UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESIGN AND DEVELOPMENT OF A SENTIMENT ANALYSIS SYSTEM ON FACEBOOK FROM POLITICAL DOMAIN

PABLO ARAMBURU GARCÍA

Junio 2017

TRABAJO FIN DE GRADO

Título:	Diseño y Desarrollo de un Sistema de Análisis de Sentimientos en Facebook en el Ámbito Político.
Título (inglés):	Design and Development of a Sentiment Analysis System on Facebook from Political Domain.
Autor:	Pablo Aramburu García
Tutor:	Carlos A. Iglesias Fernández
Departamento:	Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:

Vocal:

Secretario:

Suplente:

FECHA DE LECTURA:

CALIFICACIÓN:

Resumen

Este documento es el resultado de un proyecto cuyo principal objetivo ha sido desarrollar y desplegar un cuadro de mandos que muestra los resultados del análisis de información sobre los principales partidos políticos en sus páginas públicas de Facebook, basado en Web Components.

Para ello, la primera tarea que se ha realizado es la extracción de información. Esta extracción se ha desarrollado en Python, creando un flujo entre Facebook, Senpy y Elasticsearch que está gestionado por Luigi. Así, esta herramienta nos ayuda a analizar toda la información y poder almacenarla en Elasticsearch; haciendo posible su acceso posteriormente para mostrarla.

A continuación, y con la ayuda de Sefarad, se ha desplegado toda esa información en forma de widgets que muestran todas las estadísticas, y que ayudan al usuario a interpretar de una manera rápida y visual toda la información disponible.

Como resultado, este proyecto permitirá realizar un estudio y análisis de esta red social, haciendo énfasis en las emociones y sentimientos mostradas por sus usuarios y centrándose en este caso en los principales partidos políticos; pero pudiéndose aplicar en el futuro a cualquier otra rama de la sociedad.

Palabras clave: Sefarad, Senpy, Elasticsearch, Luigi, Polymer, Data Scraping, Web Component, Política, Facebook

Abstract

This thesis is the result of a project whose main goal has been to develop and deploy a dashboard showing the information analysis' results about political domain on Facebook, based on Web Components.

To do so, the first task carried out has been the data extraction. This was developed on Python language, setting up a pipeline between Facebook, Senpy and Elasticsearch, managed by Luigi. This tool help us to analyse data and store it in Elasticsearch; making viable its access to view it.

Right after the first task and using Sefarad, data has been deployed in widgets which show statistics estimated from it; helping users to read data quickly and easily on a striking interface.

As a result, this project will allow us to make a Facebook research and analysis, making emphasis on emotions and sentiments that users show through it. This exploration will focus on the main Spanish political parties; but it can be applied to any social subject.

Keywords: Sefarad, Senpy, Elasticsearch, Luigi, Polymer, Data Scraping, Web Component, Politic, Facebook

Agradecimientos

Gracias a mi tutor Carlos, que me ha sabido guiar y ayudar durante todo el proceso de este proyecto.

A toda mi familia y, en particular, a mis padres y mi hermana, que me han sabido apoyar y animar en todo momento a lo largo de estos cuatro años de carrera.

A mis amigos de La Coruña, que los conozco desde que éramos pequeños y han estado en todo momento.

También a todas esas personas que he conocido durante mi estancia en Madrid, cada una de ellas aportándome algo diferente y animándome en la nueva etapa de mi vida que empezaba fuera de mi hogar.

Y en último lugar, gracias a los grandes amigos que me llevo de esta universidad y a mis compañeros de piso, que han estado al pie del cañón conmigo cuando los he necesitado.

A todos vosotros, gracias.

Contents

Resumen **V**

Abstract **VII**

Agradecimientos **IX**

Contents **XI**

List of Figures **XV**

1 Introduction **1**

1.1 Context 1

1.2 Project goals 2

1.3 Project tasks 2

1.4 Structure of this document 3

2 Enabling Technologies **5**

2.1 Sefarad 5

2.1.1 Luigi 6

2.1.2 Senpy 7

2.1.3 Elasticsearch 8

2.1.4 Polymer 8

2.2 Web Scraping 9

2.2.1 Obtaining Data from Facebook 10

2.2.1.1 Facebook Graph API 10

2.2.1.2 Facebook Graph API Explorer 11

2.3 Conclusion 11

3 Architecture **13**

3.1 Overview 13

3.2 Orchestrator 14

3.3 Facebook Scraper System 15

3.4 Analysis System 19

3.4.1 sentiText 19

3.5 Indexing System 21

3.6 Visualization System 22

3.6.1 Structure 22

3.6.2 Elements 23

3.6.2.1 Paper-material 23

3.6.2.2 Number-chart 24

3.6.2.3 Google-chart 24

3.6.2.4 Comments-table 26

3.6.2.5 Field-chart 27

4 Case of study **29**

4.1 Extracting data 29

4.2 Analysing data 31

4.3 Indexing data 31

4.4 Displaying data 31

4.5 Conclusions 34

5 Conclusions and Future work **37**

5.1 Conclusions 37

5.2 Achieved goals 38

5.3 Problems faced 38

5.4 Future work 39

Bibliography **40**

List of Figures

2.1 Sefarad architecture 6

2.2 Senpy processes 7

2.3 Facebook Graph API Explorer 11

2.4 Final Structure of the Project 11

3.1 Architecture of the Project 14

3.2 Example of a Facebook public page 18

3.3 Example of a result of Scraping System 19

3.4 Accessing Elasticsearch 21

3.5 Dashboard Mock-up 23

3.6 Paper-material Widget 24

3.7 Number-chart Widget 24

3.8 Google-chart Column Widget 25

3.9 Google-chart Pie Widget 26

3.10 Comments-table Widget 27

3.11 Sentiment evolution widget 27

4.1 First and second dashboard's section 32

4.2 Third dashboard's section 33

4.3 Third dashboard's section (2) 34

Listings

3.1 Executing Luigi pipeline 15
3.2 Generating a valid access-token 16
3.3 Retry in error case 16
3.4 IDs of the four political parties we are going to analyse 17
3.5 Example of extracting data from Facebook 17
3.6 URL requesting a sentiText analysis 20
3.7 Result of the URL requesting a sentiText analysis 20
3.8 Starting Elasticsearch 21

CHAPTER 1

Introduction

1.1 Context

Nowadays, social networks have gained a lot of importance in people's daylife. This importance has achieved political panorama [4]; political parties use these social networks not only for the purpose of self-presentation but also to maintain contact with society directly and easily, and spreading every subject they are into. Having presence on social networks is considered a necessary step on public organizations, showing closeness to the population. In addition, it is possible to see reflected in these social networks multiple impacts and effects that their movements have in the population [12], making viable its analysis.

With all those movements made on social networks, this project will be based on Sentiment Analysis Techniques [7]. Political parties are constantly publishing news, comments or just simply interactions between them, so it will be easy and useful to establish a sentiment relation between them. Furthermore, any person has the chance of answering or publishing something in political parties' Facebook pages; so not only they will be analysed from the inside, but also how they look from outside. The result of this analysis could be used in the future for statistics of voting intentions, for example.

As we have introduced previously, the main aim of this project is to develop and deploy a dashboard which reflects the results of the analysis of the population's reactions and feelings to the last publications of four principal Spanish political parties in their public Facebook's pages; making them easy to visualize and understand. To do so, the project will be divided into different phases, using multiple tools on each one. The first task of the project involves a scraper data in Facebook, trying to gather as much information as possible in order to analyse it later. The result of this Facebook scraping will be stored in a json file, with a predefined structure. Secondly, this information will be gathered on ElasticSearch and analysed helped by Senpy. Luigi, from Python, will help doing these issues. Finally, the results of that analysis will be displayed on a Dashboard managed by Sefarad, making it easy to visualize for the user. Programming languages such as Python, HTML, CSS or JavaScript will be used on every step of this project.

1.2 Project goals

In this project we will make an analysis of different reactions population have to the latest political parties' movements on their Facebook pages.

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

- Get access to people reactions and comments, and collect them.
- Build a pipeline for sentiment and emotion analysis.
- Design a dashboard based on widgets that will show the information collected in a striking and attractive way.

1.3 Project tasks

During this project, the following tasks will be implemented:

- Learn how to use and implement Facebook Grapher API, in order to extract data from this social network.
- Store Facebook data in a json file, with a predefined structure.
- Analyse Facebook data with Senpy.
- Store data analysed with ElasticSearch.

- Build and deploy a dashboard showing the information collected and stored in ElasticSearch.

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 explains the context in which this project is developed, describes the main goals to achieve in this project and introduce the topic to the audience.
Chapter 2 provides a description of the main technologies used oh this project.
Chapter 3 describes the architecture of this project, including the design phase and implementation details.
Chapter 4 presents a detailed example of this project.
Chapter 5 discusses the conclusions extracted from this project, problems faced and suggestions for a future work.

CHAPTER 2

Enabling Technologies

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

In order to accomplish this task we have been helped by another project called Sefarad and developed in the GSI (Grupo de Sistemas Inteligentes) of ETSIT-UPM, giving us the chance of showing data on a easy and striking way.

Extracting data has been possible thanks to Facebook Graph API. Analysing and storing data from public pages of Facebook has been made through a pipeline created by Luigi. This pipeline allow us to connect the Scraping System, Senpy and ElasticSearch in order to achieve those tasks, and later visualize its results.

2.1 Sefarad

Sefarad[1] is the main tool used in this project due to it includes every system used to achieve the goals of this project, so first of all we will explain what it is. Sefarad is an application developed by the GSI to explore and display data by making SPARQL queries

[1]http://sefarad.readthedocs.io/en/latest/sefarad.html

to the endpoint you choose without writing more code. You can also create your own cores if you have a big collection of data (LMF required). To view the data you want to explore, you can create and customize your own widgets and visualize it through them, or you can use multiple available widgets in the system in order to do so, due to they are reusable.

