UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Departamento de Ingeniería de Sistemas Telemáticos

Grupo de Sistemas Inteligentes

TRABAJO FIN DE GRADO

DESIGN AND IMPLEMENTATION OF A SCRAPING SYSTEM FOR SPORT NEWS

Javier Ochoa Serna

Enero de 2017

TRABAJO FIN DE GRADO

Título:	Diseño e implementación de un sistema de Scraping de noticias deportivas
Título (inglés):	Design and implementation of a Scraping system for Sport News
Autor:	Javier Ochoa Serna
Tutor:	Carlos A. Iglesias Fernández
Departamento:	Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

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

FECHA DE LECTURA:

CALIFICACIÓN:

Resumen

Hoy en día, la forma en la que las personas leemos la prensa escrita ha cambiado. La venta de periódicos en su formato impreso ha caído en torno a un 60% en la última década. Este hecho se ha producido a consecuencia de la invención de Internet y la posibilidad de acceder a miles de contenidos a través de la World Wide Web.

Conociendo esta información, se ha realizado este proyecto de diseño e implementación de un sistema de extracción de noticias en la Web para su posterior análisis. La temática del proyecto podría ser variada, pero se ha decidido centrarlo en noticias deportivas, más concreto en las noticias sobre los equipos de fútbol más populares de España.

El proyecto se divide en varias fases relacionadas con el procesado de la información. Para las tres primeras fases se ha utilizado un flujo de trabajo entre ellas utilizando Luigi. La primera de estas fases es la extracción de datos desde la Web y para llevarlo a cabo se han desarrollado varias arañas web utilizando Scrapy. Posteriormente, dicha información ha sido analizada utilizando Senpy para extraer los sentimientos y emociones existentes en cada artículo. El último paso dentro del flujo es el de almacenar los datos obtenidos en Elasticsearch.

La última fase del proyecto ha sido la de crear una interfaz gráfica con el fin de mostrar los datos que han resultado del análisis y poder compararlos. Para ello se ha utilizado Polymer Web Components junto a la librería de D3.js. Con estas herramientas se han creado distintos widgets que permiten comparar resultados entre los distintos equipos y periódicos.

Este proyecto permitirá al usuario realizar un análisis completo de los periódicos de noticias junto con los equipos de fútbol gracias al conocimiento de las emociones y sentimientos que se generan en la prensa escrita a lo largo del tiempo.

Palabras clave: Sefarad, Scrapy, Senpy, Elasticsearch, Luigi, Polymer, Web Scraping, Deporte, Noticias, Fútbol

Abstract

Nowadays, the way people read print media has changed. The sale of newspapers in their printed format has fallen by around 60% in the last decade. This has occurred as a result of the invention of the internet and the capability of accessing content from thousands of sites through the World Wide Web.

Based on this information we have completed this project of design and implementation of a system of extracting news on the Web for further analysis. The thematic of the project could be diverse, but it has been decided to focus on sports news, in particular news about the most popular football teams in Spain.

The project is divided into several phases related to the processing of information. For the first three phases it has been used a workflow between them using Luigi. First of those phases is the extraction of data from the Web and to implement it several web spiders have been developed using Scrapy. Subsequently, this information has been analyzed using Senpy to extract feelings and emotions in each article. The last step in the workflow is to store obtained data in Elasticsearch.

The last phase of the project has been to create a graphical interface in order to display obtained data in the analysis and being able to compare them. To do this, Polymer Web Components has been used in conjunction with D3.js library. Using these technologies, different widgets have been created in order to compare results between different teams and newspapers.

This project will allow the user to perform a complete analysis of the different newspapers along with the different football teams thanks to the knowledge of emotions and feelings that are generated in the written press over time.

Keywords: Sefarad, Scrapy, Senpy, Elasticsearch, Luigi, Polymer, Web pScraping, Sport, News, Football

Agradecimientos

En primer lugar quiero dar las gracias mi tutor, Carlos, por saber guiarme en cada momento de la realización de este proyecto.

A toda la gente del GSI por brindarme todo su conocimiento en el momento que lo necesitaba.

A mi familia en general, y en concreto a mis padres y mi hermana por apoyarme en todo momento durante estos largos años de carrera.

A mis amigos de siempre por seguir ahí apoyándome después de tantos años.

A todas las personas con las que me he cruzado en este paso por la universidad por aportarme cada uno algo diferente.

Y por último, a esas grandes personas que he conocido en este camino universitario y que me llevo para siempre.

Gracias.

Contents

List of Figures

CHAPTER 1

Introduction

1.1 Context

From the invention of the Internet, the way of communicating written news has changed. Classic newspapers have had to open their own websites in order to satisfy what people prefer.

It is much easier for people to get all the latest news from anywhere and with just one click thru the Internet than buying the physical version. In addition, this option is less expensive and better for the environment.

Thanks to these technological advances, today we can analyze the different news published by each newspaper.

In this project we are focusing on the opinions given in every newspaper website about one topic or event.

The topic we are managing is football because it is known that sporting events evoke strong emotions amongst fans. The idea is to relate these sentiments generated in news with these football events in order to analyze correlations between them.

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

easy access to data.

1.2 Project goals

In this project we will carry out an analysis of the treatment of different sports news media. To this end, the project will focus on obtaining analysis of sports news for further analysis.

This main goal includes two smaller ones:

- Design and implement a sports news extraction system.
- Evaluate the objectivity of these media regarding their opinions on teams, allowing an analysis of both a specific fact and its feeling over time.

1.3 Project tasks

The following tasks have been programmed in the project:

- Review sports news aggregation systems using available APIs or scraping techniques.
- Design the news scraping system, focused on sports news.
- Integrate the designed system into a text analysis system based on a big data platform.
- Design a dashboard based on widgets that allow users interact with them which will show the information collected.
- Evaluate the system.

1.4 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 is an explanation of the context in which this project is developed, the main goals to achieve, the tasks programmed and the structure of the document.

Chapter 2 provides a description of the technologies used in this project.

Chapter 3 explains the complete architecture of the project, the modules that compose it and the workflow that manages it.

Chapter 4 presents experimentation details of a concrete case study with real data.

Chapter 5 exposes the problems faced in this project, the conclusions we have reached and some suggestions for future work.

CHAPTER 2

Enabling Technologies

2.1 Introduction

In this chapter, we are going to give an insight into the techniques used in this project.

In order to carry out this project we have been helped by another project called Sefarad 3.0 and developed in the GSI of ETSIT-UPM. So far, the Sefarad project has been used to analyze tweets, comments and brief opinions. In our case we will add the functionality of extract, analyze, and display news.

To reduce our spectrum we will focus on the sport news about football from the main newspapers in our country. But it could be extrapolated to any type of news topic.

2.2 Sefarad 3.0

First of all we will explain how Sefarad [1] is structured and what technologies it uses. It uses four different technologies all gathered in a pipeline. This four technologies are shown in *Figure 2.1*:

Figure 2.1: Sefarad 3.0

2.2.1 Senpy

Senpy [2] is a technology developed by the GSI of ETSIT-UPM. It is an open source software and uses a data model for analysis of feelings and emotions. It is based on NIF, Marl and Onyx [3] vocabularies.

It aims to facilitate the adoption of the proposed data model to analyze feelings and emotions, so that the services of different providers are interoperable. For this reason, the design has focused on its extensibility and reuse.

There are six different plugins available to use[1]:

- *emoTextAnew*: it extracts the VAD (valence-arousal-dominance) of a sentence by matching words from the ANEW dictionary [4] .
- *emoTextWordnetAffect*: it is based on the hierarchy of WordnetAffect [5] to calculate the emotion of the sentence.
- *vaderSentiment*: it uses the software from vaderSentiment to calculate the sentiment of a sentence.
- *sentiText*: it is a software developed during the TASS 2015 competition[2]:, it has been adapted for English and Spanish.
- *meaningCloud*: it has been adapted from the sentiments analysis API developed by a company called MeaningCloud.

[1] *http://senpy.readthedocs.io/en/latest/demo.html*
[2] *https://gplsi.dlsi.ua.es/sepln15/en/node/36*

- *affect*: it is created in order to analyze sentiments and emotions at the same time. The user has to choose one of the sentiment plugins explained above ("*vaderSentiment*", "*sentiText*" or "*meaningCloud*") and another from the emotions plugins above ("*emoTextAnew*" or "*emoTextWordnetAffect*").

2.2.2 Luigi by Spotify

Luigi[3] is a Python module that helps you build complex pipelines. It allows the user control from the dependencies to the visualization.

Luigi's purpose is to address all pipelines typically associated with long-lasting batch processes. Many tasks can be chained and automated. These tasks can be anything, but they are usually long running things like Hadoop jobs, downloading data to / from databases, running machine learning algorithms or anything else.

Luigi handles a lot of workflow management so you can focus on the tasks themselves and their dependencies.

