UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA DE
REDES Y SERVICIOS TELEMÁTICOS

TRABAJO FIN DE MÁSTER

DESIGN AND DEVELOPMENT OF A PLATFORM TO TRACK SUSTAINABLE MOBILITY SOCIAL TRENDS

MARTÍN GONZÁLEZ CALVO
JUNIO 2021

TRABAJO DE FIN DE MÁSTER

Título:	Diseño y desarrollo de una plataforma de seguimiento de tendencias sociales sobre movilidad sostenible
Título (inglés):	Design and Development of a Platform to Track Sustainable Mobility Social Trends
Autor:	Martín González Calvo
Tutor:	Óscar Araque
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

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

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO FIN DE MÁSTER

DESIGN AND DEVELOPMENT OF A PLATFORM TO TRACK SUSTAINABLE MOBILITY SOCIAL TRENDS

Martín González Calvo

Junio 2021

Resumen

Con el paso del tiempo, el uso de herramientas de aprendizaje automático se ha convertido en un gran método para obtener información práctica de diferentes fuentes de datos. La acción de estudiar las interacciones de las personas a través de las redes sociales y el Procesamiento del Lenguaje Natural es un gran ejemplo de cómo se pueden extraer y analizar las opiniones sobre determinados temas.

La movilidad sostenible es una preocupación creciente en nuestra sociedad moderna y es importante investigar cómo reacciona la gente a las nuevas políticas relacionadas con este tema. Para ello, se ha desarrollado un cuadro de mando para seguir tendencias sociales sobre movilidad sostenible. Este proyecto está alineado con el objetivo de desarrollo sostenible número 11 de las Naciones Unidas, que promueve ciudades y comunidades sostenibles.

Para ello, se analizan los tweets para categorizarlos, detectar entidades con nombre y proporcionar el sentimiento de los mensajes de Twitter. El pipeline que procesa los datos se ha creado utilizando tecnologías de Big Data: Luigi para orquestar los diferentes procesos, el ecosistema científico Python para analizar los datos y evaluar los modelos, y ElasticSearch para almacenar los datos generados.

Para analizar la polaridad de los mensajes de Twitter (positivo, negativo, neutro), se ha creado, entrenado y evaluado exhaustivamente un modelo de análisis de sentimientos construido con deep learning. Una vez que los datos de entrada se enriquecen semánticamente, un panel dinámico creado con Polymer, proporciona una visualización completa de los datos recuperados junto con diferentes gráficos para agregar la información analizada.

Palabras clave: Procesado del Lenguaje Natural, Twitter, Monitorización Social, Movilidad Sostenible

Abstract

Over time, the use of Machine Learning tools has become a great method to build actionable insights. The action of studying the people interactions through social networks and Natural Language Processing is a great example on how opinions about certain topics can be retrieved and analyzed.

Sustainable mobility is a growing concern in our modern society and it's important to research how people react to new policies regarding this topic. In order to so, a dashboard to track sustainable mobility social trends has been developed. This project is aligned with the United Nations' sustainable development goal number 11, which promotes sustainable cities and communities.

To achieve this, tweets are analyzed to be categorized, detect named entities and provide the sentiment of Twitter messages. The pipeline that processes the data has been created using Big Data technologies: Luigi to orchestrate the different processes, the Python scientific ecosystem to analyze data and evaluate models, and ElasticSearch for storing the generated data.

To analyze the polarity of Twitter messages (positive, negative, neutral), a deep learning sentiment analysis model has been created, trained, and thoroughly evaluated. Once the input data is semantically enriched, a dynamic dashboard, created using Polymer, provides a comprehensive visualization of the retrieved data alongside different charts to aggregate the analyzed information.

Keywords: Natural Language Processing, Twitter, Social Monitoring, Sustainable Mobility

Agradecimientos

A mis padres por haberme ayudado a llegar aquí.
A Óscar por ayudarme y guiarme durante el desarrollo de este trabajo.
A Guille por estar siempre ahí para ayudarme.
A mi compañera de vida Clara, que aunque este año no haya estado presente en cuerpo, sí lo ha estado en alma.
A todos mis amigos que me han acompañado durante este viaje.

A todos, gracias.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Context

Over time, the use of Machine Learning tools has become an incredible and powerful method to recognize patterns and build actionable insights on them [14]. These tools cover a wide range of approaches. Identifying tendencies in the overall population by looking into people's interactions in social networks [15] [16] is a great example. This is specially true when taking a look in those networks that are open and enable the user to interact with any posts by expressing their opinion about different topics, like Twitter [17].

When talking about sustainable mobility, over the last years a growing concern about climate has been detected amongst the overall population, as denoted by the last European Investment Bank climate survey [18]. In this new environmentally conscious society, politicians, governments and independent organizations promote a more sustainable way of life through legislation and recommendations [19].

It is very important to track how this is done and what is the general population approach towards these measures, including how legislators and agencies promote and defend this measures. A great way to do so is through dashboards aggregating these posts alongside different charts [20].

These posts are acquired through social media monitoring [21], which is the systematic observation and analysis of social media networks and social communities. This is done with the final goal of attending to what is being said about certain subjects on the web.

Social media monitoring provides advantages like the accessibility for researching people's opinions about certain topics, which far surpasses the more traditional survey approaches. In comparison, it is normally more precise and economic, offering event detection capabilities. In the sustainable mobility field, the activity of monitoring social media provides insights about individuals' reactions to new legislation, alongside feedback about companies that offer services in this field, like ride-hailing companies or scooter and bicycle rental companies.

The **Cabify-UPM Chair for Sustainable Mobility**[1] is an association between the ride-hailing company Cabify and the Technical University of Madrid. Its goal is to stimulate the development of alternative and ecological transport models through their study and observation. This is done by promoting intelligent technologies as a precursor of leadership, leading to the development of sustainable transportation models through the study and analysis of large volumes of data.

To help pursue this task, the Cabify Observatory for Sustainable Mobility was created. This platform consists on a news tracker, where relevant information related to sustainable mobility in Spanish is presented alongside useful metrics. The objective of this work is to enlarge the observatory adding posts from the social network Twitter as a data source.

1.2 Project goals

The main purpose of this project consists on the design and development of a platform to track sustainable mobility social trends. The social network selected for the purpose of this project is Twitter. The platform will be integrated into the Cabify Observatory for Sustainable Mobility, which already offers a sustainable mobility news tracker.

The development of this platform has been carried out by creating a Natural Language Processing (NLP) model to study the sentiment of the collected tweets. The tweets will go through two other models already created to identify named entities and the category to which they belong. After this, the tweets will be enriched following Linked Data principles and finally stored. Finally, all the processed data will be presented in a dashboard in a timeline-like style alongside different charts to group the metrics generated.

[1] `https://catedra-cabify.gsi.upm.es/`

1.3 Structure of this document

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

- **Chapter 1: Introduction**: It is the presentation of the project. Its context and the main goals are described.
- **Chapter 2: Enabling Technologies**: Describes the different technologies used to carry out this project.
- **Chapter 3: Architecture**: Provides a description of the system as a whole.
- **Chapter 4: Sentiment Analysis Model**: Discusses the creation of the Natural Language Processing model used in the tool.
- **Chapter 5: Case Study**: Shows a demonstration of the tool working.
- **Chapter 6: Conclusions**: The conclusions, achieved goals and future works are discussed.

CHAPTER 2

Enabling Technologies

In this chapter, the technologies used throughout the development of this project will be discussed.

2.1 Natural Language Processing

Natural Language Processing [22] is the area within the Artificial Intelligence field which investigates how can computers asimilate and process text and speech to achieve certain goals. This is done by researching how humans use language and interact with it so it can be translated into tasks for machines to do.

NLP mixes a wide range of disciplines in order to achieve its goals [22], going from computer and information sciences to linguistics, mathematics, artificial intelligence and psychology. NLP studies fields such as machine translation, natural language text processing & summarizing, cross-language information retrieval, speech recognition, language modeling, part-of-speech tagging, named entity recognition, sentiment analysis and paraphrase detection. For this project, the most relevant are named entity recognition and sentiment analysis. For the first one, an already created model will be used. For the second one, a sentiment analysis model will be created.

NLP techniques can be divided into [23]:

- **Stochastic Machine Learning Approaches:** Stochastic models [24] represent processes in the real world based on observed data. These models describe probabilistic relationships between the different variables and conclusions can be acquired from these relationships. Some examples include Naive Bayes [25] or Logarithmic Regression [26].
- **Symbolic Machine Learning Approaches:** This field can be divided into:
  - Decision Trees [27]: Extraction of rules from training data that lead to hierarchical sequential structures that recursively partition the data. In NLP it can be applied to problems such as speech recognition, part-of-speech tagging, parsing & text categorization, among many others.
  - Decision Lists [28]: Creation of lists with sequential rules where data is checked against in order to find matching patterns to classify it.
  - Transformation-Based Error-Driven Learning [29]: Error based procedure that creates a set of rules. This is done by iterating and at every step, fix the present errors. That way, specific rules are acquired.
  - Linear Separators [30]: Algorithms for binary classification problems, where a linear combination of the different inputs is calculated (through a series of weights for each input) and returns 0 or 1 depending on the threshold. When a training example is predicted wrong, the weights are changed to achieve a better performance.
  - Instance-based Learning [31]: Set of techniques that keep in memory all the training data and compare new inputs against it.
- **Sub-symbolic Machine Learning Approaches:**
  - Neural Networks [32]: Nature-inspired pattern where learning is achieved through a set of interconnected elements that process data, called neurons. Neurons work as a whole to solve a problem. The set of neurons make up what is known as a neural network, which is trained for a specific task like pattern recognition or data classification. Neural networks are made up of three layers: Input, hidden and output.
  - Genetic Algorithms [33]: Optimization methods based on natural selection. This is done by creating an initial solution that has "off-springs" (similar programs) with different "mutations" (slight modifications). The fittest of all is selected and the process is repeated until an optimal solution is found.

