5 Future work

In the following section I will explain the possible new features or improvements that could be done to the project.

Video analysis This is the main point for future development, now we can not see real time analysis. The idea is to watch the match and while you are watching see the sentiment polarity chart drawing, and also see real time updates in all dashboard widgets.

Adding new events, teams and players Now we only have one event analysed, the idea is to have many events and select each of them to get details. The more events you analysed more data you get and better results are shown. Adding new events means you get a larger list of teams and players analysed, this will provide better rankings to study they popularity and emotions generated.

Add player detail view In this project we have provided a general view of all players available, the idea is to be able to click in each player to get a personalised view: see sentiment evolution, emotions generated in the last matches, etc.

Add team detail view This idea is the same as described above, the idea is to be able to view each team analysed in detail. Sentiment and emotion widgets for each team and with evolution analysis.

Bibliography

[1] Mike Bostock. D3.js - data-driven documents.

[2] Ignacio Corcuera Platas. Development of an emotion analysis system in a university community.

[3] Clinton Gormley and Zachary Tong. *Elasticsearch: The Definitive Guide*.

[4] Jarrod Overson and Jason Strimpel. *Developing Web Components: UI from jQuery to Polymer*. O'Reilly Media.

[5] Jorge Perez and Marcelo Arenas. Querying semantic web data with SPARQL: State of the art and research perspectives.

[6] Constantino Román Gomez. Development of a named entity recognition system based on ensemble machine learning algorithms.

[7] Spotify. Luigi python module.

[8] Evan Tahler. elasticsearch-dump - import and export tools for elasticsearch.

[9] Swizec Teller. *Data Visualization with d3.js*.