In our project it is the tool used to create a pipeline that manages the different tasks of which Sefarad is composed.

2.2.3 Elasticsearch

Elasticsearch[4] 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).

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.

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, this file describes where data is going to be stored and the query terms that are going to be indexed.

[3] *https://luigi.readthedocs.io/en/stable/*
[4] *https://en.wikipedia.org/wiki/Elasticsearch*

Second way is much easier and faster, it consists in adding a whole index by using the Elasticsearch's bulk API. In this case you just need all your data stored in a JSON file. You manually add the index where data is going to be stored. [6]

2.2.4 Polymer web components

Polymer[5][6] is a lightweight library that helps you to take full advantage of Web Components. The library is being developed by Google developers and contributors on GitHub. Modern design principles are implemented as a separate project using Google's Material Design design principles.

Web Components[7] are a set of features that are being added by the W3C to HTML and DOM. They aim to introduce component-based software engineering on the World Wide Web.

They allow the creation of widgets or reusable components in web documents and web applications. As well as the encapsulation and interoperability of individual HTML elements.

Web Components consists of 4 main features:

- Custom Elements: APIs to create new elements.
- Shadow DOM: Encapsulated DOM and styling, with composition.
- HTML Imports: Import HTML documents into other documents.
- HTML Templates: Allows documents to contain inert chunks of DOM.

Using Polymer with Web Components, you can create reusable custom elements that interoperate seamlessly with the browser's built-in elements, or break your app up into right-sized components, making your code cleaner and less expensive to maintain.

2.3 Scrapy and other web scraping tools

Once we know the features and functions of Sefarad 3.0, now it is time to explain how we get the content of the news.

[5] *https://www.polymer-project.org/1.0/*
[6] *https://en.wikipedia.org/wiki/Polymer_(library)*
[7] *https://en.wikipedia.org/wiki/Web_Components*

2.3.1 Definition of Web Scraping

Before describing some web scraping tools, it is necessary to explain the meaning of that concept.

Web scraping[8] is a computer software technique of extracting information from websites. This is accomplished by either directly implementing the Hypertext Transfer Protocol or embedding a web browser.

Web scraping is closely related to web indexing, which indexes information on the web using a bot or web crawler and is an universal technique adopted by most search engines.

In contrast, web scraping focuses more on the transformation of unstructured data on the web, typically in HTML format, into structured data that can be stored and analyzed in a central local database or spreadsheet.

Web scraping is also related to web automation, which simulates human browsing using computer software.

Applications of web scraping include online price comparison, contact scraping, weather data monitoring, website change detection, research, web mashup and web data integration.

2.3.2 Some Web Scraping tools

In order to make sure that we are using the best technique for our possibilities and preferences, we have made a study of the different options available.

2.3.2.1 Apache Nutch

It is a highly extensible and scalable open source web crawler software project. It can be used as a robot for crawling or as a search engine.

Nutch[9] is coded entirely in Java, but data is written in language-independent formats. It has a highly modular architecture, allowing developers to create plugins for media-type parsing, data retrieval, querying and clustering.

[8] *https://en.wikipedia.org/wiki/Web_scraping*
[9] *https://en.wikipedia.org/wiki/Apache_Nutch*

2.3.2.2 Newspaper

Newspaper[10] is just a Python3 library for extracting and curating articles.

Some of its main features are the download framework for multi-threaded articles, news URL identification, extraction of Google trends terms and works in more than 10 languages.

2.3.2.3 Event Registry

Event Registry[11] is a tool capable of analyzing news articles and identifying in them different world events.

The system is able to identify groups of articles that describe the same event and represent them as one, even in different languages.

From the data of each event, you can extract the basic information of the event, such as the location, the date, who is involved and what is involved.

The extracted information is stored in a database. It can be queried through a user interface that allows users to search for events using extensive search options.

2.3.2.4 SearchBlox

Searchblox[12] is a business-oriented search engine solution that includes sentiment analysis and text analytics.

It simplifies adding search functionality to portals, intranets, or websites. With a unique combination of ease of customization and simplified search software management, it offers the most cost-effective solution to corporate search challenges.

2.3.2.5 Scrapy

Scrapy [7] is a robust web framework for scraping data from various sources. It is built upon years of experience in extracting massive amounts of data in a robust and efficient manner. With Scrapy you are able to do with a single setting what would take various classes, plugins, and configuration in most other scraping frameworks.

Some of the features of Scrapy are:

[10] *https://github.com/codelucas/newspaper*
[11] *http://eventregistry.org/about*
[12] *http://www.searchblox.com/*

- Its event-based architecture allows us to disconnect latency from throughput by operating smoothly while having thousands of connections open.
- Provides selectors and understands broken HTML code.
- It has a vibrant community that is always helpful.
- Well-organized code that is maintained by the community. It requires a standard way of organization based in modules called spiders and pipelines.
- It allows you to export content to JSON or CSV.

Comparison between different web scraping tools		
Tool	Advantages	Disadvantages
Nutch	Highly scalable and robust	Not to make "scrappers", just to index information
Newspaper	Multi-threaded article download framework and 10+ languages	Limited to certain newspapers
Event Registry	Capable to identify groups of articles that describe the same event and represent them as one, even in different languages	Just a search engine already developed
SearchBlox	Includes all the features we need	Focused for Business and not free
Scrapy	Highly customizable, plenty of documentation and very used	It is not robust against webpages changes

Table 2.1: Comparison between different web scraping tools.

2.3.2.6 Conclusion

We decided to use Scrapy because of the many opportunities it offers. It is also the best option because we want a fine adjustment of the scraper and we need to generate a JSON-

LD[13]. document according to a scheme. In addition, it is being used in the department and it is always good to continue with the line followed in the department.

2.4 Schema.org

In our project we haven't used just Scrapy connected to Sefarad 3.0, we have used some other technologies like the one we are describing in this section, Schema.org[14].

Schema.org[15] is an initiative developed by the world's largest search engines (Bing, Google, Yahoo and Yandex). It has the mission to create, maintain and promote schemas for structured data on the Internet.

Its vocabulary is used along with different encodings, including JSON-LD, RDFa [8] and Microdata in order to mark up website content with metadata about itself. Such markup can be recognized by search engine spiders and other parsers.

There are some other options for structuring data [9] One of these alternatives is NewsML, a XML standard designed by the IPTC to provide a media-independent, structural framework for multimedia news. We didn't use this technology because it is more oriented to content providers.

SportsML is another XML standard created by the IPTC but in this case it is oriented for the interchange of sports data. It is open and highly flexible. We haven't used this option neither because it is enough with the NewsArticle vocabulary of Schema.org.

In addition, we would like to point out that all the technologies commented on in this section are Linked Data technologies [10] . They use this method that is based on publishing structured data so that they can be interlinked and become more useful through semantic queries.

2.5 Conclusions

We are familiar with all of the technologies involved in the project. Therefore, we can represent the final structure of our project. It is composed of Scrapy connected to Sefarad and it is shown in *Figure 2.2*.

[13] *http://json-ld.org*
[14] *https://schema.org*
[15] *https://en.wikipedia.org/wiki/Schema.org*

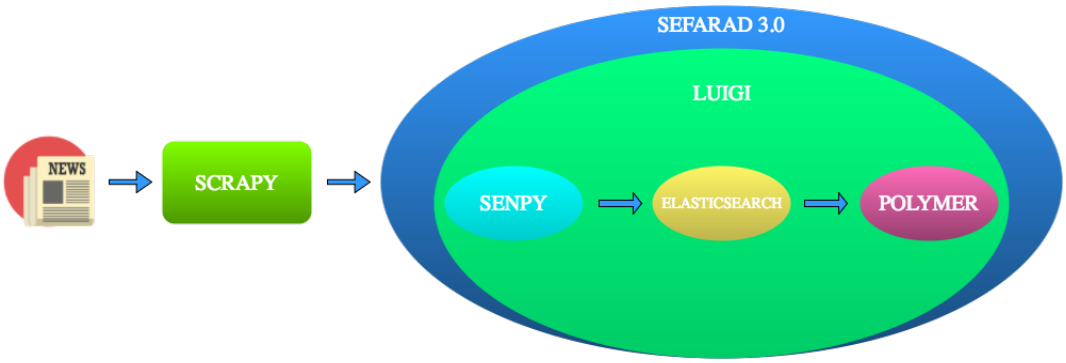

Figure 2.2: Sefarad 3.0 with Scrapy

CHAPTER 3

Architecture

3.1 Introduction

In this chapter we are going to explain the architecture of the project and how it is composed. We will first give an overview of all the modules together and later we will explain separately each module in more detail.

The explanation in this chapter will not be oriented just in showing the architecture of our project. We will also show what we have created and we will give some examples of it.

3.2 Architecture

In this section we present the architecture of the project in general, defining the different modules that participate in it. The project is composed of the following modules shown in *Figure 3.1*:

- **Orchestrator**: We get help from Luigi with the goal of building a pipeline to connect different modules.
- **News extraction system**: This is the main part of the project, where we obtain

data from different news websites.