It mainly uses four technologies, all gathered in one pipeline. These technologies are shown in Fig. 2.1, as explained below:

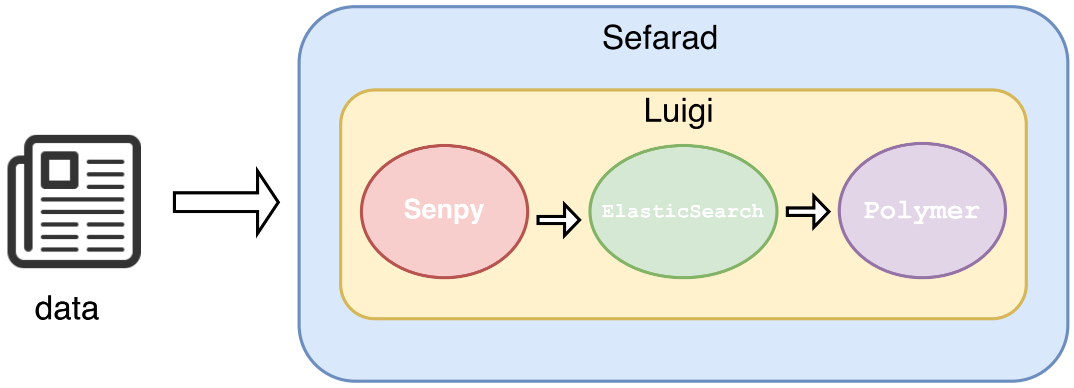


Figure 2.1: Sefarad architecture

2.1.1 Luigi

Luigi[2] is a Python package that helps you in the process of building complex pipelines of batch jobs. It handles dependency resolution, workflow management, handling failures, visualization, command line integration, and much more.

The purpose of Luigi is to address all the pipelines typically associated with long-running batch processes. Many tasks can be chained with Luigi's help; these tasks can be anything, but are typically long running things like Hadoop jobs, downloading data from databases, running machine learning algorithms, uploading data to databases, or anything else.

In this project, Luigi is used to create a pipeline that manages every task composing Sefarad, making easier and faster to develop and deploy the project.

[2]https://github.com/spotify/luigi

2.1.2 Senpy

Senpy [11] is a technology also developed by the GSI. It is an open source software and uses a data model to make an analysis of feelings and emotions to it.[3] It is based on semantic vocabularies (e.g. NIF, Marl, Onyx) and formats (turtle, JSON-LD, xml-rdf).

This framework consists of two main modules: Senpy core, including the building block of the service, and Senpy plugins, which consist of the analysis algorithm. The following picture shows a simplified example of Senpy processes.

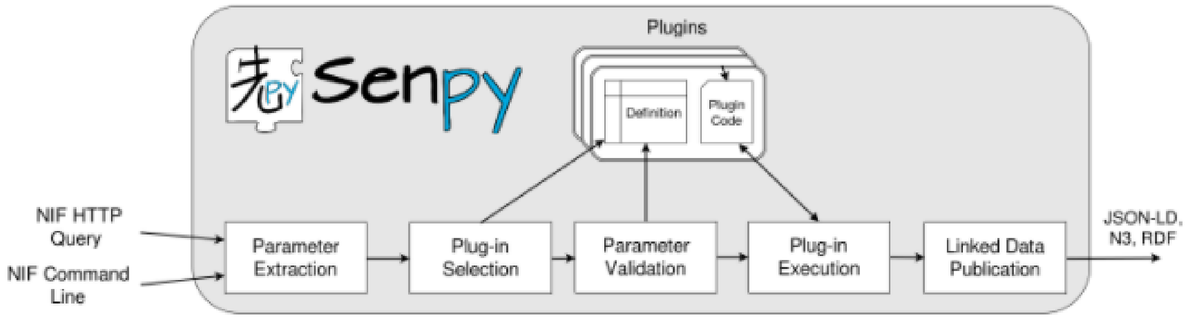


Figure 2.2: Senpy processes

There are seven different plugins available to use:

- sentiText: It extracts sentiments from the text given as input. It is adapted for English and Spanish.
- meaningCloud: an adaptation made from the MeaningCloud[4] Sentiment Analysis API. This plugin is available in English and Spanish too.
- emoTextWordnetAffect: it is based on the hierarchy of WordnetAffect [9] to calculate the emotion of the sentence. This plugin is only available in English, which represents a disadvantage if you want to analyse Spanish texts.
- sentiment140: this plugin allows users to discover the sentiment of a brand, product, or topic on Twitter.[5] It is possible to analyse English and Spanish texts.
- emoTextAnew: it extracts the VAD (*valence-arousal-dominance*) of a sentence by matching words from the ANEW dictionary. [1]

[3] http://senpy.readthedocs.io/en/latest/
[4] https://www.meaningcloud.com/developer/sentiment-analysis
[5] http://help.sentiment140.com/home

- affect: It allows users to analyze sentiments and emotions at the same time by choosing one sentiment plugin and another emotions one.

- vaderSentiment: it uses the software from vaderSentiment [3] to calculate the sentece's sentiment.

2.1.3 Elasticsearch

Elasticsearch [2] is a search engine based on Lucene. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.

It is developed in Java and is released as an open source under the terms of the Apache License, which means anyone can have access to it.

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 request body or just using a simple query string as a parameter.

Elasticsearch[6] organizes data in indexes. An Elasticsearch index is some type of data organization mechanism, that allows the user to partition data in a certain way. There are two main ways for adding data to an Elasticsearch index:

- The first one allows us to capture real-time data and is based on using Logstash. In this case a configuration file is needed, which describes where data is going to be stored and the query terms that are going to be indexed.
- The second one is easier and faster. It consists in adding a whole index by using the Elasticsearch's bulk API [8]. In this case you just need all your data stored in a JSON file and you manually add the index where data is going to be stored.

2.1.4 Polymer

Polymer[7] is an open-source JavaScript library for building web applications using Web Components. This project is possible thanks to Google developers and contributors on GitHub contributing to the development of this library. Modern design principles are implemented as a separate project using Google's Material Design design principles.

[6] https://en.wikipedia.org/wiki/Elasticsearch
[7]https://en.wikipedia.org/wiki/Polymer_(library)

Polymer helps users on building their very own custom HTML elements, making this task easier and faster. By being based on the Web Components API's built in the browser, Polymer elements are interoperable at the browser level, and can be used with other frameworks or libraries that work with modern browsers.

This fact can make the process of building web applications easier and more efficient, due to programming time is significantly decreased. These custom elements are particularly useful for building re-usable UI components. This is very effective: instead of being continually re-building a specific element for multiple projects and in different frameworks, you can define this element once using Polymer, and then reuse it throughout your project or in any future one.

Polymer uses a declarative syntax to make the creation of your own custom elements easier. They use standard web technologies: HTML to define the structure of the element, CSS for style personalization and 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 on a direct way to simply custom elements' creation.

Furthermore, Web Components[8] are based on four main and independent (they can be used separately, giving more resilience) features [6]:

- Custom Elements: APIs to create new elements, providing a component model for the web.
- Templates: Allows documents to contain inert chunks of DOM.
- Shadow DOM: Encapsulated DOM and styling, with composition.
- HTML Imports: : Imports HTML documents into other documents.

2.2 Web Scraping

Before describing how we obtained all the data from Facebook, it is necessary to explain the meaning of *Web Scraping* [5].

Web Scraping[9] is a computer software technique used for extracting data from websites. It involves two phases: fetching it and extracting from it. Fetching is the download of a page, this task is made by the browser when a user visits a page. Once fetched, the

[8]http://octuweb.com/introduccion-web-components/

[9]https://es.wikipedia.org/wiki/Web_scraping

extraction can take place. The content of a page may be parsed, searched, reformatted, its data copied into a spreadsheet, and so on. Web scrapers typically take something out of a page, to make use of it for another purpose somewhere else.

2.2.1 Obtaining Data from Facebook

In our project, we developed a Facebook Scraper which obtains data from the main political parties' Facebook pages in Spain: "Partido Popular", "Partido Socialista Obrero Espanol", "Podemos" and "Ciudadanos".

2.2.1.1 Facebook Graph API

Facebook Graph API[10] helped us to achieve this task. It is the primary way to get data out of, and put data into, Facebook's platform. It's a low-level HTTP-based API (it works with any language having an HTTP library, such as cURL and urllib) that you can use to sequentially query data, post new stories, manage ads, upload photos, and perform a variety of other tasks that an app might implement.

It is composed of:

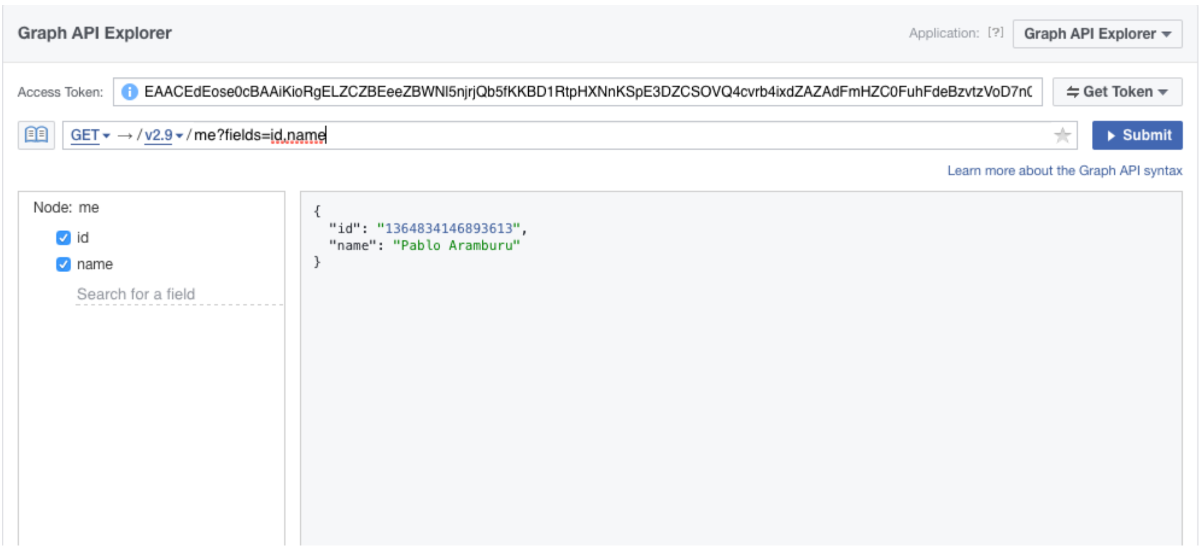
- Nodes: basically "things" such as a user, a photo, a page or a comment.
- Edges: connections between two nodes, such as photos of a page, or comments of a photo.
- Fields: information about nodes, such as a birthday of a person, or the name of a Page.