2.1.1 NLP in this project

The NLP technologies used in this project will be explained more thoroughly.

Bag of Words

The bag-of-words model [34] consists on a method to extract features from text, representing it as the set of the words that it contains. This model only takes into account the number of times words are repeated in the text (multiplicity), meaning that information such as the order, structure or grammatical features are ignored. Once all the words that appear among the texts to study are collected, vectors where each element represents the presence or absence of each word are created.

The main idea behind bag-of-words is that texts are similar if they contain similar content, providing a great way to classify documents. The complexity of bag-of-words can vary according to the size of the vocabulary. The more known words, the larger the vector representation of the text will be. To tackle the high dimensionality problem, some solutions like the creation of a vocabulary of grouped words, known as n-grams [35] provide a dimension reduction and a boost in performance.

Word Embeddings

Word embeddings [36] consist on the representation of words as continuous vectors which encode their meaning in a way that the similar in meaning the words are, the closer in the vector space they are. Word embeddings can be obtained using a set of language modeling and feature learning techniques where words or phrases from the vocabulary are mapped to vectors of real numbers. In Figure 2.1 a visual representation of word embeddings for gender and verb tense is provided.

To generate the different mappings, a variety of methods like probabilistic models or neural networks are used. Furthermore, in recent years embeddings that take into account the context of the words to disambiguate polysemes have been created. An example is BERT [37], which stands for Bidirectional Encoder Representations from Transformers.

It has been developed by Google and provides great results. In fact, the large majority of English querys processed by Google Search go through BERT [38]. Although it provides great state-of-the-art results it is not entirely comprehended why [39]. This is done through Transformer neural network architectures, which will be explained below.

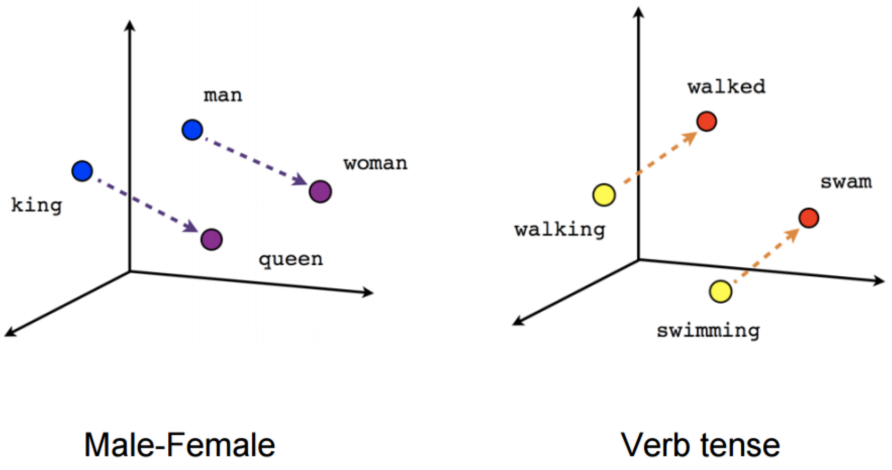


Figure 2.1: Example of word embeddings for gender (left) and verb tense (right) [1]

Convolutional Neural Networks

To understand what a Convolutional Neural Network (CNN) is, the concept of convolution must be explained.

In the field of neural networks, a convolution is the process of applying a filter to an input signal. This could be understood as applying a sliding window function (filter) to a matrix. This filter is also a matrix. In Figure 2.2, an example of the last step of a 3x3 convolution in a 5x5 matrix is provided. In this case, the filter multiplies by the values in the current window (marked as yellow) by the values marked in red. Finally, results are added to create an element of the convolved feature matrix. To get this result, the filter has slid through the whole matrix generating the different elements. The results of the convolution process may vary according to the filter and the operations performed.

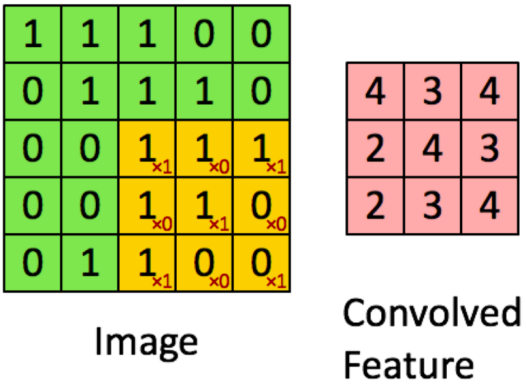


Figure 2.2: Example of a 3x3 convolution in a 5x5 matrix. [2]

Convolutional Neural Networks [40] consist on multiple layers of convolutions followed by pooling operations. Pooling consists on the reduction of variance and computation complexity using a window function like in convolution. The three most common types are:

- Max Pooling: Select the maximum value of the window
- Min Pooling: Select the minimum value of the window
- Average Pooling: Calculate the average of the window

In Figure 2.3 visual representations of max and average pooling are provided.

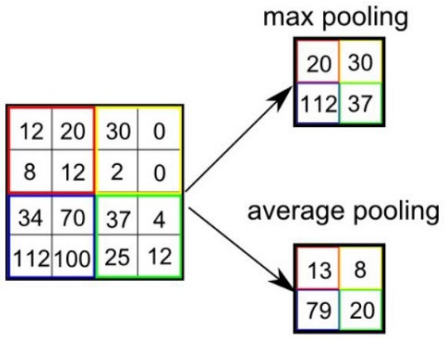


Figure 2.3: Example of max pooling and average pooling [2]

Once the operations are finished, the output layer acts as a classifier. In NLP, words are vectorized using word embeddings or bag-of-words models and then go through the CNN to obtain results. In Figure 2.4, an example of a CNN applied to an NLP problem is provided

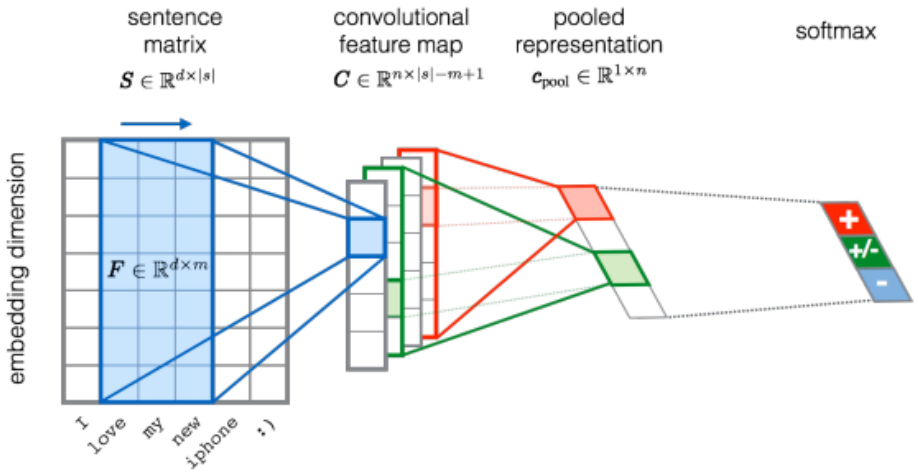


Figure 2.4: Example of a CNN in NLP [3]

Transformers

Transformers [41] are neural network models that use the attention mechanism. This means that it enhances the most important parts of the introduced data and ignores the rest. In Figure 2.5 a detailed overview of this process is provided.

The transformer architecture converts one sequence into another using to main parts:

- **Encoder:** Shown at the left side of the figure, finds correlations among words, normalizes them and repeats the process again, to pass the output to the decoder.
- **Decoder:** Shown at the right side of the figure, does the same tasks as the encoder but adding the encoder output as an input to calculate the output probabilities.