- **News analysis system**: We use Senpy to analyze sentiments and emotions in the extracted news.
- **Indexing system**: In this module we index the information obtained in previous modules using Elasticsearch.
- **News visualization system**: This stage is responsible for processing data and showing it in different views.

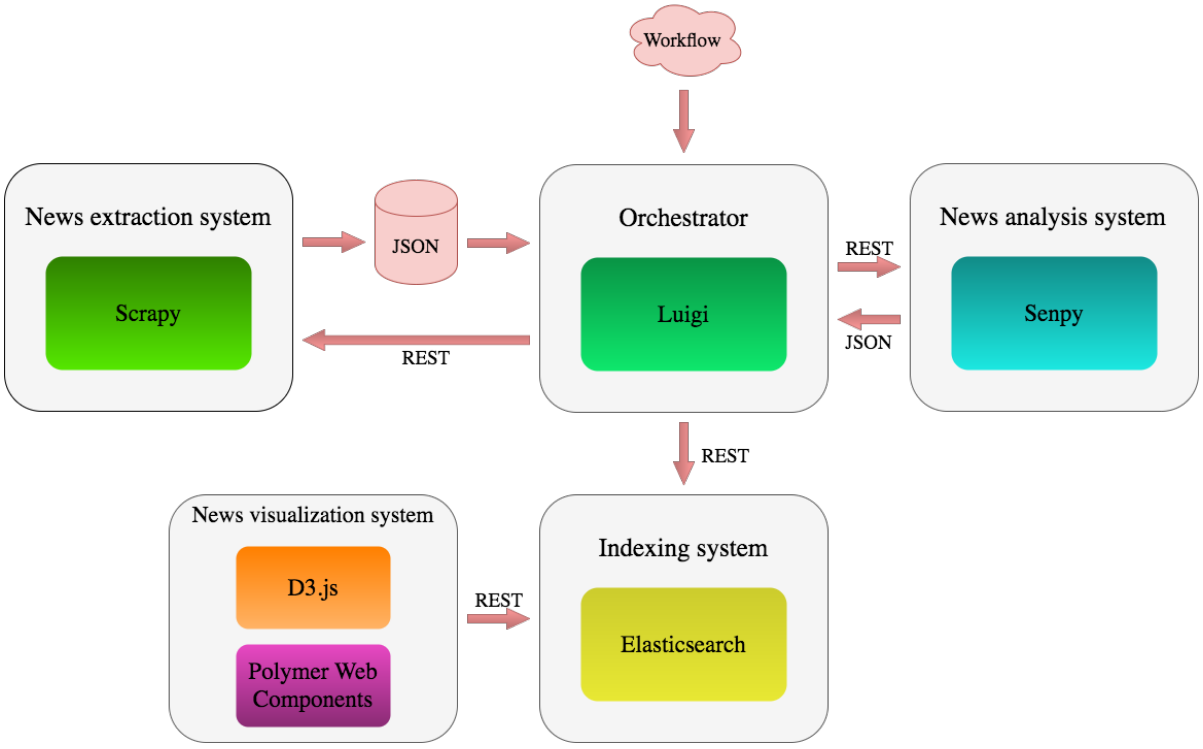

Figure 3.1: Architecture

3.3 Orchestrator

Our orchestrator is Luigi, a Python module developed by Spotify that helps us to build a pipeline through the indexing system and the news analysis system of our project.

Luigi needs a script describing the pipeline to follow. The pipeline can be named workflow as well. This workflow is described next.

3.3.1 Workflow

In this subsection we are going to explain the tasks that take part inside a pipeline called *pipelinenews.py* that we have created using our orchestrator Luigi.

This pipeline is composed of these four different tasks: ***ScrapyTask***, ***FetchDataTask***, ***SenpyTask*** and ***Elasticsearch***. They correspond to the stages shown in next figure:

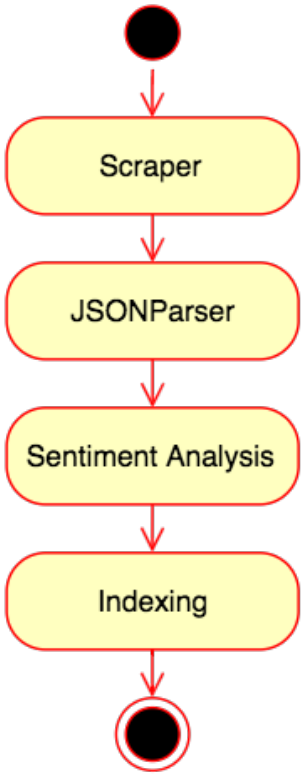

Figure 3.2: Activity workflow

There has been a small inconvenience when using Luigi. We have had to add the creation date of the JSON document to its name in ScrapyTask. The reason is because Luigi is idempotent and it tries to rerun the same task if the parameters introduced are the same. Luigi does it because when running big pipelines it is more effective to rerun just the failed tasks and not all of them. In our case it is a disadvantage but we have solved it this way.

When running this workflow, we use the following commands to execute it:

Listing 3.1: Example of commands to execute the workflow

```
$ python pipelinenews.py Elasticsearch --filename1 marca --filename2 as --filename3 md
    --filename4 sport --index footballnews --doc-type news --local-scheduler
```

3.3.1.1 Scraper Task

This is the first task that takes place in the workflow and it is in charge of executing the necessary Scrapy commands in order to obtain the JSON documents with all the extracted information.

As we have explained before, thanks to the addition of the date of creation in JSON's name, this task can work properly. This task is related with the News Extraction System and it will be explained in detail in *Section 3.4*.

3.3.1.2 JSONParser

JSONParser or FetchDataTask is the next task after extracting data and it is in charge of reading the JSON files that we indicate as parameter. In our case, we introduce as many file parameters as newspapers we are analyzing.

3.3.1.3 Sentiment Analysis Task

This task corresponds to the News Analysis System. It loads data fetched with previous task and sends it to Senpy tool in order to analyze sentiments and emotions.

When both analysis are done (sentiments and emotions), it gathers the analyzed data and stores it in a JSON-LD document with the name analyzed-filename.jsonld. This document is ready to be used in the next task.

3.3.1.4 Indexing Task

This is the last task and it corresponds to the Indexing System. It is in charge of loading the JSON-LD document obtained from previous task in the indicated Elasticsearch index. We also indicate the Elasticsearch doc-type as a parameter.

3.4 News extraction system

This is the most important stage of the project. It is responsible for the extraction of all the information that we are going to use in the next stages.

As we have stated in *chapter 2*, we have chosen Scrapy as the ideal technology for this task. Scrapy uses Python files called "spiders" to extract any information that is wanted from any website. In our case, we have used it to extract news articles from the most relevant sport newspapers from our country.

We already know that when using Scrapy, you have to create spiders in order to go through a website and extract the specific information that you want. In our case, we have created four different spiders because we are analyzing four different sport newspapers (Marca, As, Mundo Deportivo and Sport).

Although we have different teams and news, it is only necessary to create one spider per newspaper. The reason is because the structure of articles is the same in each website.

For each of these spiders, we had to analyze the CSS of each website, since the extraction is done from CSS selectors.

The process that we have had to follow when creating every spider is described next:

- First of all, we have had to obtain the URL from the website we want to explore.
- The next step is to get the specific URL of each team inside every newspaper website.
- Once we know every team URL, we have had to explore the CSS code in order to know the CSS selector to access every article in particular.
- Finally, inside every article, we have had to explore CSS code again in order to obtain the CSS selectors of every part of article that we want to extract (headline, body, images, videos, comments, etc.).

In the next figure and listing (*Figure 3.2* and *Listing 3.1*) we are giving an example of the same article comparing two points of view. *Figure 3.2* represents the article when it is shown in the Web and *Listing 3.1* represents the Python code from the spider in charge of extracting information from this article's newspaper.

Figure 3.3: Example of a news article shown in Web browser.