This API requires authentication via access tokens[11]. Users can get Short-Term tokens, but as their name suggests, they expire quickly, so they are not recommended due to their short lifetime. Because of it, we decided to access with a Long-Term token concatenating the App ID from a user-created App and the App Secret, giving us unlimited access to the platform.

[10] https://developers.facebook.com/docs/graph-api/overview
[11]http://minimaxir.com/2015/07/facebook-scraper/

2.2.1.2 Facebook Graph API Explorer

Facebook gives users the chance to try their API and make different calls on a visual platform named Facebook Graph API Explorer, which is really helpful and intuitive. With this API Explorer, users can prove every Facebook API function. An example of a simple request, and its response, made to this API is given below in Fig 2.3.



```
EAACEdEose0cBAAiKioRgELZCZBEeeZBWNl5njrjQb5fKKBD1RtpHXNnKSpE3DZCSOVQ4cvrb4ixdZAZAdFmHZC0FuhFdeBzvtzVoD7n(
```

```
GET → /v2.9 / me?fields=id,name
```

```
{
  "id": "1364834146893613",
  "name": "Pablo Aramburu"
}
```

Figure 2.3: Facebook Graph API Explorer

2.3 Conclusion

Once we know every technology used is on this project, we are able to shape the final structure of this project, as it is shown in Figure 2.4, including previous Sefarad structure connected to the Facebook Scraper.

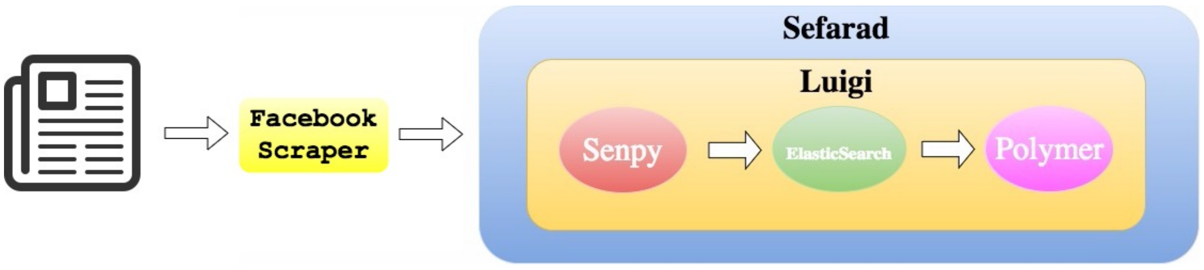


Figure 2.4: Final Structure of the Project

CHAPTER 3

Architecture

In this chapter, we are going to explain the architecture of this project. First of all, we will give an overview presenting a global vision about the whole architecture of this project. Finally, we will explain each module integrated into this project in detail, giving figures, listings and examples to make it easer to the reader.

3.1 Overview

This section gives us a general view of the modules used in this project and their connections. These links are shown in the Fig. 3.1 below.

- *Orchestrator*: in this case, as it is mentioned before, Luigi is used to create a pipeline that manages different tasks of this project, making them easier to us.
- *Facebook Scraper System*: this part of the project is used to extract data from Facebook, obtaining it through Facebook Graph API and analysed later to achieve our goals.
- *Analysis System*: this module uses Senpy to extract emotions and sentiments from Facebook data obtained on the previous task.

- *Indexing System*: this part of the project is in charge of indexing the information we obtained from the previous analysis, using ElasticSearch.

- *Visualization System*: this is the final stage of the project. It is responsible for processing data and showing it in different ways, making them attractive to users.

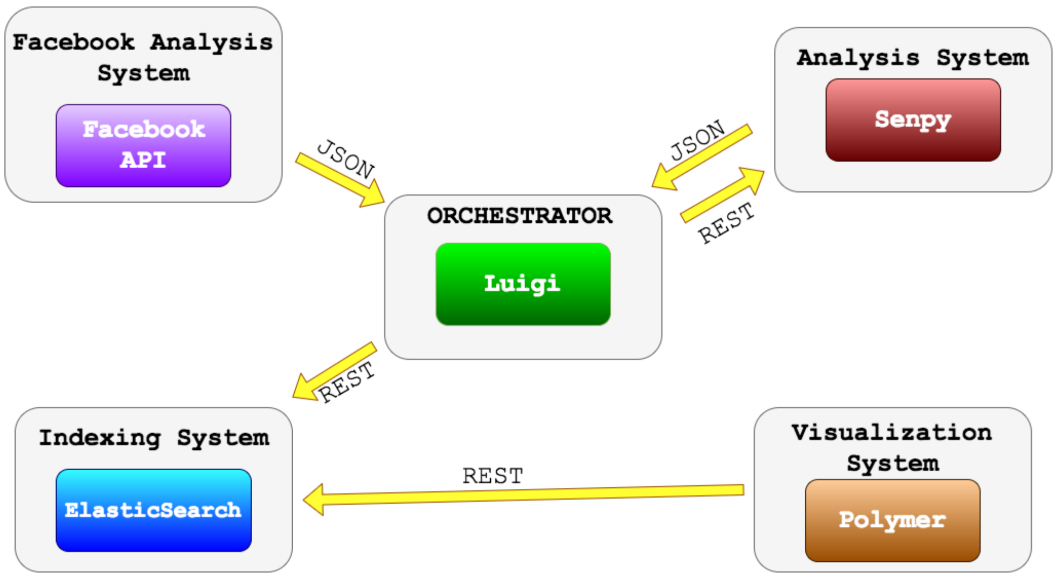


Figure 3.1: Architecture of the Project

3.2 Orchestrator

Luigi [10] is a Python package developed by Spotify used to build complex pipelines of batch jobs, as it is mentioned before. In our project, we use Luigi as an orchestrator, building a pipeline between the Facebook Scraper System, the Analysis System and the Indexing System.

We need a script describing the pipeline Luigi has to follow, which includes Senpy and ElasticSearch. This pipeline will be explained next.

The pipeline used on this project is composed of three tasks:

- *FetchDataTask*: this task reads the JSON file indicated as a parameter and it checks its structure and syntax. In our case, this JSON file is the result of the Facebook Scraper System.

- *SenpyTask*: this one loads data fetched with previous task and send it to Senpy tool in order to analyse data retrieved and check sentiments expressed. Users can choose the analysis they want to make by selecting the plugin they wan to use. It stores data in a JSON file.
- *Elasticsearch*: the last task of the pipeline is in charge of indexing the JSON file obtained from the previous task into the ElasticSearch index indicated as a parameter, so user can have full access to it later.

An example of the commands used to execute this pipeline is given in the next Listing 3.1, and are explained after.

Listing 3.1: Executing Luigi pipeline

```
python -m luigi --module sefarad Elasticsearch --index politicos --doc-type
    facebook --filename facebookPolitics.json --local-scheduler
```

The different parameters used on this execution are:

- *module*: it is the script you use to manage the different tasks.
- *Elasticsearch*: here you denote which is the last task you want to execute in the pipeline. In our case, it was named Elasticsearch.
- *index*: this is the name of the partition of the data as it will appear on Elasticsearch. It is important to establish an easy name to remember, so you can find it fast later.
- *doc-type*: this is the subindex data will have in the database.
- *filename*: this is the JSON file containing all data extracted before.

3.3 Facebook Scraper System

This is the first and most important task of the project. It is in charge of extracting data from Facebook and store it in a JSON file, ready to its analysis and indexing.

As we have introduced previously in Chapter 2.2, Facebook gives users the chance to extract and interact with their data through an API available to everyone who wants to use it. This API allows users to post, get or even delete Facebook data, as long as they have permission to do so.

Facebook Graph API is not a public API; it requires user authentication via access tokens. This authentication can cause problems because these access tokens are Short-Term, so they expire quickly and users have to renew them in order to continue developing on this platform. This fact can represent a problem when developing an app which uses Facebook updated information, because programmer has to be continue renewing his tokens. In this project, we used a trick allowed by Facebook. We used as our access-token the concatenation of an user-created Application and the App Secret (both of them given by Facebook), making sure we have full access to the API, as illustrated in Listing 3.2.

Listing 3.2: Generating a valid access-token

```
  # I generate the access_token by creating an empty Facebook App which
  # gives me the app_id and app_secret needed in order to have full access
  # to the API.
  app_id = "1352103248145811"
  app_secret= "ee1c01590f72108218e533f67c955454"
  # Concatening them I am sure it won‘t expire.
  access_token= app_id + "|" + app_secret
```

Sometimes connection between our App and Facebook Graph API can be lost, or an error can appear. To solve this situation, we developed the following function.

Listing 3.3: Retry in error case

```
def request_until_succeed(url):
  req = urllib2.Request(url)
  success = False
  while success is False:
    try:
      response = urllib2.urlopen(req)
      if response.getcode() == 200:
        success = True
    except Exception, e:
      print e
      time.sleep(5)
      print "Error for URL %s: %s" % (url, datetime.datetime.now())
  return response.read()
```

Once we make sure we have access to Facebook Graph API, it is time to start scraping data from it. First of all, we have to know the ids of the public pages we want to extract information to build the url we are going to make the request. These ids can easily be found in the URL of the page we want to get access to. In our case, they are: "pp", "psoe", "ahorapodemos" and "Cs.ciudadanos", as it is illustrated in Listing 3.4.