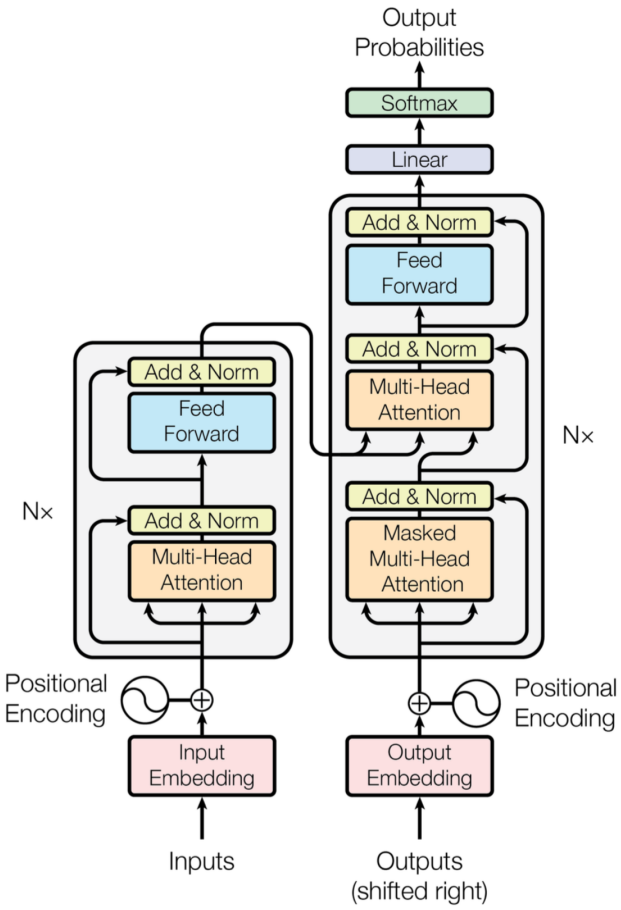
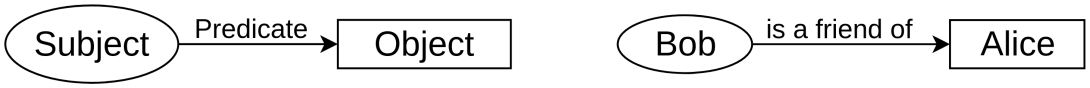


Figure 2.5: Transformer Architecture [4]

2.1.2 Resource Description Framework

The Resource Description Framework (RDF) [42] consists on a series of specifications by the W3C that describe a data model and were initially used as a metadata model. Over the course of time, it has become the main method to model information on the web. The main structure in RDF is the triple, formed by subject, predicate and object, it can be understood as a graph, as seen on Figure 2.6.



(a) RDF Triple Scheme

(b) RDF Triple Example

Figure 2.6: RDF Triples

Subjects can be objects on other triples and vice-versa, thus forming a network of interconnected entities known as the semantic web. Since the vocabulary that the RDF provides might come in as short for some tasks, support for creation of new vocabularies is implemented thorugh RDF Schema [43]. Also, the Web Ontology Language (OWL) [44] has been created in order to provide additional semantics that RDF Schema doesn't provide. In Figure 2.7 an example of the use of thse vocabularies, called ontologies is provided. This data can be serialized into different formats like Extensible Markup Language (XML), JavaScript Object Notation (JSON) and Turtle.

2.1.3 Linked Data

Linked Data [45] consists on a combination of best practices for posting and interconnecting structured data on the internet. The assimilation of these principles has lead to the creation of the web of data (data is linked as in web pages are linked on the internet). Tim Berners-Lee, the creator of the world wide web, proposed these principles for Linked Data [46]:

1. Use Uniform Resource Identifiers (URIs) as names for things.
2. Use Hypertext Transfer Protocol (HTTP) URIs so people can look up those names.
3. When someone looks up a URI, provide useful RDF information.
4. Include RDF statements that link to other URIs so related things are discovered.

2.1.4 Ontologies

Ontologies [47] consist on vocabularies used to enhance the RDF Schema vocabulary capabilities. For the purpose of this project the following ontologies have been used:

- **Schema.org** (schema) [48]: Mantained by Google, Microsoft, Yahoo, Yandex and a public community. Provides a variety of topics including people, places, events, etc.
- **Semantically-Interlinked Online Communities** (sioc) [49]: Connects discussion sites such as social networks or forums. It includes information contained in the sites.
- **Dublin Core** (dc) [50]: Created to describe resources. It is maintained by the Dublin Core Metadata Initiative (DCMI).
- **Friend of a Friend** (foaf) [51]: Describes people and their relationships with other people & different entities.
- **Provenance** (prov) [52]: Offers a set of classes, properties, and restrictions used to denote the source from where the data was created.
- **Simple Knowledge Organization System** (skos) [53]: Represents taxonomies, classification schemes and other kinds of vocabularies that are structured.
- **NLP Interchange Format** (nif) [54]: OWL-based format with the goal of exchanging data between NLP tools, language resources and annotations.
- **Marl** (marl) [55]: Used for subjective opinions expressed in text. Describes sentiment expressions and is used to collect results of sentiment analysis.
- **Senpy** (senpy): Ontology not published yet, used internally.

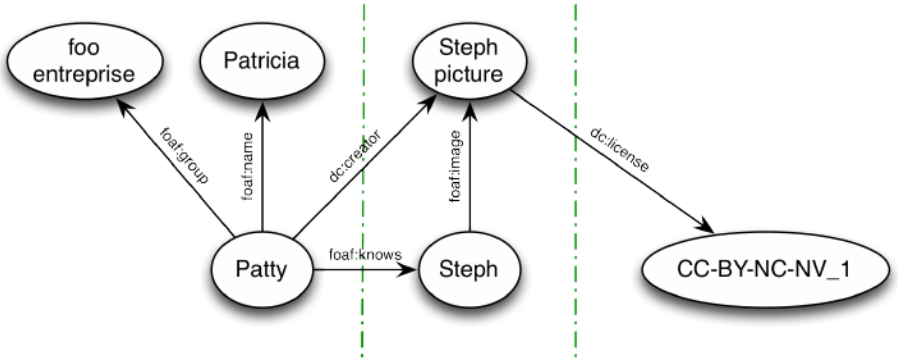


Figure 2.7: A simple example of Linked Data [5]

2.2 Python Scientific Ecosystem

Python [56] consists on a interpreted high-level, general-purpose, cross-platform programming language published under an Open Source Initiative (OSI) open source license. It is a language known for its high-level data structures and easy approach to object-oriented programming.

It is is dynamically-typed language, this means that the types of a variables is checked during run-time as opposed to statically-typed languages, that check variable types at compile-time, like in Java or C. It is also garbage-collected, meaning that the memory that is allocated by the program, but no longer gets referenced is re-used by the program. Furthermore, this programming language interpreter can be extended with data types and functions implemented in C or languages that are callable from C.

It has been designed with a philosophy that focuses on the readability of the code. This is achieved thanks to significant indentation, forcing programmers to make code that is clear and logical.

2.2.1 NumPy

NumPy [57] is the fundamental package for scientific computing with Python. Among all the features it provides, the following can be highlighted [58]:

- ndarray, NumPy key feature, it consists on a multidimensional array object that is very fast and efficient.
- Built-in functions with the capability of performing diverse operations between arrays but that are also able to operate element by element.
- Tools to perform read and write operations of datasets into disk.
- Supports linear algebra operations, Fourier transform and random number generation.
- Integration of C, C++, and Fortran code into Python.

The main application of NumPy is its potential to process arrays. However, it is also used because of its application as a generic data container and capability to define any type of data.

2.2.2 Pandas

Pandas [58] consists on an open source Python library programmed for data modeling and analysis.

It provides a wide range of functions designed to work with structured data. It also provides two main data structures:

- **Pandas Series**: It consists on a one-dimensional object, where each element of the object has its own index and value. It could be understood as a simple array.

- **Pandas DataFrame**: The main Pandas data structure. It is formed as a two-dimensional object that can be seen as a spreadsheet, or a database table, where each element is indexed.

An example for both a Pandas Series and a Pandas DataFrame is provided on Figure 2.8.

Pandas combines NumPy capabilities alongside other data manipulation features from relational databases and spreadsheets.

Data can be imported in different formats like Structured Query Language (SQL), Comma-Separated Values (CSV), JSON, or even text files to a pandas data structure.

That way, any kind of structured data from any source can be reshaped, sliced, diced, aggregated, have subsets of it selected and many more operations in a fast, easy and expressive way.

```
>>> s1
AAPL   0.044
IBM    0.050
SAP    0.101
GOOG   0.113
C      0.138
SCGLY  0.037
BAR    0.200
DB     0.281
VW     0.040
```

(a) Two Pandas Series

```
>>> s2
AAPL   0.025
BAR    0.158
C      0.028
DB     0.087
F      0.004
GOOG   0.154
IBM    0.034
```

```
>>> df
   A    B      C        D
0  foo  one   -1.834    1.903
1  bar  one    1.772   -0.7472
2  foo  two   -0.67    -0.309
3  bar  three  0.04931  0.3939
4  foo  two   -0.5215   1.861
5  bar  two   -3.202    0.9365
6  foo  one    0.7927   1.256
7  foo  three  0.1461  -2.655
```

(b) A Pandas DataFrame

Figure 2.8: Pandas Data Structures [6]

2.2.3 Scikit-Learn

Scikit-learn [59] is a free Machine Learning (ML) library that exposes a wide variety of ML algorithms. This is done by offering task-oriented and consistent interface. Thanks to it, users are able to easily compare different methods and approaches for a given problem.