Listing 3.2: Example of the "Spider" in charge of extracting information from the news article in previous figure.

```
# -*- coding: utf-8 -*-
import scrapy
import re
import datetime
import random

class MarcaSpider(scrapy.Spider):
    name = "marca"
    start_urls = [
        'http://www.marca.com/futbol/atletico.html',
        'http://www.marca.com/futbol/real-madrid.html',
        'http://www.marca.com/futbol/barcelona.html'
    ]
```

```
team = ""
teamsURL = ['atletico','real-madrid', 'barcelona']
teamsParse = ['atletico','madrid', 'barca']

# ------------------------------------------------------------
# ORDENES
# $ scrapy crawl marca -o marca.json
# ------------------------------------------------------------

def parse(self, response):
    # follow links to each news item
    for href in response.css('h3.mod-title a::attr(href)').extract():
        yield scrapy.Request(href, callback=self.parse_news)

def parse_news(self, response):
    headline = response.css('h1.js-headline.izquierda::text')[0].extract()
    articleBody = ''.join(response.css('div.row.content.cols-30-70 span.capital-
        letter::text ,div.row.content.cols-30-70 p::text ,div.row.content.cols-30-70
         p strong::text').extract())
    author = response.css('ul.author strong::text')[0].extract()
    articleSection = response.css('span.section-type::text')[0].extract()
    commentCount = response.css('li.comments-tool strong::text')[0].extract()
    datePublishedString = response.css('div.row.content.cols-30-70 time::attr(
        datetime)')[0].extract()
    datePublished = datetime.datetime.strptime(datePublishedString, '%Y-%m-%d %H:%M
        :%S').strftime('%Y-%m-%dT%H:%M:%S+01:00')
    images = response.css('div.row.content.cols-30-70 figure img::attr(src)').
        extract()
    videos = response.css('div.row.content.cols-30-70 meta[itemprop="contentURL"]::
        attr(content)').extract()
    keywords = response.css('ul.item-tags a::text').extract()
    comments = ' - '.join(response.css('div.comentario strong.numero_comentario a::
        text, div.comentario p.nombre span.nombre_usuario::text, div.comentario p.
        nombre span.nick::text, div.comentario span.date::text, div.comentario span+
        p::text, div.comentario p.nombre img::attr(src)').extract())
    dateCreated = datetime.datetime.now().isoformat()
    url = response.url
    _id = str(random.randint(0,10000)) + dateCreated
    self.team = ""
    for indexURL, elemURL in enumerate(self.teamsURL):
        if bool(re.search(elemURL, url)):
            self.team = self.teamsParse[indexURL]

    if self.team == "":
        return

    item = {'@context':'http://schema.org',
        'headline':headline,
        'articleBody':articleBody,
        'author':author,
        'articleSection':articleSection,
        'commentCount':commentCount,
```

```
            'datePublished':datePublished,
            'images':images,
            'videos':videos,
            'keywords':keywords,
            'comments':comments,
            'dateCreated':dateCreated,
            'newspaper': "marca",
            'url': url,
            'team': self.team,
            'id':_id
            }
        yield item
        return
```

When we know all the CSS code required and we have created the spiders as we desire, it is time to execute the following commands in order to obtain a JSON document with all the information.

Listing 3.3: Scrapy commands to extract information from the news article in previous figure.

```
$ scrapy crawl marca -o marca.json
```

Finally, as we have said before, we obtain four JSON documents (one document per spider) with all the information extracted from the different media. An example of one of those JSON documents is shown in *Figure 3.3*.

▼ 0 {15}

@context : http://schema.org

newspaper : marca

articleSection : Atlético de Madrid

headline : Simeone: \"No, no voy a dejar el Atlético\"

articleBody : Simeone insistió de nuevo en que continuará en el equipo. Tras el encuentro, volvieron a cuestionarle al respecto y el entrenador fue muy claro en su respuesta. \"No, no voy a dejar el Atlético\", aseguró. El Cholo también explicó por qué está colocando ahora a Juanfran en el extremo derecho. \"Ha jugado toda la vida ahí y ofensivamente siempre nos ha dado las mejores alternativas, tanto en pase como en profundidad. Y con el presente muy bueno de Vrsaljko se nos abre una posibilidad en esa banda. Siempre busco soluciones, quieto no me voy a quedar. Eso no quiere decir que ahora vayamos a jugar siempre así\", aseguró el técnico argentino.Simeone, entrenador del AtléticoEl ambiente y el encuentro. \"La fiesta no la vi, estaba en el vestuario, oí los ruidos y creo que fue bonita. Además, el partido entusiasmó a la gente porque vio goles y pudo disfrutar de un buen encuentro\".El Al-Ittihad. \"Había visto imágenes del rival, entendía que tiene buena posesión e individualidades en el centro del campo. No me sorprendió, sabíamos al tipo de equipo al que nos íbamos a enfrentar\". La Liga árabe. \"No he visto muchos partidos de la Liga árabe para dar una opinión, pero sí fue fantástico el ambiente del partido y el rival. Nos hizo bien para recuperar después del parón y nos servirá para los encuentros de Copa y Liga que vienen\". Los jugadores destacados del rival. \"Me gustó el 11, es el vértice, el 10 tiene velocidad, el 9 es profundo y fuerte. No digo los nombres, porque los diría muy mal\".

author : ISAAC SUÁREZ

datePublished : 2016-12-30 21:48:51

▼ images [1]

0 : http://e00-marca.uecdn.es/assets/multimedia/imagenes/2016/12/30/14831308404217.jpg

▶ videos [0]

▼ keywords [2]

0 : Atlético de Madrid

1 : fútbol

url : http://www.marca.com/futbol/atletico/2016/12/30/5866c761e5fdeabc398b4683.html

id : 38702016-12-31T19:30:54.078850

dateCreated : 2016-12-31T19:30:54.078850

commentCount : 20

comments : //e03-marca.uecdn.es/iconos/v1.x/v1.0/community/avatars/generic-50.png - titicadi - //e02-marca.uecdn.es/social/ugc/avatars/003/036/medium_3036233.png - lestrellaenelpecho - //e01-

Figure 3.4: Example of the JSON extracted from the news article in previous figure.

To conclude, it is important to mention that it is necessary to name every JSON document that is created with the name of its associated newspaper, but it is also important to add the actual date of the moment it is created. We have explained this issue in *Section 3.3.1.*

3.5 News analysis system

In this project we have used one analytic service. It is called Senpy and we have used it to analyze sentiments and emotions in the extracted news. *(See Chapter 2)*

With Senpy you can use several analyzers, but in our case we have used ***meaningCloud*** to analyze feelings and ***EmoTextANEW*** to analyze emotions. In order to execute both analyzers at the same time, we have used the ***affect*** plugin that allows to gather emotions analysis and sentiments analysis in one execution. Other options have been ruled out as they were not appropriate or did not admit the Spanish language.[1]

[1] *http://senpy.readthedocs.io/en/latest/demo.html*

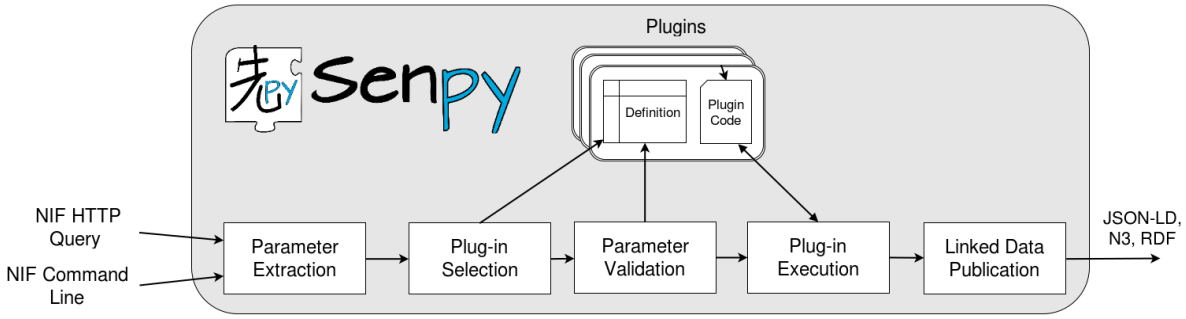

Figure 3.5: Senpy architecture

3.5.1 EmoTextANEW

We have used this plugin to analyze some emotions as ***anger, disgust, fear, joy, neutral emotion or sadness***.

This plugin uses the ANEW lexicon dictionary to calculate de VAD (valence-arousal-dominance) of a news item and determinate which emotion is closer to this value.

It looks for the words in the article that appear in the ANEW dictionary and calculates the average VAD score for this news article. Once this score is calculated, it is going to seek the emotion that is closest to this value.

The way of using this plugin is by accessing an specific URL with the required parameters. We have introduced three parameters when running this URL. The first parameter is called "algo" and here is where we introduce the type of analysis that we are running, *algo=EmoTextANEW* in this case. The second parameter is called "language" and as its name says, it refers to the language in what the content to analyze is written. In the last parameter we introduce the text that we want to analyze, it is called "input".

An example of the URL used and its output are given next.