Listing 3.4: IDs of the four political parties we are going to analyse

```
#We store the ids we are going to analyse
ids = ["pp","psoe","ahorapodemos","Cs.Ciudadanos"]
```

Reading the documentation of Facebook Graph API[1] we can find out a lot of information, including what parameters we can extract from pages, as long as they are public so everyone can access it. Users can extract people's reactions to a publication (e.g. like, love, angry, wow, haha, sad ...), number of comments and their message, the time when post was made, the message of the post and the id of the post, for example.

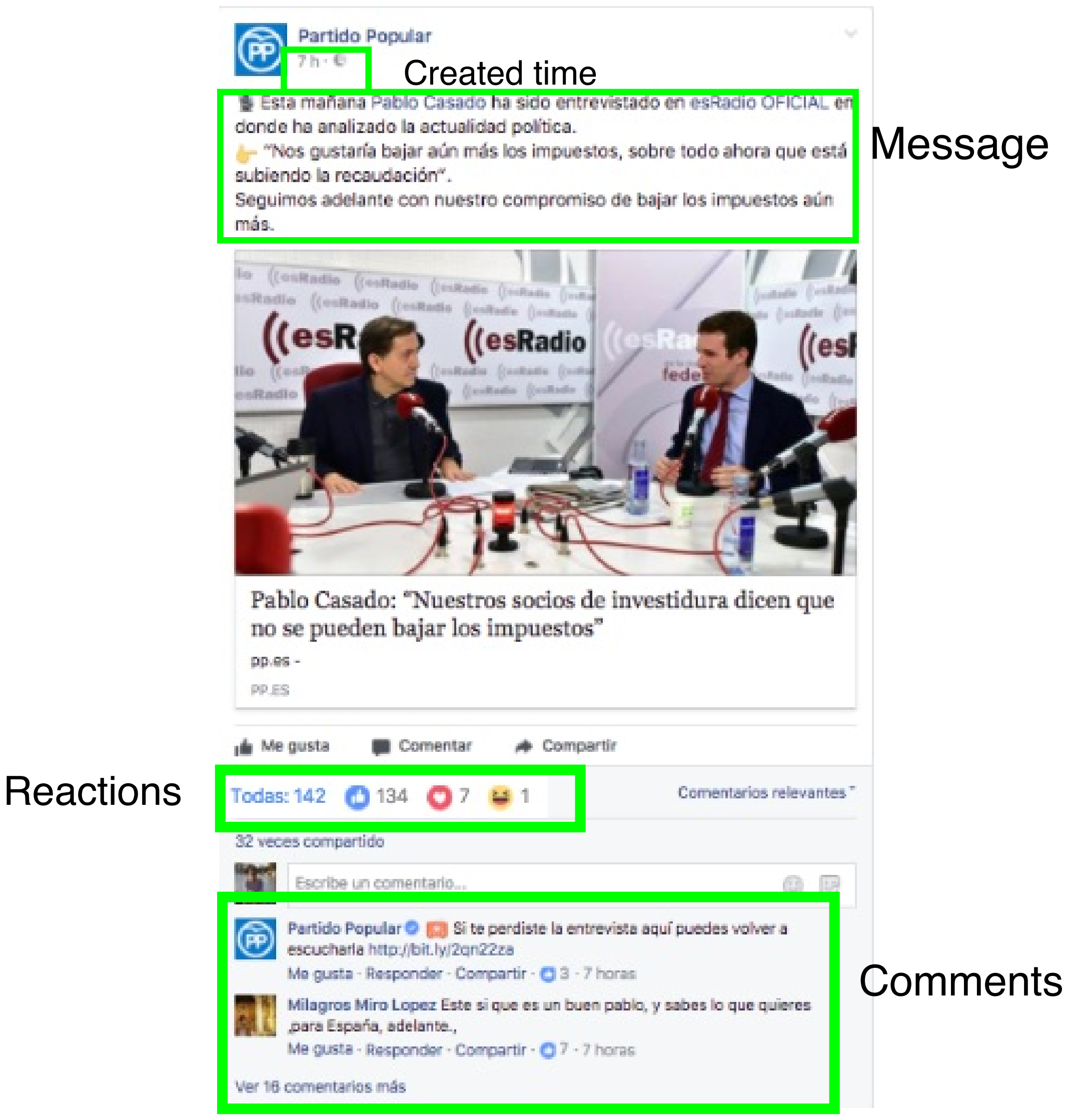
Fig. 3.2 shows a post made in a public page of Facebook, including every element mentioned before.

Listing 3.5: Example of extracting data from Facebook

```
# We use this function to extract data from Facebook
def getFBPageFeedData (page_id, access_token, num_status):
base = "https://graph.facebook.com/v2.9"
node = "/" + page_id + "/feed"
parameters = "/?fields=created_time,message,name,id,
reactions.type(LIKE).summary(total_count).limit(0).as(like),
reactions.type(LOVE).summary(total_count).limit(0).as(love),
reactions.type(WOW).summary(total_count).limit(0).as(wow),
reactions.type(HAHA).summary(total_count).limit(0).as(haha),
reactions.type(SAD).summary(total_count).limit(0).as(sad),
reactions.type(ANGRY).summary(total_count).limit(0).as(angry),
comments.limit(100).summary(true)
&limit=%s&access_token=%s"
%(num_status, access_token)
url = base + node + parameters
data = json.loads(request_until_succeed(url))
print "Analisis de %s realizado!" %page_id
return data
```

[1]https://developers.facebook.com/docs/graph-api/reference/

In order to make this request, in our project we used the function shown in the Listing 3.5, which extracts the data we will analyse at the end of the system.



Partido Popular
7 h ·

Esta mañana Pablo Casado ha sido entrevistado en esRadio OFICIAL en donde ha analizado la actualidad política.
👉 "Nos gustaría bajar aún más los impuestos, sobre todo ahora que está subiendo la recaudación".
Seguimos adelante con nuestro compromiso de bajar los impuestos aún más.

Pablo Casado: "Nuestros socios de investidura dicen que no se pueden bajar los impuestos"
pp.es -
PP.ES

Todas: 142 134 7 1
32 veces compartido

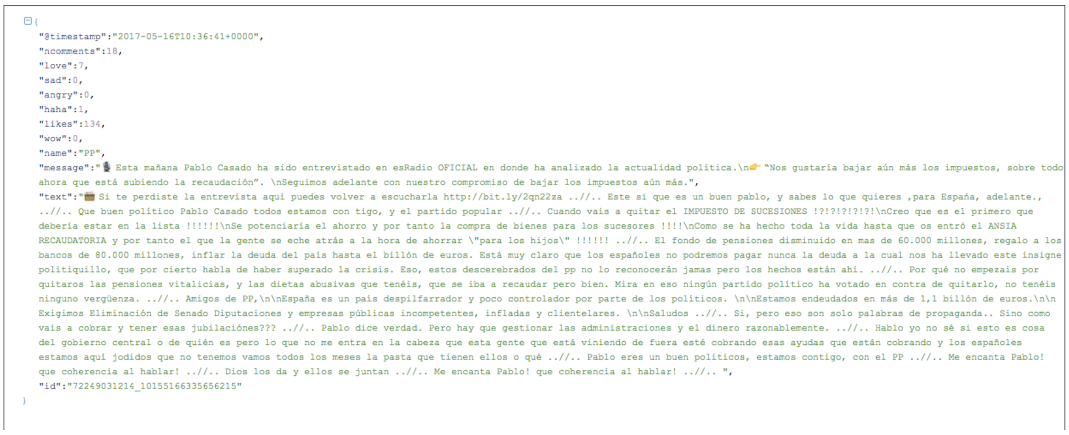
Partido Popular Si te perdiste la entrevista aquí puedes volver a escucharla http://bit.ly/2qn22za
Me gusta · Responder · Compartir · 3 · 7 horas

Milagros Miro Lopez Este si que es un buen pablo, y sabes lo que quieres ,para España, adelante.,
Me gusta · Responder · Compartir · 7 · 7 horas