In order to work properly, scikit-learn uses the scientific Python ecosystem. Because of this, implementing it out of the conventional data analysis spectrum is very easy. Furthermore, algorithms written in a high-level language can be used as parts of a bigger processes that challenge specific use cases, for example, medical imaging. Among the most important tools, scikit-learn includes:

- **Pre-processing**: Different utilities to change raw feature vectors into a representation that is more suitable for the downstream estimators. Among the most important ones we find:
  - Feature Extraction: As its name indicates, extracts features from objects such as images or text.
  - Feature Selection: To identify which attributes are meaningful to create models later.
  - Standardization & Normalization: To make data distribution similar to a Gaussian with zero mean and unit variance, since it's a common requirement for many estimators in ML.
- **Dimensionality Reduction**: To eliminate the number of random attributes in the data.
- **Clustering**: To group unlabeled data.
- **Cross Validation**: To estimate the accuracy of supervised models on unseen models.
- **Model selection and evaluation**: To get the best possible metrics between all the models and parameters available.

For this project, scikit-learn has been used in order to select and evaluate the different NLP models created.

2.2.4 Matplotlib

Matplotlib [60] is a Python plotting library. In its beginnings it was designed to imitate MATLAB functionalities. It is built with an object-oriented Application Programming Interface (API) that enables plots to be embedded. Among all the plotting functionalities the most important are line plots, scatter plots, histograms, polar plots, contour plots, image plots and 3D plots, some examples can be seen on Figure 2.9.

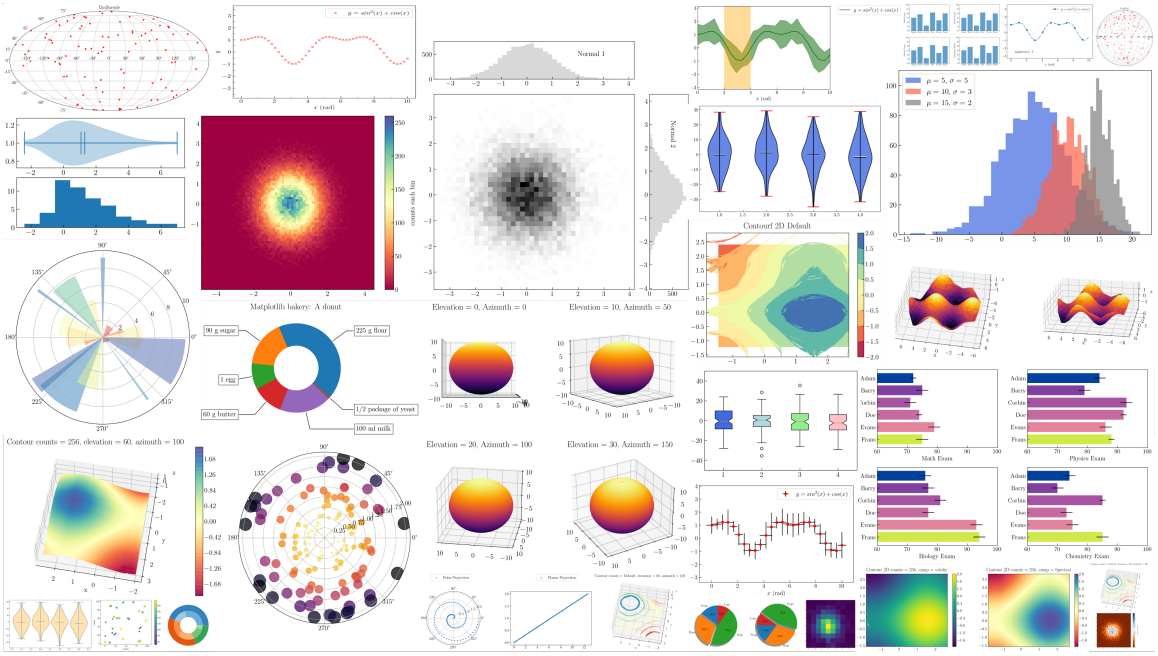


Figure 2.9: Example of matplotlib figures [7]

2.2.5 Seaborn

Seaborn [61] is a Python data visualization library that offers an interface to use matplotlib easily and permits quick data exploration. Among its functionalities, can be highlighted:

- Dataset-oriented API to examine relationships between multiple variables.
- Matplotlib figure styling control with several built-in themes.
- Support to use categorical features to show observations or make aggregations.
- Abstractions for structuring multi-plot grids to build complex visualizations.
- Plotting univariate or bivariate distributions to compare them between data subsets.
- Automated estimation and plotting of linear regression models.

2.2.6 Beautiful Soup

Beautiful Soup [62] consists on a Python library that enables the user to extract data from HyperText Markup Language (HTML) and XML files. The library supports different parsers, like the one included in Python's standard library or third party ones like lxml [63].

2.2.7 Jupyter Notebooks

Jupyter Notebook documents [64] are files that contain both computer code like Python or other programming languages and rich text elements like paragraphs, equations, figures and links. Notebook documents have the extension .ipynb (which stands for IPython Notebooks).

These documents are both human-readable documents containing the analysis, description and results presented with multiple components like figures and tables as well as executable documents which can be run to perform different tasks.

Jupyter Notebooks are run in the Jupyter Notebook App, which is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access or can be installed on a remote server and accessed through the Internet.

2.2.8 Anaconda

Anaconda [65] is a cross-platform data science framework designed for Linux, Windows and macOS operative systems. It enables the user to quickly download over 7,500 data science libraries. It also has the capability to create virtual environments to quickly operate with all the packages that the framework offers.

Anaconda comes in different versions. Except for the individual version, all have commercial purposes and require to pay. For this project, the Individual Edition which is open-source, has been used.

2.2.9 spaCy

spaCy [66] is an Industrial-Strength Natural Language Processing library developed in Python. The sentiment model in this project will be developed with the latest version of this tool. Among the most important features, the following can be highlighted [67]:

- Support for 64 languages.
- 55 trained pipelines for 17 languages.
- Linguistically-motivated tokenization.
- Components for Named Entity Recognition (NER), part-of-speech tagging, dependency parsing, sentence segmentation, text classification, lemmatization, morphological analysis, entity linking and more.
- Built in visualizers for syntax (as seen on Figure 2.10) and NER.
- Pre-trained word vectors.
- State-of-the-art speed.
- Production-ready training system.
- Easily extensible with custom components and attributes.
- Support for custom models in PyTorch, TensorFlow and other frameworks.
- Easy model packaging, deployment and workflow management.
- Robust, rigorously evaluated accuracy.

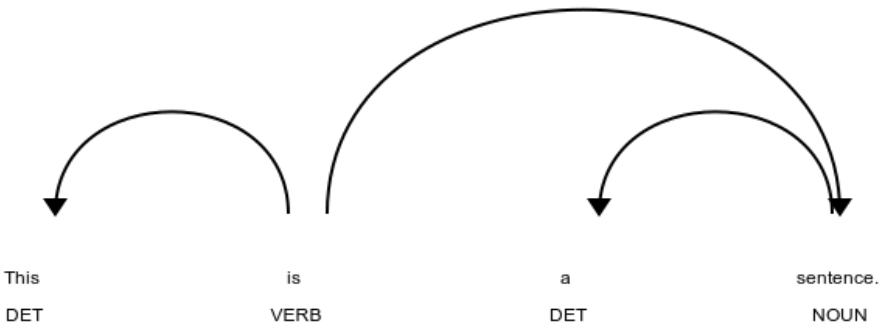


Figure 2.10: spaCy syntax visualizer [8]

2.3 Twitter API

Twitter [68] is a micro-blogging social network with millions of active users. It lets users post text messages that can be accompanied with media such as pictures, videos, voice notes or live streams, as shown on Figure 2.11. These messages have a maximum of 280 characters and are called tweets.

The social platform offers an API [69] that enables programmatic access to Twitter, letting developers:

- Retrieve data from tweets, users, direct messages, lists, trends, media, and places through advanced queries.
- Analyze past conversations and measure tweet performance.
- Listen for important events and stream tweets in real time.

Some of the features provided by the Twitter API require premium access. However, for the development of the project only the free function to retrieve tweets is needed.

Figure 2.11: Example of a tweet published by the NASA.

2.4 Senpy

Senpy [70] consists on a open source framework developed in Python by Universidad Politécnica de Madrid's (UPM) Grupo de Sistemas Inteligentes (GSI) that allows users to develop, evaluate, publish and consume web services for sentiment and emotion analysis in text. Its main advantage is that allows users to use services (plug-ins) from different providers from the same platform.

This is done by combining an API aligned with the NLP Interchange Format (NIF) service specification, the use of semantic formats and a series of established ontologies. As seen on Figure 2.12 Senpy's architecture works as follows:

1. The service is requested through a query with the data to analyze, indicating the plug-in(s) desired to use.
2. Parameters are validated and plug-ins are executed.
3. Data is serialized, formatted and validated
4. The output is returned following Linked Data principles alongside a graph-like visualization and the text in the desired format (JSON, Turtle, RDF, etc.).