Listing 3.4: Example of Senpy EmoTextANEW access URL

```
http://senpy.cluster.gsi.dit.upm.es/api/?algo=EmoTextANEW&language=es&input=alegre
```

Listing 3.5: Example of Senpy EmoTextANEW output

```
"entries": [
{
  "@id": "Entry_1483214337.57",
  "emotions": [
```

```
  {
    "@id": "Emotions0",
    "onyx:hasEmotion": [
    {
      "@id": "Emotion0",
      "http://www.gsi.dit.upm.es/ontologies/onyx/vocabularies/anew/ns#arousal": 7.66,
      "http://www.gsi.dit.upm.es/ontologies/onyx/vocabularies/anew/ns#dominance": 6.37,
      "http://www.gsi.dit.upm.es/ontologies/onyx/vocabularies/anew/ns#valence": 8.41,
      "onyx:hasEmotionCategory": "http://gsi.dit.upm.es/ontologies/wnaffect/ns#joy"
    }
    ]
  }
  ],
  "language": "es",
  "nif:isString": "alegre"
}
]
```


3.5.2 meaningCloud

We have used this plugin to analyze sentiments. It classifies feelings as positive, negative, or neutral. It extracts sentiment at a document and aspect-based level. To do this, the local polarity of the different sentences in the text is identified and the relationship between them evaluated, resulting in a global polarity value for the whole text.[2]

This plugin has the following differentiators:

- Extracts aspect-based sentiment.
- Discriminates opinions and facts.
- Detects polarity disagreement and irony.

This plugin is used the same way as EmoTextANEW plugin. In this case, the URL is different but we also use those two parameters that we have explained in previous section. In addition, it is needed to introduce a new parameter when using this plugin because MeaningCloud API requires to use an "apiKey" for accessing its tools. This parameter is called *"meaningCloud-key"*.

[2] *https://www.meaningcloud.com/developer/sentiment-analysis*

Next, we give an example of the URL used and output in this case.

Listing 3.6: Example of Senpy meaningCloud access URL

```
http://senpy.cluster.gsi.dit.upm.es/api/?algo=meaningCloud&meaningCloud-key=
    XXXXXXXXXXXXX&language=es&input=alegre
```

Listing 3.7: Example of Senpy meaningCloud output

```
{
  "@context": "http://reed.gsi.dit.upm.es:4000/api/contexts/Results.jsonld",
  "@id": "Results_1484767496.34",
  "analysis": [
    {
      "@id": "meaningCloud_1.0",
      "@type": "marl:SentimentAnalysis",
      "author": "GSI UPM",
      "description": "Sentiment analysis with meaningCloud service",
      "extra_params": {
        "apiKey": {
          "aliases": [
            "meaningCloud-key",
            "apiKey"
          ],
          "required": true
        },
        "language": {
          "aliases": [
            "language",
            "l"
          ],
          "default": "en",
          "options": [
              "en",
              "es"
          ],
          "required": true
        },
        "model": {
          "aliases": [
            "model"
          ],
          "default": "general",
          "options": [
            "general"
          ],
          "required": true
        }
      },
```

```
      "is_activated": true,
      "maxPolarityValue": 1.0,
      "minPolarityValue": 0.0,
      "module": "meaningCloud",
      "name": "meaningCloud",
      "requirements": {},
      "version": "1.0"
    }
  ],
  "entries": [
    {
      "@id": "Entry0",
      "nif_isString": "alegre",
      "sentiments": [
        {
          "@id": "Opinion0",
          "marl:hasPolarity": "marl:Positive",
          "prov:wasGeneratedBy": "meaningCloud_1.0"
        }
      ]
    }
  ]
}
```


3.5.3 affect

This is the plugin that we have really used, both two plugins described in previous subsections are gathered when using this other plugin.

We have used this plugin to analyze sentiments and emotions at the same time. It allows us to take advantage of the features of both EmoTextANEW and meaningCloud plugins.

This plugin is used in a similar way as the previous ones. In this case, it is needed to specify which plugins are wanted to be used. Because of this reason, *"sentiplug"* and *"emoplug"* parameters are added to the other parameters explained before.

Next, we give an example of the URL used.

Listing 3.8: Example of Senpy affect access URL

```
http://senpy.cluster.gsi.dit.upm.es/api/?algo=affect&sentiplug=meaningCloud&meaningCloud
    -key=XXXXXXXXXXXXX&emoplug=EmoTextANEW&language=es&input=alegre
```

An output example is not shown in this case because it is similar as the ones we have shown in EmoTextANEW and meaningCloud plugins.

3.6 Indexing system

This system is in charge of indexing all the news information extracted and analyzed in the previous tasks. It is composed of Elasticsearch, the searching server that connects the analyzed information with the news visualization system.

There are two different ways for adding data to an Elasticsearch index. In our case, we have chosen the easier and faster way; it consists in adding a whole index by using the Elasticsearch's bulk API. We just need to store all the data in a JSON file and manually add the index where data is going to be stored.

Although at the beginning we were thinking about making one index per newspaper and team, we later realized that it is faster and more effective to make just one index with all information gathered. We have called this index *"footballnews"*.

To conclude, Elasticsarch allows you to consult indices status by accessing this URL: *http://localhost:9200/_cat/indices* . We are showing an example of results next.

```
yellow open update_log   LKovCYa0Qu67Y7X4sgQT8A 5 1  34 0 140kb 140kb
yellow open footballnews uEVKS6ItTN6TqbI5YA_yfg 5 1 400 0 1.9mb 1.9mb
```

Figure 3.6: Elasticsearch index displayed in browser

3.7 News visualization system

Although this is the last part of the project, it is very important. It depends on if the target user understands what it is wanted to show and transmit. It doesn't matter how good the previous stages are if you are not able to show clearly what you have developed.

This visualization server is based on Polymer Web Components and is developed from Sefarad 3.0 [1]. We have also used D3.js [11] library to design some new dynamic Web Components.

3.7.1 Structure

When creating any new visual project, first step always should be to create a sketch or mock-up. Is the first stage, but also the most important stage because it is necessary to know what structure is wanted the project will have at the end. There will always be time

to make changes if you change your mind.

In our case, we were creating a website with the need of showing correctly how the emotions and sentiments extracted change between different newspapers and football teams.

We reached the conclusion that it was needed to use some different charts that allow the user to compare between those different medias and teams. It was also a good idea to show some other interesting information like headlines or number of news.

The second conclusion that we reached is that we needed to structure the website with a sidebar to access different views and different tabs inside every view. The reason was because we had different information interconnected that needed to be compared.

Knowing these conclusions, we have created the following mock-ups using draw.io[3]. They are shown next in *Figure 3.6* and *Figure 3.7*:

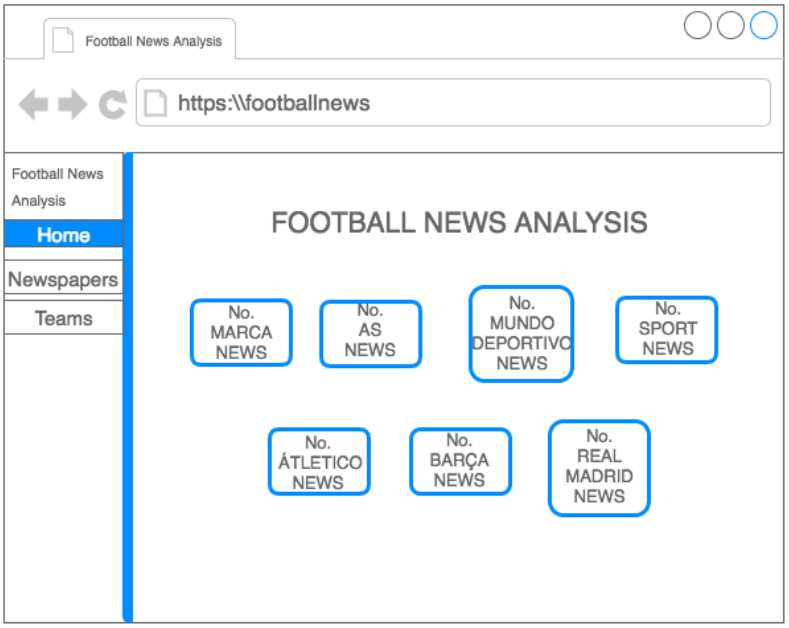

Figure 3.7: Home tab mock-up

[3] *https://www.draw.io/*

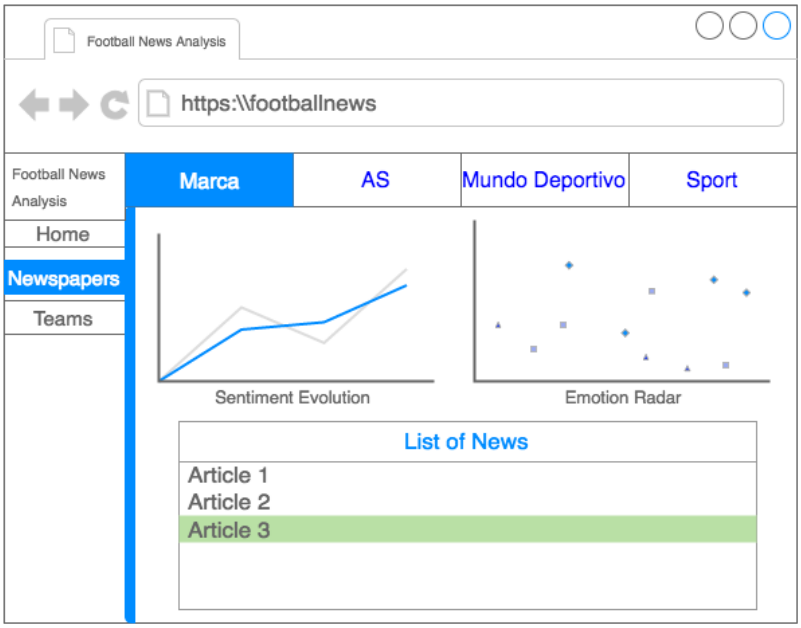

Figure 3.8: Newspapers tab mock-up

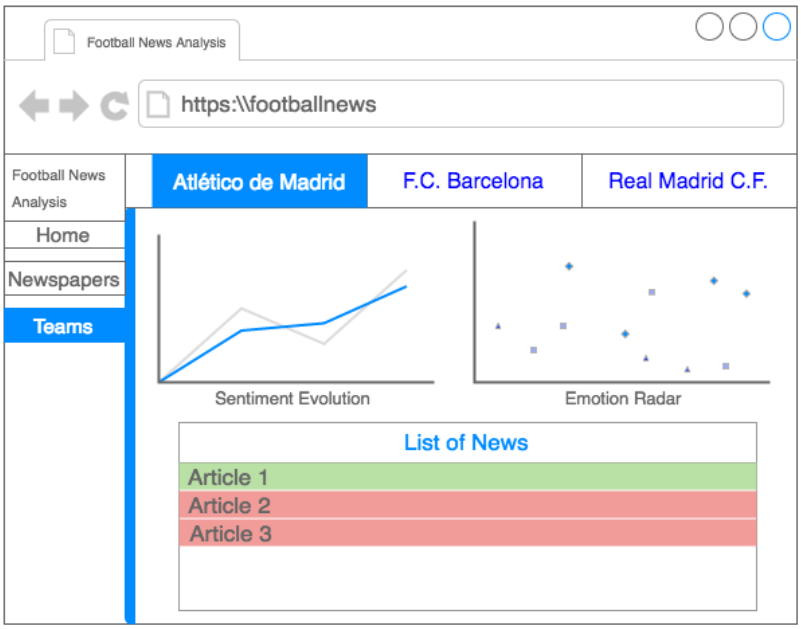

Figure 3.9: Teams tab mock-up

3.7.2 Elements and widgets

Once we knew how our website was going to be structured, it was time to create it and find some different widgets that satisfied our requirements. Some of them were from the Polymer Element Catalog[4] and other from the D3.js [11] library. This second one needed to be developed in order to be adapted to the structure and requirements of Polymer.

3.7.2.1 Polymer elements

We have used some different Polymer elements, but two of them are more important than the others because they are responsible of building and organizing the main structure of the website. They are described next:

- **Paper-drawer-panel**

  This element is in charge of building the sidebar that contains the main menu of the website. In our case, we have divided this sidebar and our website in three views explained next:

  - **Home:** It is the view in charge of giving a good welcome when accessing the website and gives you a summary of the data extracted.
  - **Newspapers:** This view is in charge of giving a comparison between the three different teams analyzed when selecting the desired newspaper.
  - **Teams:** This view is in charge of giving a comparative between the four different newspapers analyzed when selecting the desired team.

  In addition, we want to mention that this element is dynamic and it disappears on the left when the device is small, more room on the screen is needed for the main information

- **Paper-tabs**

  This second element is in charge of arranging the secondary menu that appears in newspapers and teams view. It shows one option per newspaper and per team depending in what view it is being displayed.

[4] *https://elements.polymer-project.org*

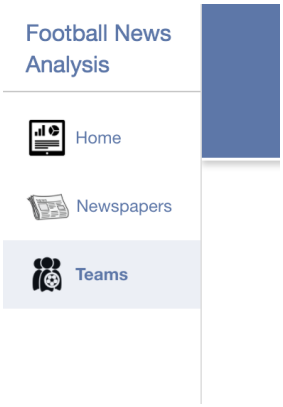

Figure 3.10: Paper-drawer-panel widget

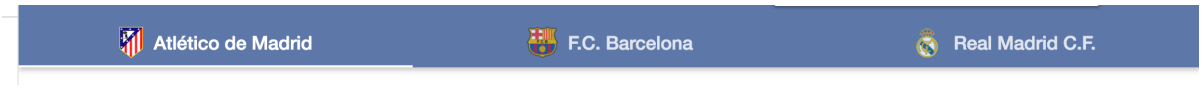

Figure 3.11: Paper-tabs widget

3.7.2.2 D3.js widgets

We have also used and adapted some D3.js widgets because Polymer Element Catalog is not enough sometimes, when representing charts for example.

- **Trend-chart-multiple**

  This widget is used to show how the sentiment results change among the days in the specific case chosen by selecting the different tabs.

  The way of showing information in this widget is by displaying a lines chart with as many lines as newspapers or teams are being compared. Time evolution is represented on x-axis and sentiment polarity on y-axis.

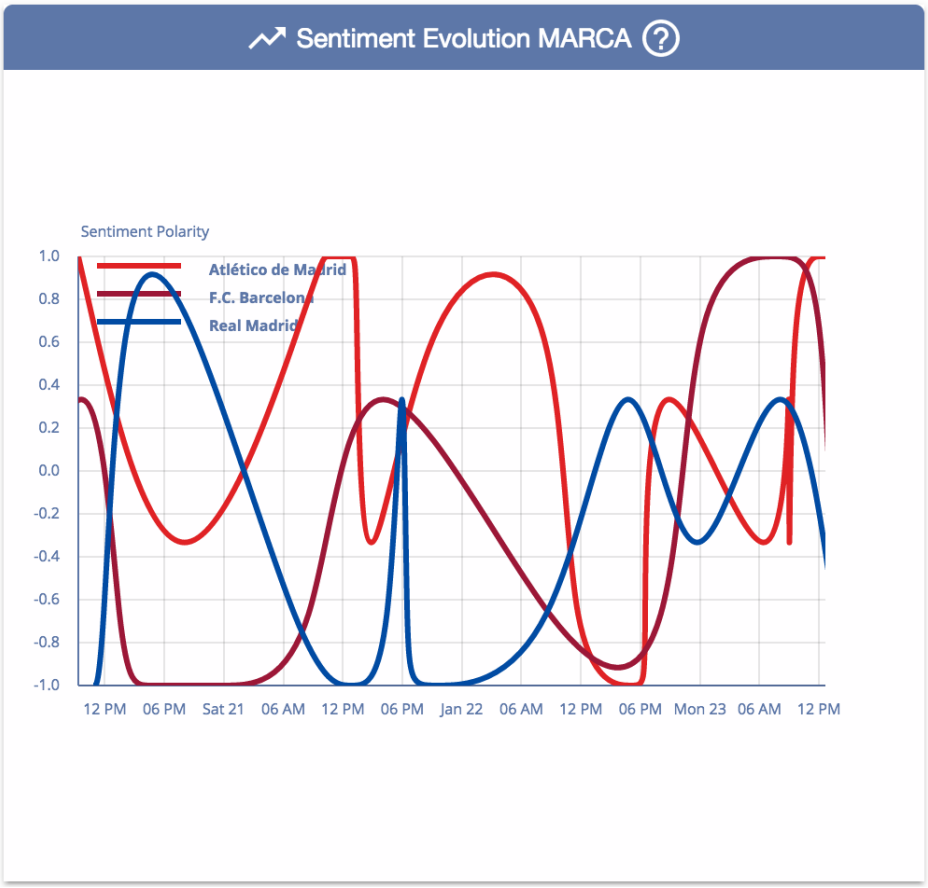

Figure 3.12: Trend-chart-multiple widget

- **Spider-chart-multiple**

  This other widget is used to show how the different news extracted are distributed if we put our eye in the most significant emotion analyzed on each of them. The information shown changes depending which tabs are selected.

In this case, the information is shown using a radar chart with six axis, one per emotion analyzed. Each vertex of the hexagon changes with the number of articles with the same predominant emotion. The results are displayed with a different color for every newspaper or team compared.