Ver 16 comentarios más

Figure 3.2: Example of a Facebook public page

The result of the call made by this function is a JSON file which includes all the information asked as *fields* on the url parameter. Here you can see an example of the result of this Scraping System: a JSON file including all the information extracted from the post showed previously on Figure 3.3.



```
{
    "@timestamp":"2017-05-16T10:36:41+0000",
    "ncomments":18,
    "love":7,
    "sad":0,
    "angry":0,
    "haha":1,
    "likes":134,
    "wow":0,
    "name":"PP",
    "message":"🎙 Esta mañana Pablo Casado ha sido entrevistado en esRadio OFICIAL en donde ha analizado la actualidad política.\n👉 "Nos gustaría bajar aún más los impuestos, sobre todo
    ahora que está subiendo la recaudación". \nSeguimos adelante con nuestro compromiso de bajar los impuestos aún más.",
    "text":"📼 Si te perdiste la entrevista aquí puedes volver a escucharla http://bit.ly/2qn22za ..//.. Este si que es un buen pablo, y sabes lo que quieres ,para España, adelante.,
    ..//.. Que buen político Pablo Casado todos estamos con tigo, y el partido popular ..//.. Cuando vais a quitar el IMPUESTO DE SUCESIONES !?!?!?!?!?!\nCreo que es el primero que
    debería estar en la lista !!!!!!\nSe potenciaría el ahorro y por tanto la compra de bienes para los sucesores !!!!\nComo se ha hecho toda la vida hasta que os entró el ANSIA
    RECAUDATORIA y por tanto el que la gente se eche atrás a la hora de ahorrar \"para los hijos\" !!!!!! ..//.. El fondo de pensiones disminuido en mas de 60.000 millones, regalo a los
    bancos de 80.000 millones, inflar la deuda del país hasta el billón de euros. Está muy claro que los españoles no podremos pagar nunca la deuda a la cual nos ha llevado este insigne
    politiquillo, que por cierto habla de haber superado la crisis. Eso, estos descerebrados del pp no lo reconocerán jamas pero los hechos están ahí. ..//.. Por qué no empezais por
    quitaros las pensiones vitalicias, y las dietas abusivas que tenéis, que se iba a recaudar pero bien. Mira en eso ningún partido político ha votado en contra de quitarlo, no tenéis
    ninguno vergüenza. ..//.. Amigos de PP,\n\nEspaña es un país despilfarrador y poco controlador por parte de los políticos. \n\nEstamos endeudados en más de 1,1 billón de euros.\n\n
    Exigimos Eliminación de Senado Diputaciones y empresas públicas incompetentes, infladas y clientelares. \n\nSaludos ..//.. Si, pero eso son solo palabras de propaganda.. Sino como
    vais a cobrar y tener esas jubilaciónes??? ..//.. Pablo dice verdad. Pero hay que gestionar las administraciones y el dinero razonablemente. ..//.. Hablo yo no sé si esto es cosa
    del gobierno central o de quién es pero lo que no me entra en la cabeza que esta gente que está viniendo de fuera esté cobrando esas ayudas que están cobrando y los españoles
    estamos aquí jodidos que no tenemos vamos todos los meses la pasta que tienen ellos o qué ..//.. Pablo eres un buen politicos, estamos contigo, con el PP ..//.. Me encanta Pablo!
    que coherencia al hablar! ..//.. Dios los da y ellos se juntan ..//.. Me encanta Pablo! que coherencia al hablar! ..//.. ",
    "id":"72249031214_10155166335656215"
}
```

Figure 3.3: Example of a result of Scraping System

3.4 Analysis System

In our project we used an Analysis System to extract sentiments showed by population on their comments to the last political parties' publications: this system is Senpy. As it is explained in Chapter 2.1.2, Senpy allows users to make sentiment and emotions analysis of the information they provide through different plugins available to everyone who wants to use them.

In order to make our analysis, we have chosen sentiText widget from the multiple options Senpy offers you. Other options were ruled out because they did not extract what we wanted or they did not admit Spanish language.

3.4.1 sentiText

We have used this widget in our project to extract sentiments from comments people make on different publications.

This element extracts the message given as a parameter and calculates its polarity, and if the content of the text is positive, neutral or negative.

The way of using this plugin is by accessing an specific URL with the needed parameters. In our case, we have introduced three parameters when running this URL.

- The first one is called "algo", an abbreviation of algorithm. We use this parameter to introduce the type of analysis we want to run. In our case, it is "sentiText".
- The second parameter is called "language". It refers to the language in what the

content to analyze is written.

- The last parameter is named as ”input” and it is followed by the text we want to make an analysis to.

An easy example of the URL accessed to use this plugin is shown in Listing 3.6 .

Listing 3.6: URL requesting a sentiText analysis

```
http://senpy.cluster.gsi.dit.upm.es/api/?algo=sentiText&language=es&input=
    Estoy%20feliz
```

In this example, we use sentiText as plugin to analyse the text ”Estoy feliz” in Spanish language. The output of the previous URL is given next.

Listing 3.7: Result of the URL requesting a sentiText analysis

```
{
  "analysis": [
    {
      "@id": "sentiText",
      "@type": "marl:SentimentAnalysis",

      "entries": [
        {
          "@id": "Entry0",
          "nif_isString": "Estoy feliz",
          "sentiments": [
            {
              "@id": "Opinion0",
              "marl:hasPolarity": "marl:Positive",
              "marl:polarityValue": "1",
              "prov:wasGeneratedBy": "sentiText"
            }
          ]}]}]}
```

3.5 Indexing System

This system receives all the information extracted and analysed in the previous tasks and indexes it. It works with ElasticSearch, which makes the connection between the Orchestrator and the Visualization Server, explained in the next section.

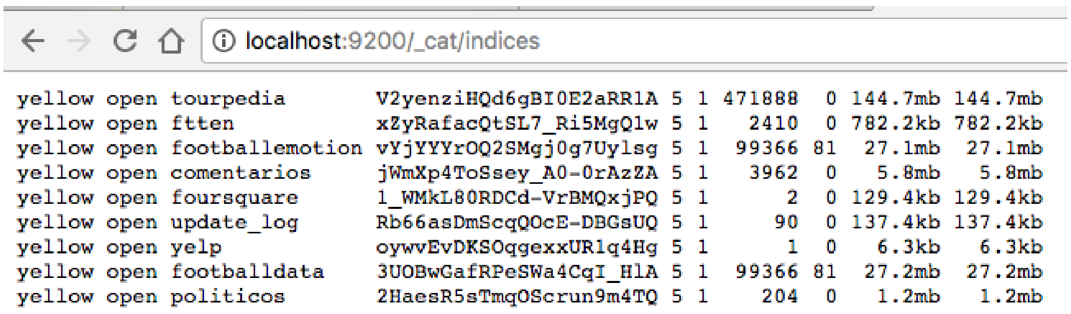
In our project, the pipeline takes this task as it is explained in Section 3.2. We just need to store all the data in a JSON file and the index where it will be stored, and Luigi do it for us. We have used two indexes in this project: while the first one contains information about each post of each political party, the other one stores every comment Facebook's users have made to those posts.

To be able to use and work with ElasticSearch, you should have launched this tool as it is shown in Listing 3.8 .

Listing 3.8: Starting Elasticsearch

```
bin\elasticsearch
```

Once you have uploaded your information, users can make sure their data is fully transfered to the database, ElasticSearch allows you to consult every index and its status by accessing the following URL: *http://localhost:9200/_cat/indices* . An example of the result of accessing it once is launched, is shown next.



```
yellow open tourpedia       V2yenziHQd6gBI0E2aRR1A 5 1 471888  0 144.7mb 144.7mb
yellow open ftten           xZyRafacQtSL7_Ri5MgQ1w 5 1   2410  0 782.2kb 782.2kb
yellow open footballemotion vYjYYYrOQ2SMgj0g7Uylsg 5 1  99366 81  27.1mb  27.1mb
yellow open comentarios     jWmXp4ToSsey_A0-0rAzZA 5 1   3962  0   5.8mb   5.8mb
yellow open foursquare      1_WMkL80RDCd-VrBMQxjPQ 5 1      2  0 129.4kb 129.4kb
yellow open update_log      Rb66asDmScqQOcE-DBGsUQ 5 1     90  0 137.4kb 137.4kb
yellow open yelp            oywvEvDKSOqgexxURlq4Hg 5 1      1  0   6.3kb   6.3kb
yellow open footballdata    3UOBwGafRPeSWa4CqI_HlA 5 1  99366 81  27.2mb  27.2mb
yellow open politicos       2HaesR5sTmqOScrun9m4TQ 5 1    204  0   1.2mb   1.2mb
```

Figure 3.4: Accessing Elasticsearch

3.6 Visualization System

Showing information clearly and easily understood is as much important as collecting and analysing it; so it does not matter if the previous tasks are correctly developed if this last one is not.

The Visualization System is based on Polymer Web Components (as it is mentioned in Chapter 2.1.4) and is developed from Sefarad 4.0.

3.6.1 Structure

First step on the process of creating a visual project should be to structure the space and to think how to place items along it. This can be made by creating a sketch or a mock-up, which can be changed until the end of the project, but will give an overview of how the interface of the application will appear.

In our case, we needed to develop a website showing all the information extracted and analysed in order to provide final users a visual, attractive, interactive and easy to use platform. This is the main goal of the Visualization System, structuring data in a way users can enjoy and easily understand its analysis and statistics.

To do so, we created and designed the following mock-up using the service given by *draw.io*[2]. In this mock-up we show how we structured and distributed space. Firstly, user will find how many posts each political party have published recently. Secondly, we will display a comparison between the main reactions people show about each political party, showing them in a column diagram per reaction. Next, we will present the results of the analysis of the people's comments in two different ways: a pie bar will compare the number of comments on each political parties' posts and finally comments of each post will be displayed on a table accompanied by their sentiment analysis. This mock-up is shown in Fig. 3.5 .

[2]https://www.draw.io/

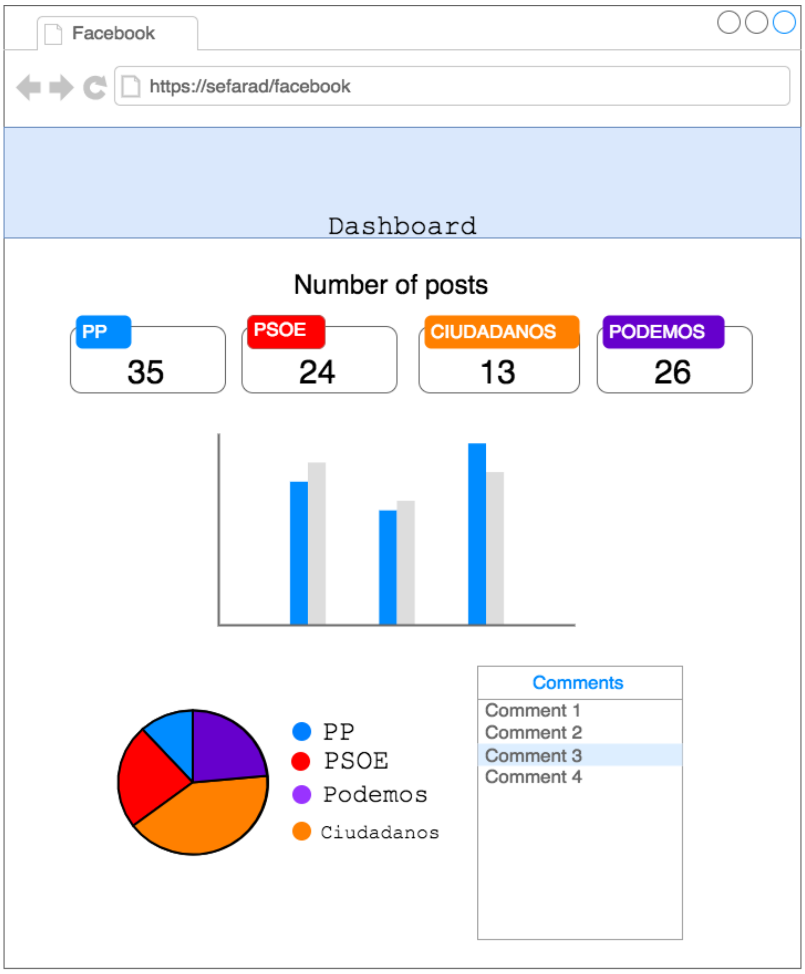


Figure 3.5: Dashboard Mock-up

3.6.2 Elements

Once we knew how to structure space and how information will be displayed, we started developing it and used different widgets that helped us with the task of showing data in an easy and striking way.