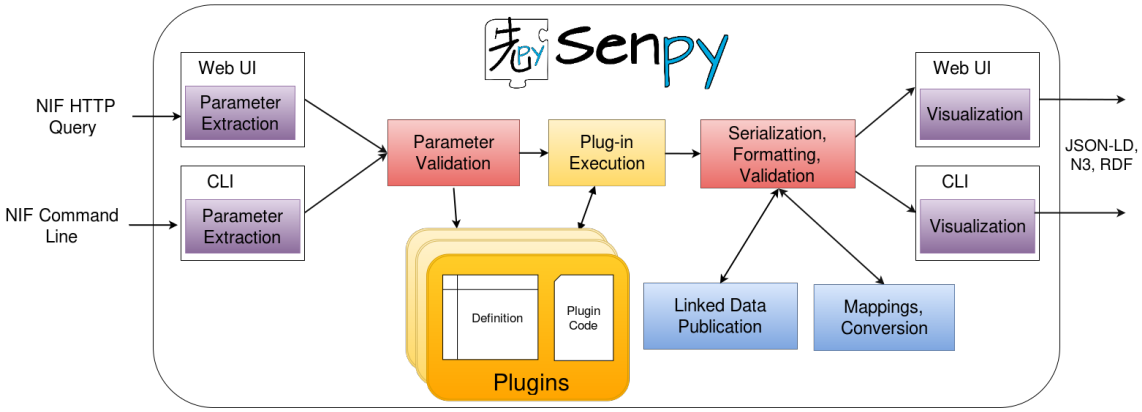


Figure 2.12: Senpy Architecture [9]

2.5 Docker

2.5.1 Docker Engine

Docker [71] consists on an open software container platform. Containers, unlike virtual machines, only pack the necessary libraries to work instead of the whole operative system, thus providing a way to virtualize applications and services in a lightweight way, a example is provided on Figure 2.14.

In addition, the Docker Engine adds a deployment layer to traditional containers. Thanks to this, deployment of applications and services can be automated independently of the host system they are being deployed on. The files that contain the instructions to deploy a service are called Dockerfiles.

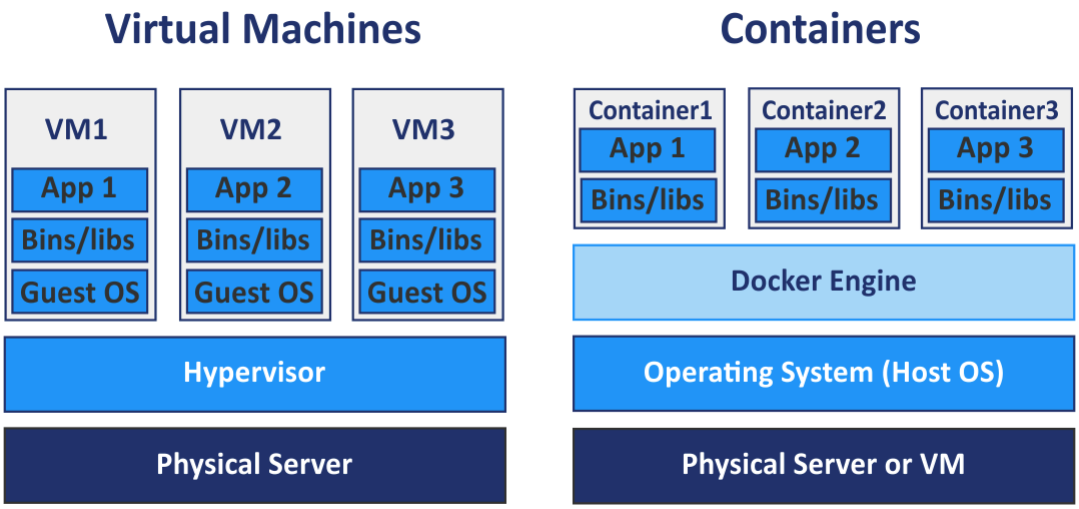


Figure 2.13: Virtual Machines vs. Containers [10]

2.5.2 Docker Hub

Docker Hub [72] is a repository of container images by community developers, open source projects and independent software vendors. For this project, all Docker Images used have been retrieved from this repository.

2.5.3 Docker Compose

Docker Compose [73] is a container orchestrator (that is, an automated manager of container services) used to deploy multi-container distributed applications with Docker. In order to deploy a service with Docker Compose the following steps are needed:

1. Define the environment of each service that the application will need with a Dockerfile and the necessary dependencies and working directories.

2. Define the different services that the distributed application will need in a YAML Ain't Markup Language (YML) file called docker-compose.yml.

3. This allows to create an inner network between the different containers with naming discovery, so containers won't need to know the Internet Protocol (IP) address every time they are deployed.

4. The docker-compose file also enables the user to declare dependencies between services so they are deployed in the correct order and shared data volumes.

5. Once all the files are defined, executing the command docker-compose up will deploy the different services the get the app running.

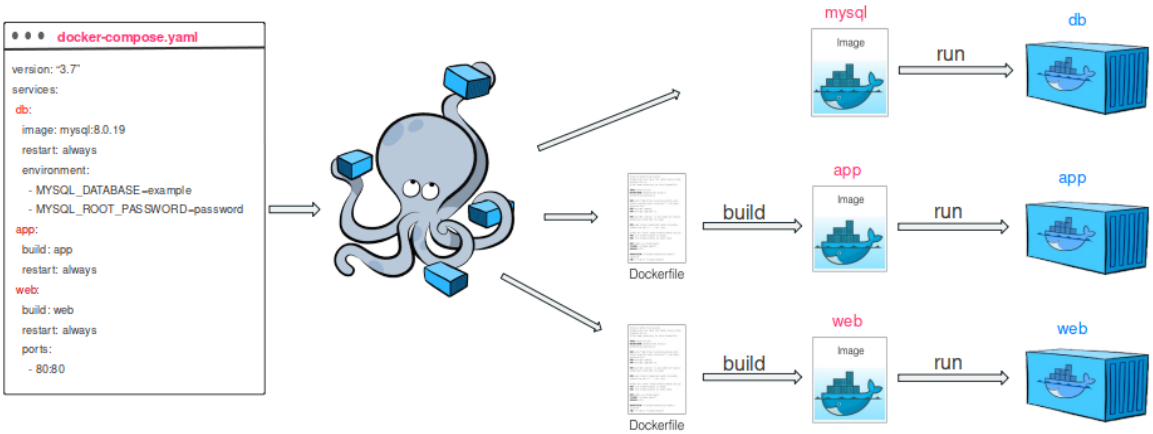


Figure 2.14: Deployment of a simple web app with Docker Compose [11]

2.6 Luigi

Luigi [74] is a module developed in Python by streaming service company Spotify. It has been designed to build pipelines to process batch jobs. The tool handles the management of workflows, resolution of dependencies and visualization through the web interface that it offers, enabling the user to search and filter.

Luigi informs about the whole process going from missing dependencies to execution failures and enables the user to separate function calls into tasks, isolating them. This is also useful to skip tasks that have already succeeded when running a batch pipeline and to debug code since it is fragmented into different tasks.

2.7 ElasticSearch

ElasticSearch [75] is an open-source, distributed, NoSQL, document-oriented database. This means that opposed to typical SQL databases, each register is a JSON object instead of a row inside of a table. It provides an analytics engine and Representational State Transfer (REST) API to request data by doing querys.

When data is requested through a query, it goes over all the saved documents through a technique called inverse indexing that provides results in a matter o milliseconds for every million documents. Querys can be done with the REST API through different methods like cURL, web browser or even dedicated programs like Postman.

Among its most important features, the following can be highlighted:

- **Scalability:** ElasticSearch provides cluster support so an scalable solution can be implemented. This can be done by replicating the data within the cluster, sharding the data (dividing it into bits within the cluster) or a mix of both. A cluster can be implemented in a standalone server or can be partitioned through the network into different servers.
- **Search Function:** The API lets the user request searches through a GET petition that can have a simple parameter format or a JSON format that indicates different parameters like the size of the returned document (number of elements), conditions that must be met and so on.

2.8 Polymer

Polymer [76] consists on an open-source JavaScript library developed by Google meant to develop web applications through the use of Web Components. Web Components [77] are a variety of technologies that allow developers to create custom, reusable, encapsulated HTML elements, thus providing interoperability of these elements between web services and applications. Web Components rely on the following technologies to work:

- Shadow Document Object Model (DOM): This allows to encapsulate code and styles to avoid unexpected interactions between components
- Custom Elements: A series of APIs to create HTML elements
- HTML Templates: Code stored and not rendered unless instantiated via Javascript

Polymer provides a layer of features over raw Web Components to make development of apps and web sites easier:

- Support for events triggered by gestures
- Data Binding (One-way & two-way)
- Cross-browser support (Some web component's features are only available for Firefox)
- Conditional and iterable HTML templates

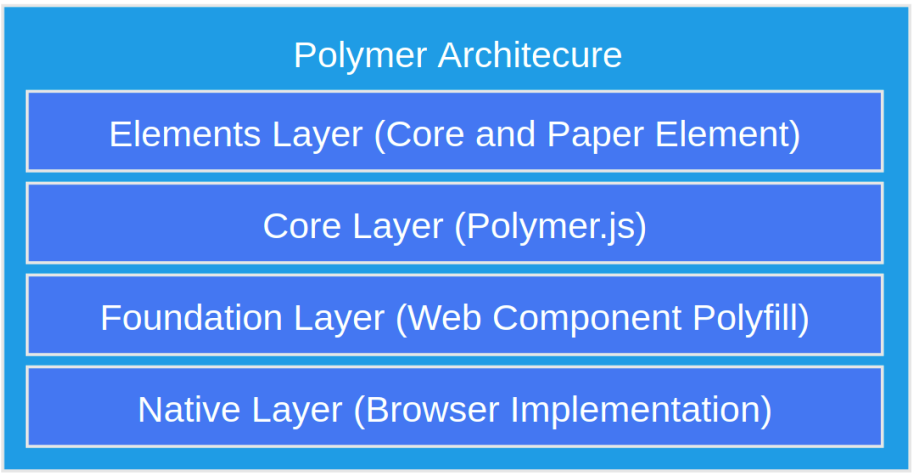


Figure 2.15: Polymer Architecture

2.9 Bower

Bower [78] is a Javascript package manager. In other words, is a tool that allows the user to record all project dependencies to keep them up to date. This is done through a file called bower.json, which saves the name of the project and the name and version of all the packages that the project is composed of.