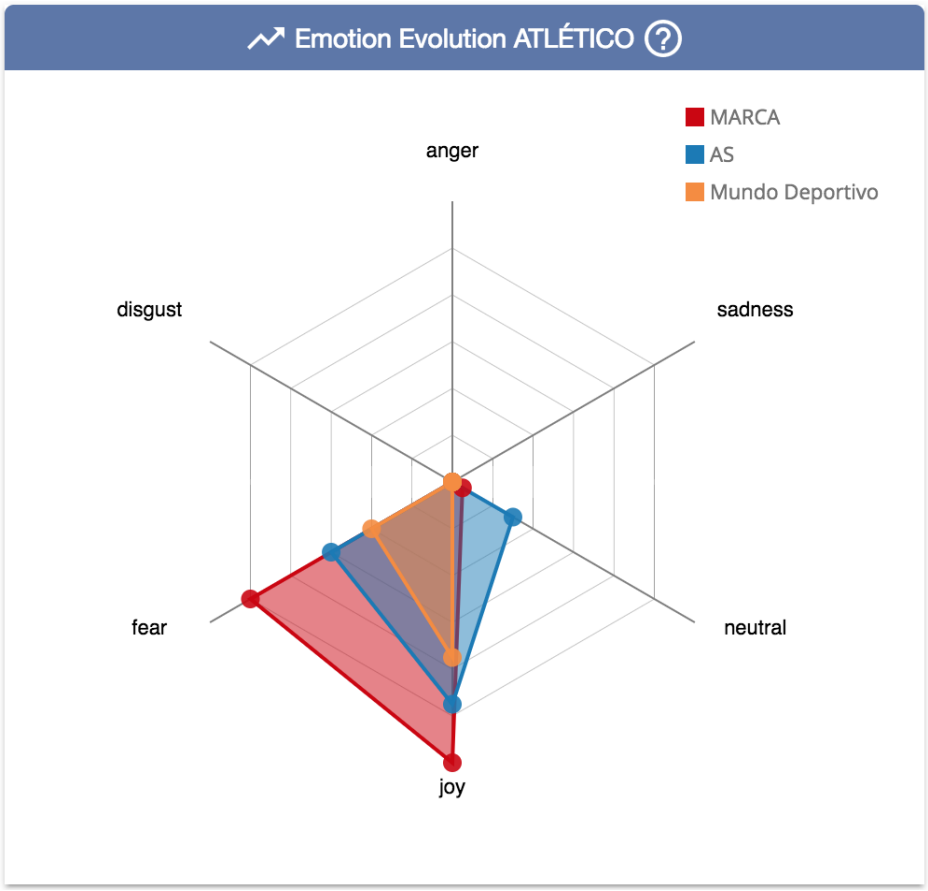

Figure 3.13: Spider-chart-multiple widget

- **News-chart**

  We are using news-chart widget in order to display some news by representing their headline, their author and their newspaper or team. The news shown change depending which tabs are selected.

  The most relevant information that this widget provides is displayed in the background of each article's container. It changes the color depending if the news represented has positive, negative or neutral sentiment. The color can be green, red or grey respectively.

Figure 3.14: News-chart widget

CHAPTER 4

Case study

4.1 Introduction

This chapter is a description of the process followed in order to extract news information, analyze it, store it and show it in a specific case.

The main actor in this case is the user who consults our web site with the objective to find some information about teams or newspapers in a concrete moment.

4.2 Extracting data

As we have said in previous chapters, we have been extracting information from football news in some Spanish newspapers. The first step was to decide which data to extract. We had had to decide which teams, which newspapers and which period of time we were interested in.

In our case, what we have chosen to extract data is shown next:

- Teams: the best three football teams in our country (Atlético de Madrid, Real Madrid

and F.C. Barcelona).

- Newspapers: the four most relevant newspapers in our country when talking about football.

- Period of time: the days that involve round 19 of La Liga Santander 2016-2017, that is the weekend from Friday 20 of January to Monday 23 of January, both included.

Once we knew what data we wanted, the second step was to configure the Scrapy cron to be executed automatically during that period of time obtaing the required data. In this lapse of time we obtained 400 news.

4.3 Analyzing data

After obtaing all the data, the next step was to analyze it using Senpy. As we had set up two different analyzers, we have two different results of analysis. First one is emotion analysis and we have detected 196 negative-fear, 187 joy, 0 disgust, 2 sadness, 1 anger and 14 neutral emotion news. Second analyzer is sentiment analysis and we have obtained 0 neutral, 195 negative and 205 positive news.

4.4 Indexing data

Once the data was analyzed, it was time to store and index it using Elasticsearch. Although at the beginning we were thinking of creating one index per newspaper and team, we finally decided to create just one for all the data. This final solution is more compact and easy to use.

Our index name is *"footballnews"* and it is structured with two subindex in it. There are one subindex per analysis and their names are *"sentiments"* and *"emotions"*.

4.5 Displaying data

The data is displayed using a dashboard based on a sidebar, tabs and widgets. The sidebar is placed on the left and it has three different options (*"Home" "Newspapers" and "Teams"*). We will talk about widgets and tabs when they appear in the different views.

When a user opens the website, first option (*"Home"*) in the sidebar is selected and a welcome message is displayed along with a summary of the analyzed data. There aren't tabs inside this first view and there is an example shown in *Figure 4.1*.

Figure 4.1: Home option

The second and the third options in the sidebar have a similar structure. Both of them display a view with three main widgets, two widgets on the top and one on the bottom. First widget on top is placed on the left side and shows a line chart with the evolution of the sentiment analysis along the time. Second widget on top is placed on the right side and shows a summary of the emotion obtained using a radar chart. Last widget displays a list of different articles represented with different colors depending on how the sentiment is in each of them.

Although they have a similar structure, second and third option in the sidebar have differences such as number of tabs displayed or information compared.

"Newspapers" is second option and it has four different tabs (*"MARCA"*, *"AS"*, *"MUNDO DEPORTIVO"* and *"SPORT"*), one tab per analyzed newspaper. When clicking on different tabs, the information shown change. For example, if you click on *"MARCA"* tab, the widgets mentioned before change their values with the data obtained only in this Marca newspaper. In addition, if we compare with next option in the sidebar, the information compared in the widgets placed on top are the different teams that have been analyzed. An example of this view is shown in *Figure 4.2*.

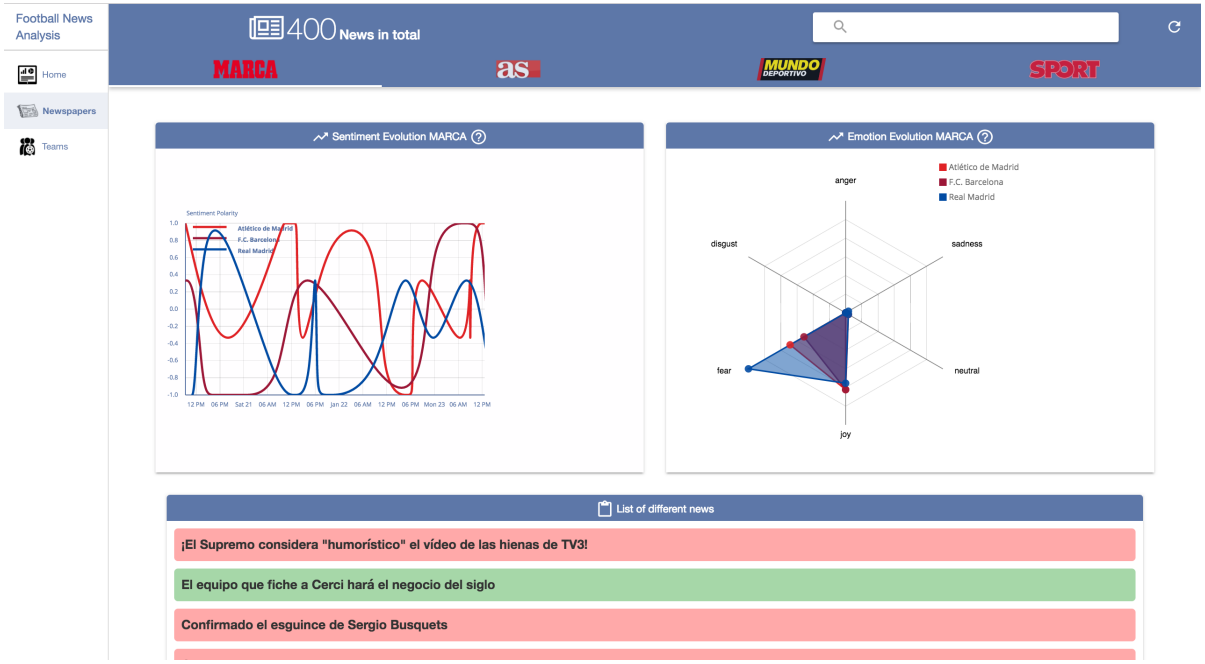

Figure 4.2: Newspapers option

The last option in the sidebar is *"Teams"* and it has one tab per analyzed team (*"Atlético de Madrid"*, *"F.C. Barcelona"* and *"Real Madrid C.F."*). As it occurs in *"Newspapers"* option, the information to be displayed is different if we click in one tab or another. For example, if you click on *"Atlético de Madrid"* tab, the widgets mentioned before change their values with the data obtained only about the Atletico de Madrid team. In this case, the information compared in the widgets on top are the different newspapers where we have obtained the information. An example of this view can be observed in *Figure 4.3*.