3.6.2.1 Paper-material

This component is a text-container[3]. It helps us to divide our dashboard into different sections. It creates the effect of a lifted piece of paper by rendering two shadows on top of each other, so it is very useful when you want to highlight the beginning of a topic in your dashboard.

[3]https://www.webcomponents.org/element/PolymerElements/paper-material

In our project we use paper-material widgets to divide space depending on the topic data will show. We show an example of this element applied to our dashboard on Figure 3.6 .

Figure 3.6: Paper-material Widget

3.6.2.2 Number-chart

Next element is called Number-chart. It analyses data passed as a parameter and extracts how many times a parameter identified as *object* is found on the JSON file containing the information. A number-chart component is completely customizable: it shows information attached to an icon that users can choose from an enormous list of figures, or just add their own pictures. It is also possible to change the text ilustrated.

In our dashboard, we used this component as a counter of how many posts have each political party published on their public pages in a certain period of time. We have decided to include each political party logo to make it more visual and impressive to users; as presented on next Figure 3.7.

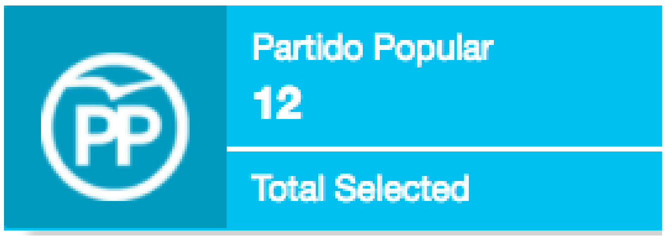


Figure 3.7: Number-chart Widget

3.6.2.3 Google-chart

Number statistics are made with the help of Google and their Google-Charts API [13]. This company provide users an API which estimate and show statistics in different ways and formats on a very simple way.

In our project we use these Google-chart[4] widgets to display different statistics extracted from analysing Facebook data. First of all, we used a column diagram to analyse and compare different Facebook reactions of people to the last publications of political parties, as

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

it is illustrated in Figure 3.8. In this case, we chose a column diagram because it is more visual and easy to compare reactions if they are shown on one diagram per reaction instead of one diagram per political party.

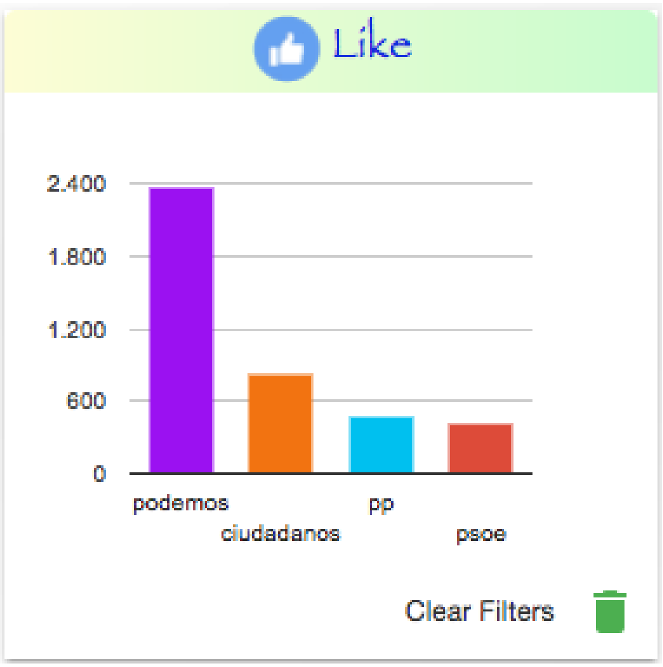


Figure 3.8: Google-chart Column Widget

We also use these Google-chart widgets to compare the activity of the users on each public page. We previously extract information about comments on the last publications of political parties; then we decided to present a comparative between the number of comments each political party had on them. In this case, we thought a pie diagram will accomplish An example of the result of this task is presented in Figure 3.9.
Pie diagrams have been also used to compare the number of positive and negative comments.

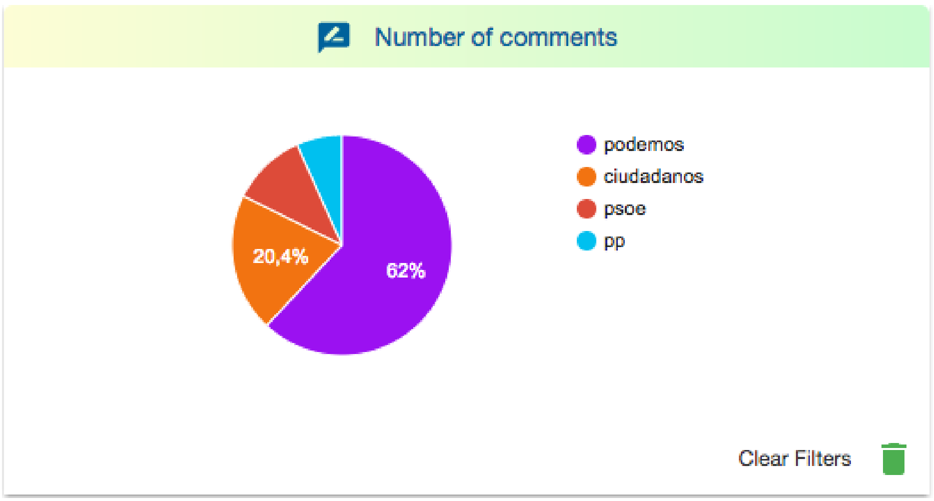


Figure 3.9: Google-chart Pie Widget

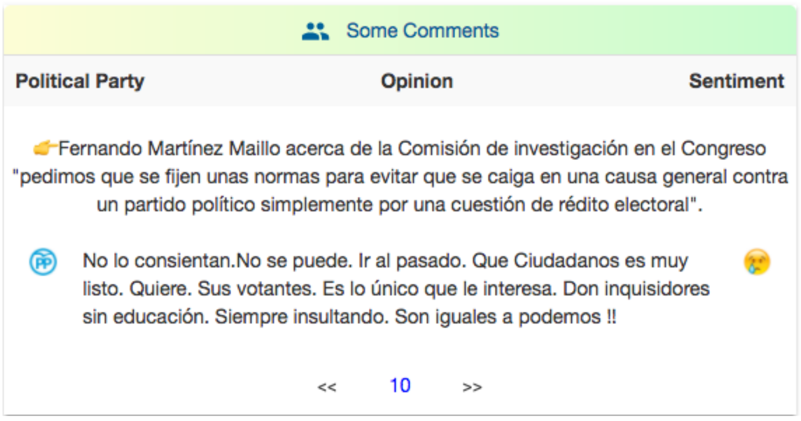
3.6.2.4 Comments-table

Showing comments that users make on Facebook is a great idea to visualize how people react to different posts. This element is in charge of illustrating different comments Facebook's users made to the posts that political parties publish.

It uses AJAX requests to obtain comments and all the information related to them, such as which political party is involved or the comment's sentiment and represents this whole information classified in a table divided into three columns:

- The first column shows the logo of the political party which comment is referred to.
- The second one contains the message of each comment.
- The last one is the result of the sentiment analysis. If the content of the comment is positive, it will display a happy face. If not, it will present a sad one.

This element is shown in Fig. 3.10.



Some Comments

Political Party	Opinion	Sentiment
Fernando Martínez Maillo acerca de la Comisión de investigación en el Congreso "pedimos que se fijen unas normas para evitar que se caiga en una causa general contra un partido político simplemente por una cuestión de rédito electoral".		
PP	No lo consientan.No se puede. Ir al pasado. Que Ciudadanos es muy listo. Quiere. Sus votantes. Es lo único que le interesa. Don inquisidores sin educación. Siempre insultando. Son iguales a podemos !!	😢

<< 10 >>

Figure 3.10: Comments-table Widget

3.6.2.5 Field-chart

This widget allows us to show how sentiments change through the period of time chosen to analyse.

This element displays a line chart showing the sentiment evolution. X-axis represents time evolution, while y-axis represents sentiment evolution.

Fig. 3.11 gives an example of this element.