Thanks to this, only a command line call is necessary to install the necessary dependencies. It's been selected for this porject because of the integration that has with Polymer. Furthermore, new packages can be install through the command line tool and Bower will add them to the bower.json file so the user doesn't have to.

2.10 Sefarad

Sefarad [12] is an environment developed by UPM's GSI to explore, analyse and visualize data. The tool provides a visualization module which main function is to represent data that is processed and enables the user to visualize interesting outputs. This visualisation gets structured in different dashboards. ElasticSearch provides the persistence layer and stores all the needed data. Data can also be retrieved from other sources as shown on Figure 2.16.

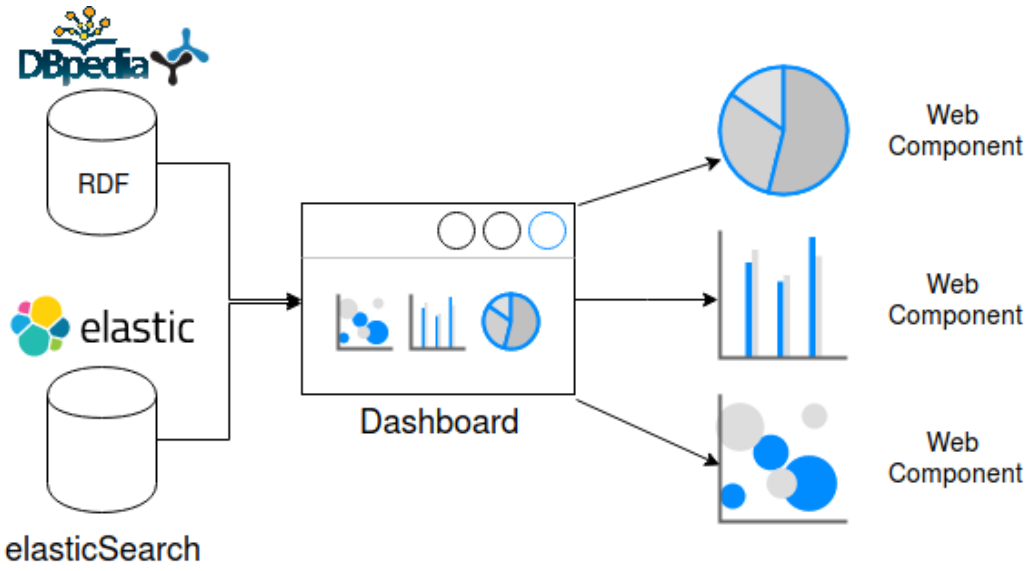


Figure 2.16: Sefarad Architecture [12]

CHAPTER 3

Architecture

3.1 Introduction

In this project, a platform to track sustainable mobility social trends will be developed. This platform consists on a dashboard that is integrated into the Cabify Observatory for Sustainable Mobility. It will share space with an already developed sustainable mobility news tracker.

The dashboard will provide the user a timeline of the latest relevant tweets about sustainable mobility alongside four charts with metrics about said tweets. The charts show the overall sentiment of the tweets (for which a sentiment model will developed), the category within the sustainable mobility space to which they belong (public transport, bicycles, trains, etc.), named entities found in the tweets and a word cloud to show the most repeated words in all the tweets. The models to categorize and detect named entities have already been developed and will be just consumed as a service using Senpy.

The dashboard will enable the user to filter through key words or info shown on the different components of the dashboard by just clicking. All the tweets collected for the purpose of the dashboard will be in Spanish. As shown in Figure 3.1 the project architecture is divided on four main sections:

Scrapping

Tweets about sustainable mobility are scrapped through the Twitter API. This process is further detailed in Section 3.4. At the end, tweets are stored as a JSON array locally.

Analysis

This is done through Senpy, which calls three plug-ins. The analysis looks for named entities, taxonomies and the sentiment to determine whether the tweet is positive, neutral or negative. Finally, the JSON is updated wih the new data acquired. A complete description of the Analysis is provided in Section 3.5.

Storing

When the processing phase has finished, tweets are uploaded from the local JSON file to the Elasticsearch database. This process is further detailed in Section 3.6.

Visualization

The dashboard is where all the processed data is shown. Database calls are done thorough Sefarad and the data is passed to the Polymer components to render the web page. A complete overview of the dashboard is provided in Section 3.7.

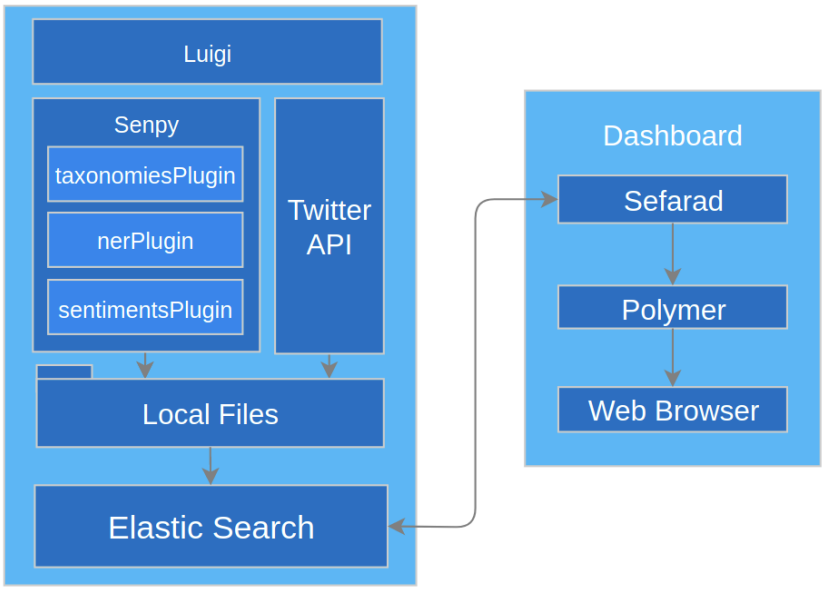


Figure 3.1: Project's Architecture

3.2 Luigi Pipeline

The connection between the process that scrapes tweets and the process that indexes in ElasticSearch is controlled and supervised by the orchestration tool Luigi. A pipeline has been created to retrieve, analyze, and publish the data. It is executed every 24 hours to retrieve recent tweets. During this process, the tweets are enriched following a Linked Data structure (Figure 3.3). As shown in Figure 3.2, the pipeline is made up of three tasks:

1. **Twitter Scrapper Task:** With the Twitter API, tweets are retrieved from the platform. They are identified by their URL (Uniform Resource Locator) so a verification to check whether they already exist on the database can be done. Only those messages with more than 2 retweets or 5 likes are retrieved. The tweets must contain the terms "movilidad sostenible", "transporte sostenible" or the term "contaminacion" alongside "no2", "nox" or "dioxido de nitrogeno". Afterwards, all tweets that consist on a response to a post are deleted so only original tweets are retrieved. Finally, the retrieved data is saved in a JSON file so the next task can pick it up and continue.

2. **Tweets Analyzer Task:** This task makes use of Senpy and three plug-ins created for the purpose to analyze the tweets. With each analysis, the Linked Data structure is enlarged. On Section 3.5 a detailed description of this process is provided. Once the analysis has finished, the JSON is updated with the new data and the last task begins. A total word count is also performed.

3. **Tweets Search and Store Task:** This last task iterates over the JSON file with all the already processed tweets and uploads them to database Elasticsearch.