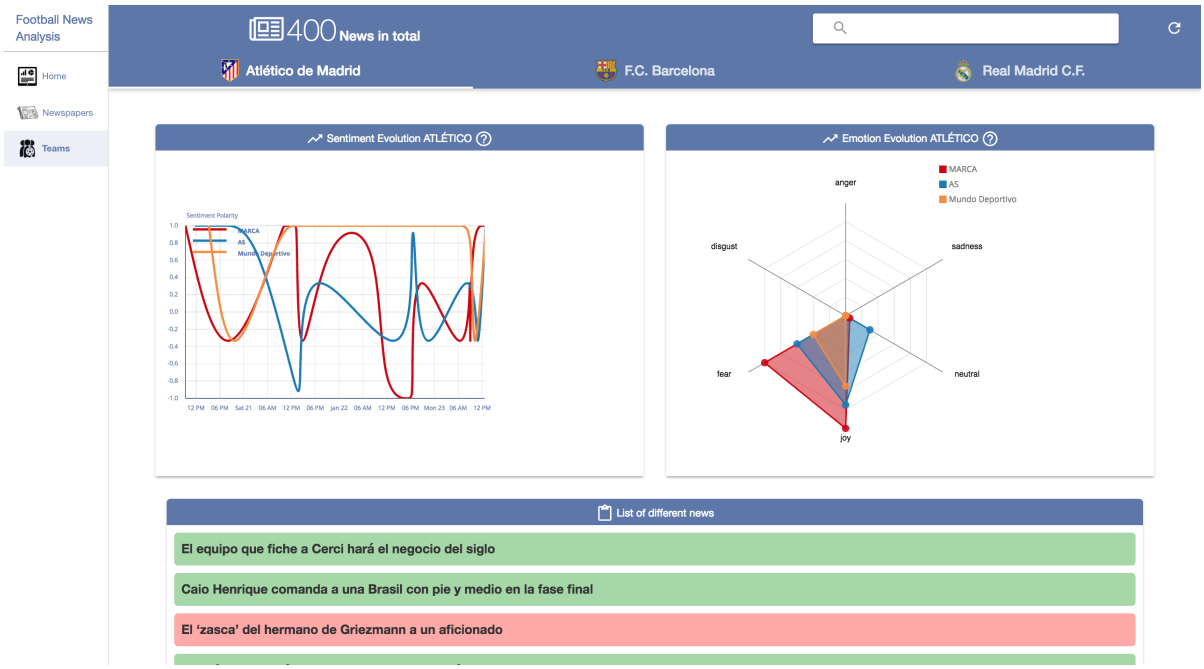

Figure 4.3: Teams option

4.6 Conclusions

In this chapter we have presented the steps we have followed for extracting data and analyzing it. We have explained the widgets displayed and the different views developed, and also how the website could be used.

These results together with the information shown have allowed us to reach some conclusions shown below:

- We can conclude that Mundo Deportivo offers really good opinions of Atlético de Madrid and F.C. Barcelona. Real Madrid C.F. articles aren't that positive.
- It can be seen a positive zone in Real Madrid C.F news right after its match on Saturday 21, 16:15. The reason could be the match result, they won (2-1) versus Malaga C.F.
- It can be seen a negative zone in Atlético de Madrid news right after its match on Sunday 22, 16:15. The reason could be the match result, they tied (2-2) versus Athletic Club de Bilbao.
- It can be seen a positive zone in F.C. Barcelona news right after its match on Sunday 22, 20:45. The reason could be the match result, they won (0-4) versus S.D. Eibar.

- It can be concluded that AS newspaper offers mostly positive articles of all three analyzed teams.

CHAPTER 5

Conclusions and future work

5.1 Introduction

In this chapter we are going to explain the conclusions extracted from this project, achievements, problems faced and options to develop in future work.

5.2 Conclusions

In this project we have created a scraping system for sentiment and emotion analysis of football news. In addition, we have created a dashboard based on widgets in order to make it interactive and accessible, and to show clearly the extracted data.

The project has the characteristic of being scalable since it is formed by five different modules. The first module is News Extraction System which is in charge of obtaining all the available in the required URLs. The second module is the Orchestrator of all the tasks in the project. News Analysis System is the third module and it is based in finding sentiments and emotions in the articles extracted. The fourth module is the Indexing System which is in charge of storing and classifying this data. The last module is News Visualization System and it finally displays in a dashboard all the information processed before.

5.3 Achieved goals

This section is made in order to explain the progress made in comparing the initial impressions and the final results as presented in this document.

Design and implement a sports news extraction system When creating a new project one of the key parts is data. Consequently, our project is nothing without a good tool to extract the information desired. This is the reason why this achievement was crucial to continue with the project. To reach this goal we have used Scrapy.

Build a pipeline for sentiment and emotion analysis Once we obtained the data using the news extraction system, next step was to analyze it in order to have useful information to be compared. So it was necessary to implement this pipeline using Luigi. Inside this pipeline we used Senpy to effect the analysis and Elasticsearch to index data.

Design a dashboard based on widgets All that analyzed data was useless if we didn't create a dashboard with different views and widgets to show the user the sentiment and emotion evolution. We achieve this goal using Polymer Web Components and D3.js library.

Evaluate the objectivity of different medias This final goal couldn't be done without the previous achievements. We used the dashboard created previously and all data analyzed to show different widgets comparing sentiments and emotions in different medias.

5.4 Problems faced

During the development of this project, not everything went as expected on first try, so we had to face some problems listed next.

- **Sentiment analysis:** Our first idea when talking about sentiment analysis was to use the Senpy plugin called *"sentiText"*. After a while using it, we realized that some theoretically positive news talking about happiness and joy were classified as negative.

  In this point we decided to implement other plugin called *"meaningCloud"* that was much more accurate.

- **Elasticsearch version:** We had to face this problem when developing News Visualization System. Polymer Web Components and Elasticsearch need to be connected in order to display the desired content. The way of connecting this two technologies has changed thru versions and our problem came when we were using the latest Elasticsearch version with the way of connecting used in previous versions. Because of this reason it was needed to update Polymer code.

5.5 Future work

Although this project has been developed as much as possible, there are always new improvements to make. Some possible features to add in the future are explained next.

Adding new teams and newspapers At the moment we have analyzed information from just four newspapers and three football teams. Next step would be to add new teams and newspapers in order to compare them with the ones that we have and make the study of sentiments and emotions more complete.

Creating new dashboard about different topic In our project we have focused in football news, but this development could be extrapolated to other different topics. First example could be other sports, but it also could be focused in topics like music news comparing songs and singers or cinema news comparing movies and actors.

Adding new widgets We have implemented a dashboard with a few widgets, but there are lots of other possibilities when using Polymer Web Components and D3.js. New widgets could be added with new data being compared or shown.

Bibliography

[1] E. Conde-Sánchez, "Development of a Social Media Monitoring System based on Elasticsearch and Web Components Technologies," Master's thesis, ETSI Telecomunicación, June 2016.

[2] J. F. Sánchez-Rada, C. A. Iglesias, I. Corcuera-Platas, and O. Araque, "Senpy: A Pragmatic Linked Sentiment Analysis Framework," in *Proceedings DSAA 2016 Special Track on Emotion and Sentiment in Intelligent Systems and Big Social Data Analysis (SentISData)*, October 2016.

[3] C. A. Iglesias, J. F. S'anchez-Rada, G. Vulcu, and P. Buitelaar, "Linked Data Models for Sentiment and Emotion Analysis in Social Networks," in *Sentiment Analysis in Social Networks*. Morgan Kauffman, October 2016, ch. Linked Dat, pp. 46–66. [Online]. Available: http://store.elsevier.com/Sentiment-Analysis-in-Social-Networks/Federico-Alberto-Pozzi/isbn-9780128044124/[1]

[4] M. M. Bradley and P. J. Lang, "Affective norms for english words (anew): Instruction manual and affective ratings," Technical report C-1, the center for research in psychophysiology, University of Florida, Tech. Rep., 1999.

[5] T. Pedersen, S. Patwardhan, and J. Michelizzi, "Wordnet:: Similarity: measuring the relatedness of concepts," in *Demonstration papers at HLT-NAACL 2004*. Association for Computational Linguistics, 2004, pp. 38–41.

[6] A. Pascual Saavedra, "Development of a dashboard for sentiment analysis of football in twitter based on web components and d3. js," 2016.

[7] D. Kouzis-Loukas, *Learning Scrapy*. Packt Publishing Ltd, 2016.

[8] G. Klyne and J. J. Carroll, "Resource description framework (rdf): Concepts and abstract syntax," 2006.

[9] J. R. Smith and P. Schirling, "Metadata standards roundup," *IEEE MultiMedia*, vol. 13, no. 2, pp. 84–88, 2006.

[10] D. Wood, M. Zaidman, L. Ruth, and M. Hausenblas, *Linked Data*. Manning Publications Co., 2014.

[11] M. Bostock, "D3. js-data-driven documents (2016)," *URL: https://d3js. org*, 2016.