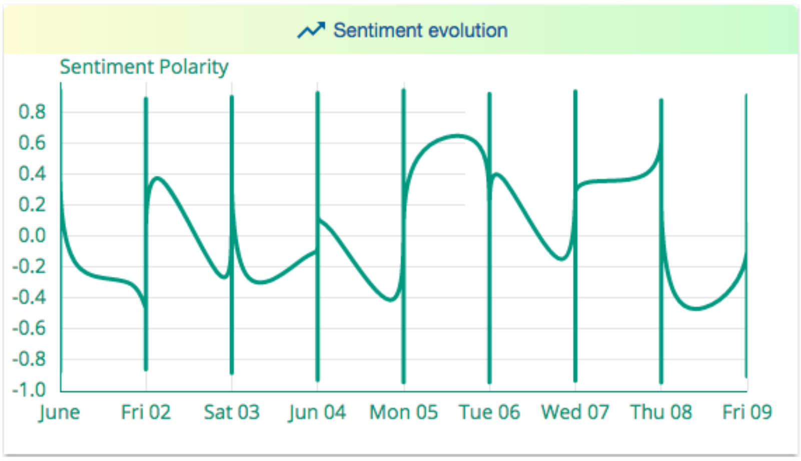


Figure 3.11: Sentiment evolution widget

CHAPTER 4

Case of study

This chapter of the work gives a detailed description of the process we followed to extract information from Facebook, analyse and store it correctly and to display it.

Examples will be given all along this chapter to make its understanding easier to the reader, so no doubt will not be answered.

The main actor in this case is the final user who access to our website looking for information about political parties, their movements and how this actions influence on population, analysing their sentiments and emotions.

4.1 Extracting data

Facebook has been the source of information in this project. We decided to use this social network because it is one of the most used platforms to communicate with 1.94 billion monthly active users[1], so it is obvious that it manages a lot of information; which benefits our project giving us a lot of data to analyse and study. Number of users and their activity are directly proportional to the quantity of information exchanged.

[1]https://newsroom.fb.com/company-info/

Facebook manages a lot of information about different topics. Celebrities, public companies or organizations have presence on this social network. Within these multiple options, we agreed to extract data from political domain because it is controversial and we can find out a lot of different opinions, which will give variety to our project. In this case, we chose "Partido Popular", "Partido Socialista Obrero Español", "Podemos" and "Ciudadanos" due to their importance and position on Spanish political panorama; they are the four most important political parties in Spain nowadays.

Not only these political parties are continuously posting news, videos, photos and comments on their public pages, but also population make comments and reacts to these publications; so the quantity of information exchanged is huge. We decided to extract information between the first and ninth of June of 2017.

The data we decided to extract from posts published by political parties is composed of:

- Publication date of each post.
- The message posted by the political party.
- Total of people most important reactions to every post, and how many there are of each type.
- Number of comments made by users of Facebook.
- Content of those comments.

In the period of time exposed previously we obtained 204 posts. Each post have its own comments, obtaining 3.962 different opinions from people to those publications (2.029 from Podemos, 840 from Partido Popular, 591 from Ciudadanos and 502 from Partido Socialista Obrero Español). In addition, we extracted the reactions to those posts classified by political party (as it is shown on the dashboard). Each political party published:

- *Partido Popular* : 52 posts, obtaining the following reactions: 12.606 likes, 966 loves, 339 hahas and 481 angries.
- *Partido Socialista Obrero Español* : 34 posts, obtaining the following reactions: 10.180 likes, 821 loves, 127 hahas and 156 angries.
- *Podemos* : 87 posts, obtaining the following reactions: 89.737 likes, 6.752 loves, 4.299 hahas and 63.190 angries.

- *Ciudadanos* : 31 posts, obtaining the following reactions: 16.059 likes, 1.117 loves, 1.819 hahas and 251 angries.

4.2 Analysing data

The second step of this project was to analyse all the data once it was extracted as described before. The tool used to make this analysis is, as it is mentioned all along this document, Senpy and sentiText.

We made a sentiment analysis obtaining 2.937 comments with a positive sentiment, while 1.025 comments were negative.

- *Partido Popular*: 648 positive comments and 192 negative.
- *Partido Socialista Obrero Español*: 382 positive comments and 120 negative.
- *Podemos*: 1489 positive comments and 540 negative.
- *Ciudadanos*: 418 positive comments and 173 negative.

4.3 Indexing data

Once data is analysed, it is needed to store and index it with ElasticSearch's help. This task is managed by the pipeline as it is mentioned in Chapter 3.2.

Our data is organised in two indexes, each one containing different information:

- First one is called *"politicos"* and it contains every post political party made, reactions to these posts, their messages and the comments' IDs made to those posts.
- The second index is named *"comentarios"* and it includes the comments made in every post.

4.4 Displaying data

Finally, indexed data requires to be displayed. In order to do so, we used a dashboard including all data and presenting it helped by interactive classified into different topics.

Every component being part of this dashboard will be explained when it appears on the different screen-shots.

Firstly, users can find out the name of the dashboard and its logo. The dashboard analyses information from political domain using Information and Communication Technologies (*TIC*); so we decided poliTICS to be our dashboard's name. The logo shows Facebook as our data source and the colour of political parties we are going to analyse.

We decided to divide this dashboard into three sections, explained and showed next.

In the first part, the number of posts published per political party is shown. Here, each political party is identified by its name and logo; which helps user to recognise it clearly and easily. The second section of this dashboard presents a comparison of the most popular Facebook reactions (like, love, haha and angry) people make to the posts analyzed. It is composed of four column diagrams, colouring columns depending on the political party that it represents.

Fig. 4.1 shows both parts explained before.

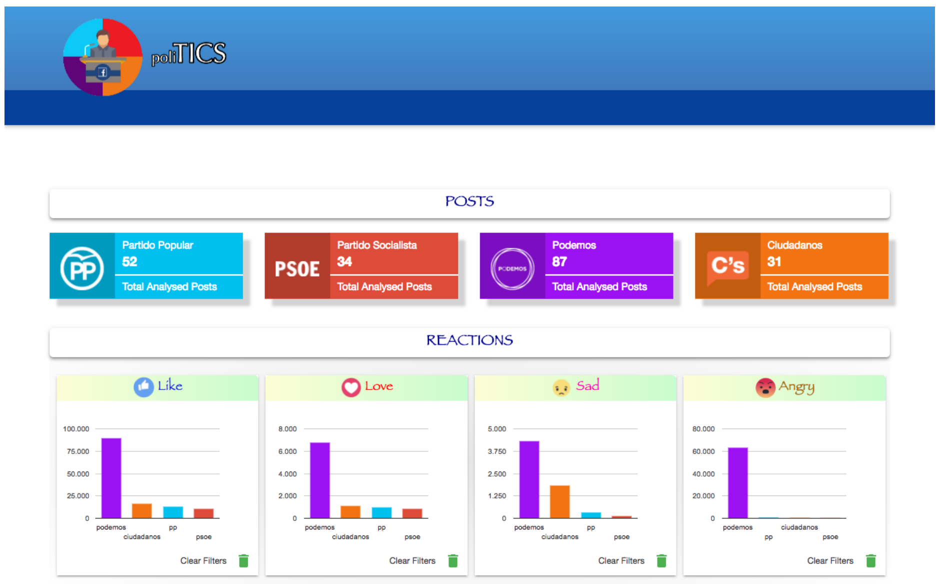


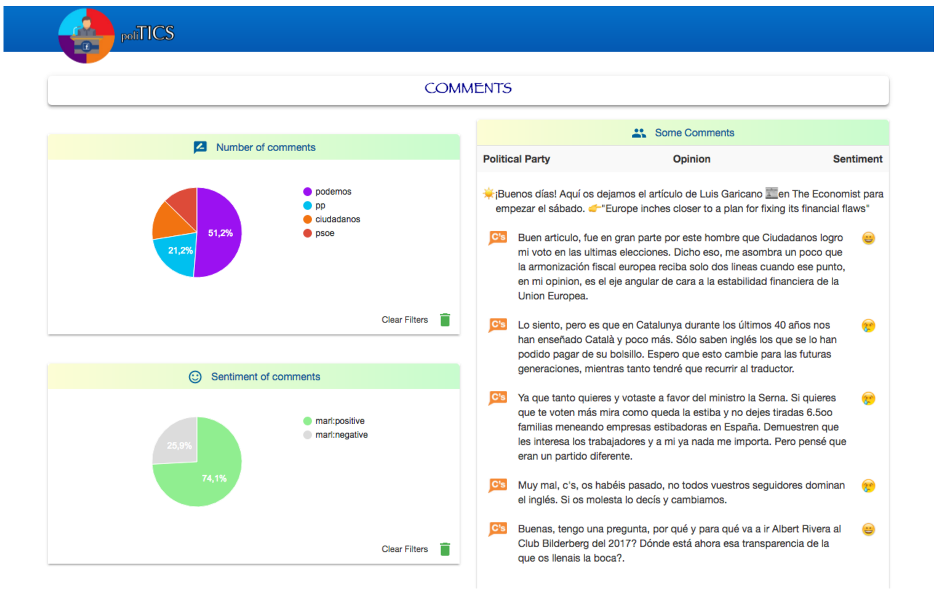
Figure 4.1: First and second dashboard's section

The last part of this dashboard is focused on the analysis of Facebook users' comments in different ways:

- The first one compares the number of comments on posts per political party. This comparison is illustrated on a pie diagram, colouring slices with the political party's colour; in order to make it more visual to users.

- Another pie diagram located below the previous one divides the number of comments depending on the sentiment reflected on them. It is also possible to filter them by political party.
- Then, a sentiment evolution graph is shown. It points out the increases and decreases of sentiment polarity comments have during the period of time analysed.
- The last part of this section show some comments including the political party they are associated and pointing out the sentiment reflected on the content of the message.