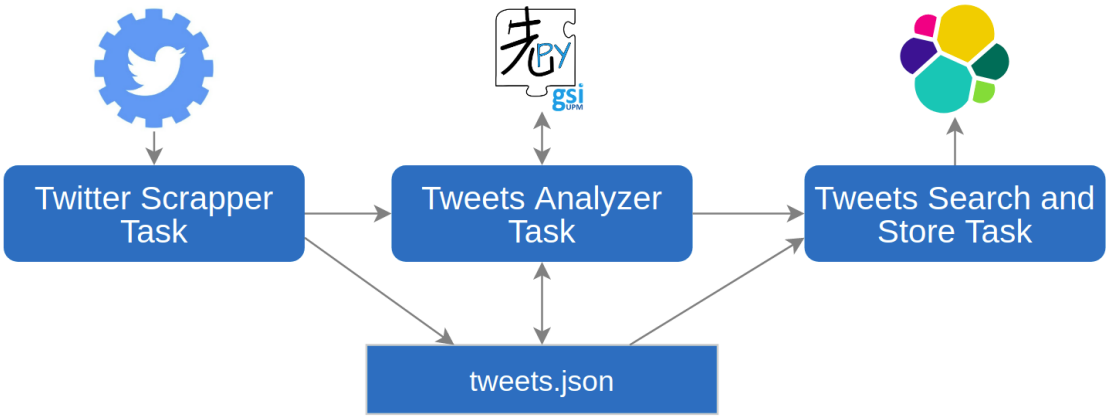


Figure 3.2: Luigi Pipeline

3.3 Tweet Linked Data Structure

The ontologies used to make up the data structure are mentioned on Section 2.1.4. The element itself consists on a **schema:SocialMediaPosting** which represents a post to a social media platform. It is made up of:

- **sioc:Content**: The content of the post, that is, the text.
- **dc:created**: Timestamp of the publication of the post.
- **sioc:id**: The identifier of the post within the social media (URL).
- **sioc:has_creator**: Denotes the existence of an author of the post.
  - **sioc:UserAccount**: Represents an online service's user account.
    - **foaf:img**: Link to an image.
    - **sioc:id**: User's account handle. Is unique.
    - **sioc:first_name**: User's account name.
- **schema:InteractionStatistic**: The number of interactions for a post. The most specific child type of InteractionCounter should be used
  - **schema:InteractionCounter**: A summary of how users have interacted with the post. A subtype to specify the specific type of interaction should be used.
    - **schema:InteractionType**: The Action representing the type of interaction. For likes, LikeAction is used. For sharing (RTs), ShareAction is used.
    - **schema:userInteractionCount**: The number of interactions for the post.
- **prov:Activity**: Denotes the occurrence of something, in this case, a **marl:sentimentAnalysis**, which generates (**prov:generated**) a **marl:opinion**. This describes the concept of opinion expressed in a certain text.
  - **marl:hasPolarity**: Indicates if the opinion is positive/negative or neutral. Use instances of class marl:Polarity.
    - **marl:Polarity**: A string representing the sentiment of the tweet. It can take the following values:
      - **marl:Positive**: Denotes a positive sentiment.
      - **marl:Neutral**: Denotes a neutral sentiment
      - **marl:Negative**: Denotes a negative sentiment.

- **senpy:hasEntitites**: The entities within the post. Generates **nif:NamedEntity**, which represents a named entity. It is composed of:
  - **schema:Name**: Name of the entity
  - **nif:anchorOf**: Text where the entity has been detected.
  - **nif:beginIndex**: Position of the first character within the text where the entity has been detected.
  - **nif:endIndex**: Position of the last character within the text where the entity has been detected.
  - **prov:wasDerivedFrom**: Link to the origin of the entity
- **dct:subject**: Used to represent a subject within a controlled vocabulary. In this case, the category within the sustainable mobility space which the tweet belongs to. The controlled vocabulary is an ontology that was created for this purpose, which can be found at Appendix C. It generates an **skos:Concept**, which represents a taxonomy. It is composed of:
  - **schema:Name**: Name of the taxonomy
  - **nif:anchorOf**: Text where the taxonomy has been detected.
  - **nif:beginIndex**: Position of the first character within the text where the taxonomy has been detected.
  - **nif:endIndex**: Position of the last character within the text where the taxonomy has been detected.

The final values that the attributes may have are:

- **rdfs:Literal**: Unicode string value.
- **xsd:Integer**: An integer number.

In Figure 3.3 a diagram of the Linked Data structure mentioned above is provided.

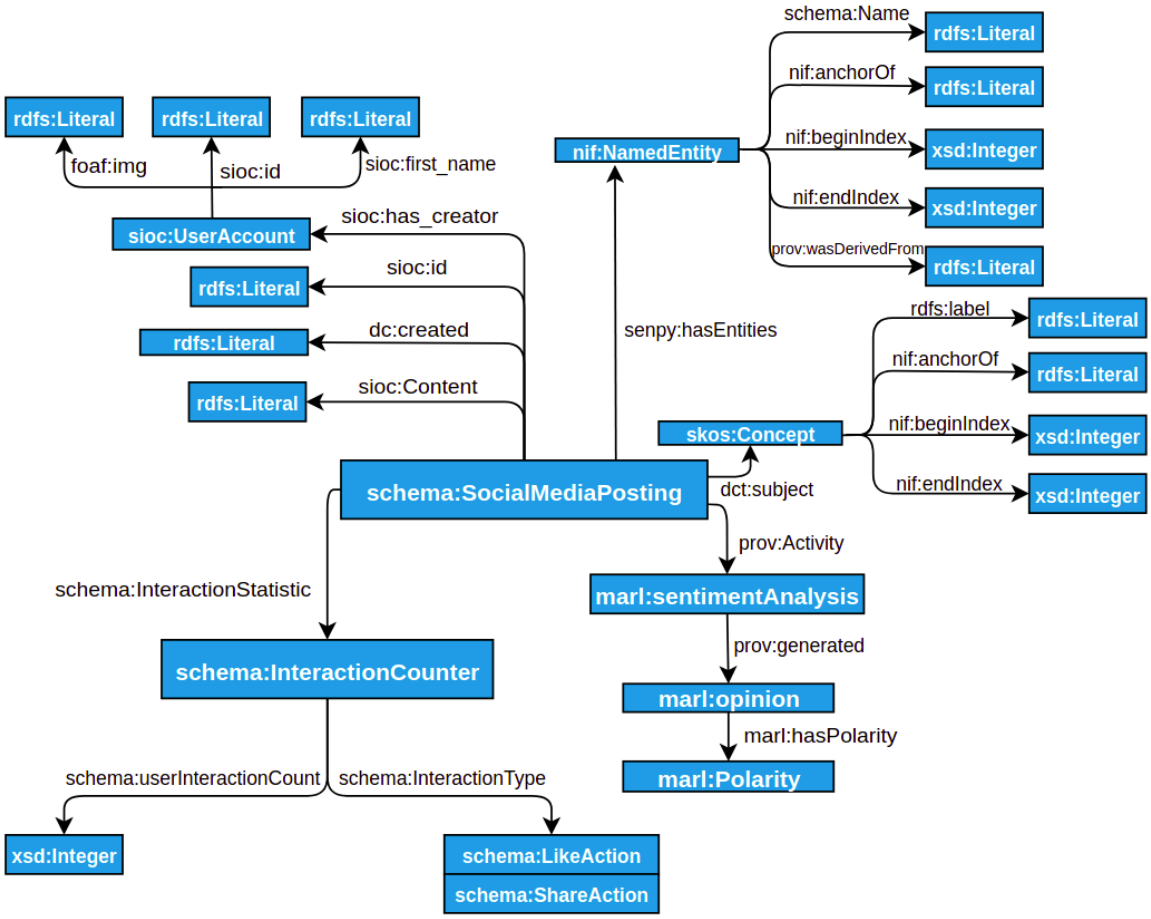


Figure 3.3: Tweet Linked Data Structure

3.4 Tweets Retrieval

In order to use the Twitter API, four API keys are needed. These keys are: consumer key, consumer secret, access token and access token secret. To get them, a developer account is needed. Since searches are limited to 100 tweets, an auxiliary Python function has been created where the number of tweets desired is indicated and repeats API calls until necessary. The query used to retrieve the tweets is:

```
("movilidad sostenible") OR ("transporte sostenible") OR (contaminacion (
    no2 OR nox OR "dioxido de nitrogeno")) -RT -filter:retweets
```

In this function, the first JSON structuring the tweet is created. Here, all the metadata related to tweet is saved (user account, id, creation date and interaction statistics).

3.5 Senpy Analysis

Once the tweets have been retrieved, they go through data analysis tool Senpy in order to be processed and analyzed (Figure 3.4). This is done thanks to three plug-ins designed to retrieve the different data attributes selected for the project, which are named entities, taxonomy and sentiment. The plug-ins are:

- **Taxonomies:** Determines Spanish categories for the tweets (Bicicleta, Infraestructuras Sostenibles, Coche Electrico, Peaton, Tren, Movilidad Inteligente, Transporte publico, Motocicleta Electrica, Autobus, Tranvia). A complete description of the taxonomy used can be found in Appendix C.
- **Spanish Sentiment Tweets:** Determines sentiment of the tweet (Positive, neutral or negative). A complete description of the development of this model is provided in Chapter 4.
- **Named Entities Recognition:** Detects named entities (real-world object with a proper name).

Because of compatibility issues with the spaCy libraries (The sentiments plugin uses spaCy 3.0 while the others use spaCy 2.3) two separate instances of Senpy have been deployed in Docker. When the process is over, the Linked Data enrichment is finished and the JSON is finally prepared to be uploaded to Elasticsearch.