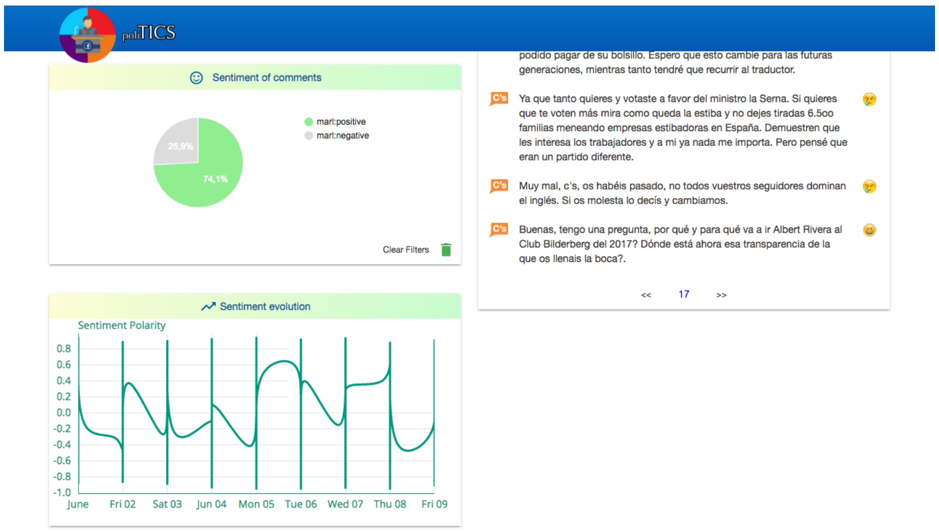
This last part is illustrated in Fig. 4.2 and Fig. 4.3.



☀️¡Buenos días! Aquí os dejamos el artículo de Luis Garicano 📰en The Economist para empezar el sábado. 👉"Europe inches closer to a plan for fixing its financial flaws"

Political Party	Opinion	Sentiment
C's	Buen articulo, fue en gran parte por este hombre que Ciudadanos logro mi voto en las ultimas elecciones. Dicho eso, me asombra un poco que la armonización fiscal europea reciba solo dos lineas cuando ese punto, en mi opinion, es el eje angular de cara a la estabilidad financiera de la Union Europea.	😀
C's	Lo siento, pero es que en Catalunya durante los últimos 40 años nos han enseñado Català y poco más. Sólo saben inglés los que se lo han podido pagar de su bolsillo. Espero que esto cambie para las futuras generaciones, mientras tanto tendré que recurrir al traductor.	😢
C's	Ya que tanto quieres y votaste a favor del ministro la Serna. Si quieres que te voten más mira como queda la estiba y no dejes tiradas 6.5oo familias meneando empresas estibadoras en España. Demuestren que les interesa los trabajadores y a mi ya nada me importa. Pero pensé que eran un partido diferente.	😢
C's	Muy mal, c's, os habéis pasado, no todos vuestros seguidores dominan el inglés. Si os molesta lo decís y cambiamos.	😢
C's	Buenas, tengo una pregunta, por qué y para qué va a ir Albert Rivera al Club Bilderberg del 2017? Dónde está ahora esa transparencia de la que os llenais la boca?.	😀

Figure 4.2: Third dashboard's section



podido pagar de su bolsillo. Espero que esto cambie para las futuras generaciones, mientras tanto tendré que recurrir al traductor.

Ya que tanto quieres y votaste a favor del ministro la Serna. Si quieres que te voten más mira como queda la estiba y no dejes tiradas 6.5oo familias meneando empresas estibadoras en España. Demuestren que les interesa los trabajadores y a mi ya nada me importa. Pero pensé que eran un partido diferente.

Muy mal, c's, os habéis pasado, no todos vuestros seguidores dominan el inglés. Si os molesta lo decís y cambiamos.

Buenas, tengo una pregunta, por qué y para qué va a ir Albert Rivera al Club Bilderberg del 2017? Dónde está ahora esa transparencia de la que os llenais la boca?.

Figure 4.3: Third dashboard's section (2)

4.5 Conclusions

In this chapter, we have presented the different steps we have followed in order to extract, analyse, store and display data from Facebook. We also have explained the different widgets used and showed the final view we reached in our dashboard.

Looking to the statistics given and analysing them, we could reach the following conclusions:

- The political party which Facebook personal activity is higher is *Podemos*, with 87 posts in 5 days. *Ciudadanos* is the political party with less posts.
- Due to *Podemos* is the political party with more posts published, it receives the higher number of reactions.
- Although *Partido Popular* has a very intensive activity with 52 posts in five days; their users do not interact with it the same way. Number of reactions make it decreases to the third position.
- The total amount of comments is directly proportional to number of posts per political party. Only *Partido Socialista Obrero Español* don't follow that proportion, with less comments per post than the other political parties.
- Facebook users usually positive comments to the posts the politicar parties publish.

This may be because most of the people visiting their public pages are their followers and voters. Otherwise, there are negative comments too.

- Although *Podemos* is the political party with most negative reactions, it has 1489 positive comments (a higher quantity than the negative ones); the largest number of the four political parties.

- Even though the period of time it is not a controversial and polemic one, we can assure that Facebook has a lot of activity in political domain.

CHAPTER 5

Conclusions and Future work

This last chapter of the project sums up the conclusions achieved during the period of work. It will announce the main goals achieved and problems faced while developing it, and will study different options to develop in future work.

5.1 Conclusions

In this project we have developed a scraper system to extract data from public pages on Facebook. In addition, we have made a sentiment analysis of this information, and illustrated it on a dashboard based on Wb Components and widgets in order to make it visible, accessible and clearly shown.

This project is composed of five subsystems. The fist one is the Facebook Scraper System, which obtains every data we want to obtain from this social network (in our case, we chose data from political domain). This system is followed by an Orchestrator. It manages the rest of the tasks of this project due to it is a module for sequencing and executing tasks and dependencies. Next module is Analysis System and it is in charge of obtaining sentiments from comments users made on Facebook. The result of this system is collected by the Indexing System, which classifies and stores data analysed. Finally, the last system is the Visualization System. This one makes data visible on a dashboard, trying to do so on a striking way.

5.2 Achieved goals

This section will explain the main goals achieved during this project, comparing them to the initial expectations.

Collect data from Facebook focused in political domain First of all, we needed to obtain information from Facebook referring to political parties. In this case, it was an easy task because each political party has its own public page and Facebook provides users an API from which you can access to data stored in this social network.

Analyse and store extracted information Extracted data was unanalysed. For our purpose, we needed to analyse it with a service providing different sentiments showed on comments. Luigi, used in our project as an orchestrator, helped us with this task. As it is mentioned in this document, it provided us a pipeline between Senpy and Elasticsearch; storing analysed data in an Elasticsearch index.

Design a dashboard based on widgets This was the last goal of this project, but it was the most important too: we had to develop a dashboard. This dashboard is dashboard is bases on widgets and Polymer Web Components, which gave our project diversity and visibility. These widgets show users data in different ways, allowing them to extract conclusions and analyse them.

5.3 Problems faced

During the development of this project, some problems appeared related to different issues. We are listing them next.

- *Facebook's Privacy Policy*: Facebook has a lot of users and it is important to keep their information privately. Facebook has a very strict privacy policy which made hard to extract users data, unless they were completely public. This way, if a profile's user is configured as private, it is almost impossible to extract from Facebook information such as their comments posted on a public page or personal data such as their age, name or gender.

- *Information format*: Facebook Graph API exports data in a very disordered way; so we had to filter it and select which information we wanted, parsing the JSON file we obtained from this API.
- *Elasticsearch version*: last version of Elasticsearch included some improvements that we have to face and update our code to make sure it will work. One of the most important issues in this new version is the way it processes String type objects. To make sure it works, we had to modify every index mapping indicating as *true* the fielddata parameter of each *text* type.
- *Python versions*: we developed the Scraper System with Python version 2.7, while Luigi pipeline worked with Python 3; so we had to install both versions of Python in order to accomplish this project's goals.

5.4 Future work

In this section, we will explain some possible new features or improvements that could be done in the future to this project, making it more interesting.

Adding new political parties In this thesis we analysed the four most important political parties at the Spanish actual panorama. Hereafter, we could include more political parties and different countries to analyse. Adding more data to our project will make this analysis more interesting and complete.

Developing a new dashboard about other topic We chose to analyse data from political domain, but we think this project could be extrapolated to other topics such as music or cinema, but also to specific people we want to analyse such as politicians or singers.

Adding new widgets We included some widgets in our dashboard, but the possibilities given by Polymer and Web Components are much greater. New widgets could be added to analyse and show data in multiple and new ways.

Analysing data in important periods of time We think this project could be very interesting if we analyse certain periods of time when political panorama is more active and controversial such as elections or important movements inside political parties.

Bibliography

[1] Margaret M Bradley and Peter J Lang. Affective norms for english words (anew): Instruction manual and affective ratings. Technical report, Citeseer, 1999.

[2] Clinton Gormley and Zachary Tong. *Elasticsearch: The Definitive Guide.* " O'Reilly Media, Inc.", 2015.

[3] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014.

[4] Scott D McClurg. Social networks and political participation: The role of social interaction in explaining political participation. *Political research quarterly*, 56(4):449–464, 2003.

[5] Ryan Mitchell. *Web Scraping with Python: Collecting Data from the Modern Web.* " O'Reilly Media, Inc.", 2015.

[6] Javier Ochoa Serna. Design and implementation of a scraping system for sport news. 2017.

[7] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, 2008.

[8] Alberto Pascual Saavedra. Development of a dashboard for sentiment analysis of football in twitter based on web components and d3. js. 2016.

[9] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics, 2004.

[10] Marcel Rieger, Martin Erdmann, Benjamin Fischer, and Robert Fischer. Design and execution of make-like, distributed analyses based on spotify's pipelining package luigi. *arXiv preprint arXiv:1706.00955*, 2017.

[11] J Fernando Sánchez-Rada, Carlos A Iglesias, Ignacio Corcuera, and Óscar Araque. Senpy: A pragmatic linked sentiment analysis framework. In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*, pages 735–742. IEEE, 2016.

[12] Weiwu Zhang, Thomas J Johnson, Trent Seltzer, and Shannon L Bichard. The revolution will be networked: The influence of social networking sites on political attitudes and behavior. *Social Science Computer Review*, 28(1):75–92, 2010.

[13] Ying Zhu. Introducing google chart tools and google maps api in data visualization courses. *IEEE computer graphics and applications*, 32(6):6–9, 2012.