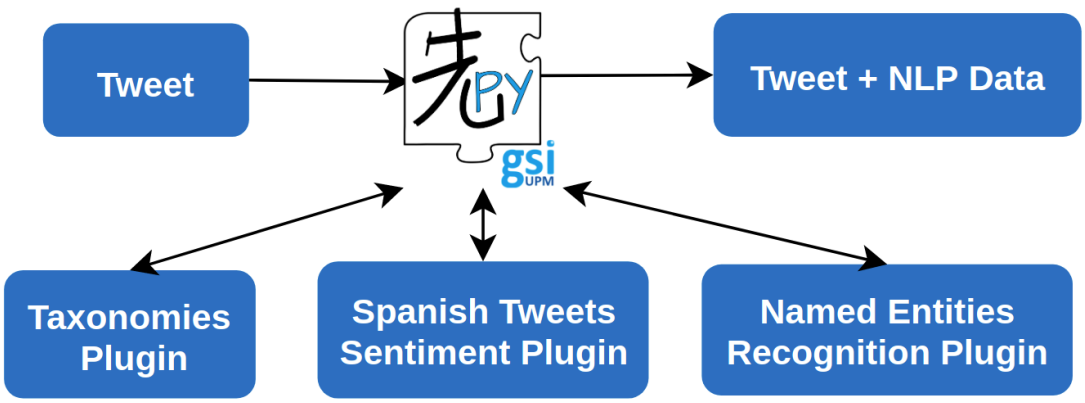


Figure 3.4: Senpy Working Scheme

3.6 Elasticsearch Indexing

Elasticsearch works with indexes. An index is to a document-oriented database what a database is to a relational database. It has a mapping which defines multiple types (tables).

A total of two indexes have been created for this project:

- **Tweets:** Contains a list of all the tweets. The documents are formatted following the Linked Data structure seen in Figure 3.3.
- **Words:** Contains a list of all the words of the processed tweets with a counter to know how many times they have been repeated. Every time new tweets are retrieved this index is updated with the new word count.

When using Elasticsearch, one of the crucial tasks is the correct selection of an indexes structure. Structuring the indexes and a precise mapping for the documents is essential for a good performance in a document-oriented database. Nevertheless, the usage of Elasticsearch in this project does not require such an in-depth exploration, since data will be pushed once a day through the Luigi pipeline.

Once Senpy has finished processing all the data, the JSON array is iterated uploading each tweet as an individual document to the Tweets index.

3.7 Dashboard

The web application is deployed with a simple http-server and doesn't have back-end per se. In order to function, it relays on Front-end technologies and Elasticsearch. Data is requested with asynchronous petitions without the necessity of a middle-ware layer thanks to Sefarad. In the web architecture we have two differentiated parts:

- Front-end:
  - Sefarad: Handles the requests and responses to ElasticSearch of filtered data or whole indexes.
  - Polymer Components: Web components that pick up Sefarad data and render different views.
  - Bootstrap, jQuery, HTML, CSS, Javascript: Ultimately, web components are made up of code helped by the polymer libraries but relay on basic web code in order to function.
- Elasticsearch: Handles request and responses to Sefarad with necessary data.

The final user accesses the web platform via a web browser. There, two options are presented, either select the news dashboard, or the tweets dashboard. After selecting the desired option, the dashboard is presented.

Figure 3.5: Web Application Scheme

The dashboard itself (Figure 3.6) consists on a Polymer component and is subdivided in 7 polymer components.

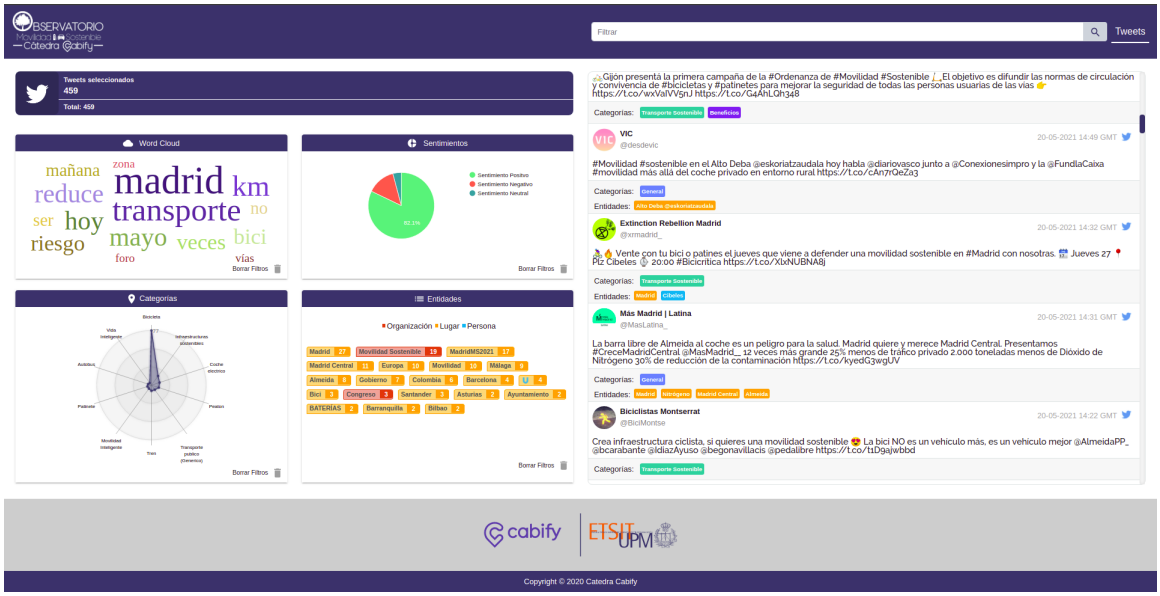


Figure 3.6: Dashboard component

3.7.1 Number-chart component

As seen in Figure 3.7, this component shows the total count of tweets currently present in the Elasticsearch database. Also, when filters are applied, the component changes to count the number of tweets that meet the filters criteria, as seen in Figure 3.8.

Figure 3.7: Number chart component

3.7.2 Filters-viewer component

The mission of this component is to show the filters that are currently active, which can be set from the search bar or clicking on other components.

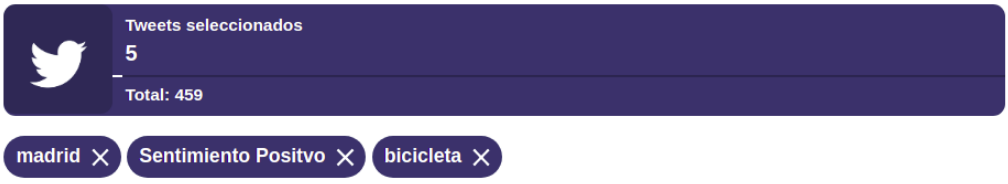


Figure 3.8: Number chart component

3.7.3 Poly-cloud component

As seen in Figure 3.9, this component presents the user a word cloud with the most repeated words. With jQuery, it makes a request to the words index (Section 3.6) and presents them in the component. Also, by clicking in the words, it applies a search filter containing that term.

Figure 3.9: Poly-cloud component

3.7.4 Google-chart component

This component presents a pie chart representing the percentage of sentiments present in the tweets. Green represents positive sentiment, red represents a negative sentiment and blue represents a neutral sentiment. By clicking in each color, or in the text, a filter is applied to show only tweets that meet the sentiment criteria. An example is provided in Figure 3.10. The sentiment data is retrieved with the Senpy plugin that depends on the model created for this purpose.

3.7.5 Radar-chart component

This component presents a radar chart, also known as spider chart, that counts the tweets belonging to certain taxonomies. Also, by clicking in the different categories, it filters through those selected. An example is provided in Figure 3.11. The taxonomies data is provided by an already created model that is consumed through Senpy.

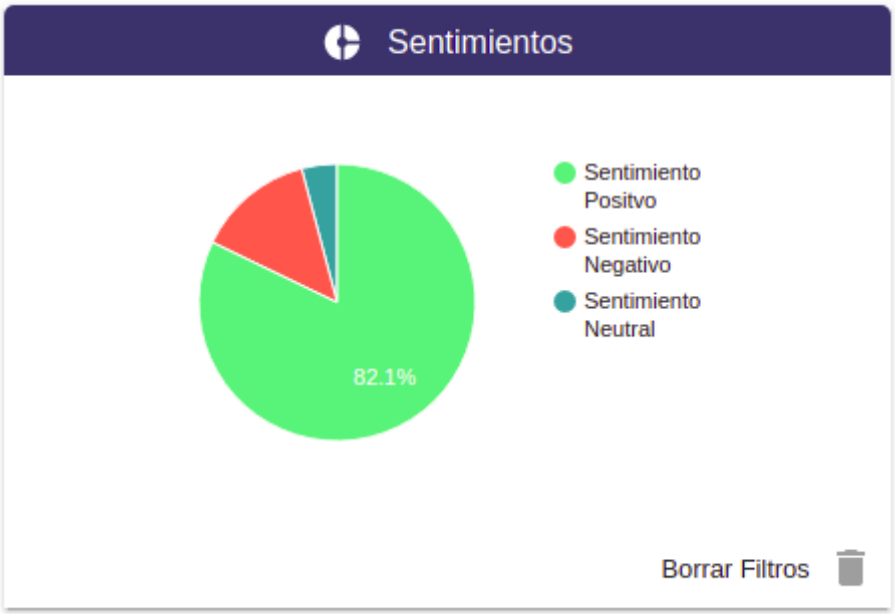


Figure 3.10: Sentiment pie chart component

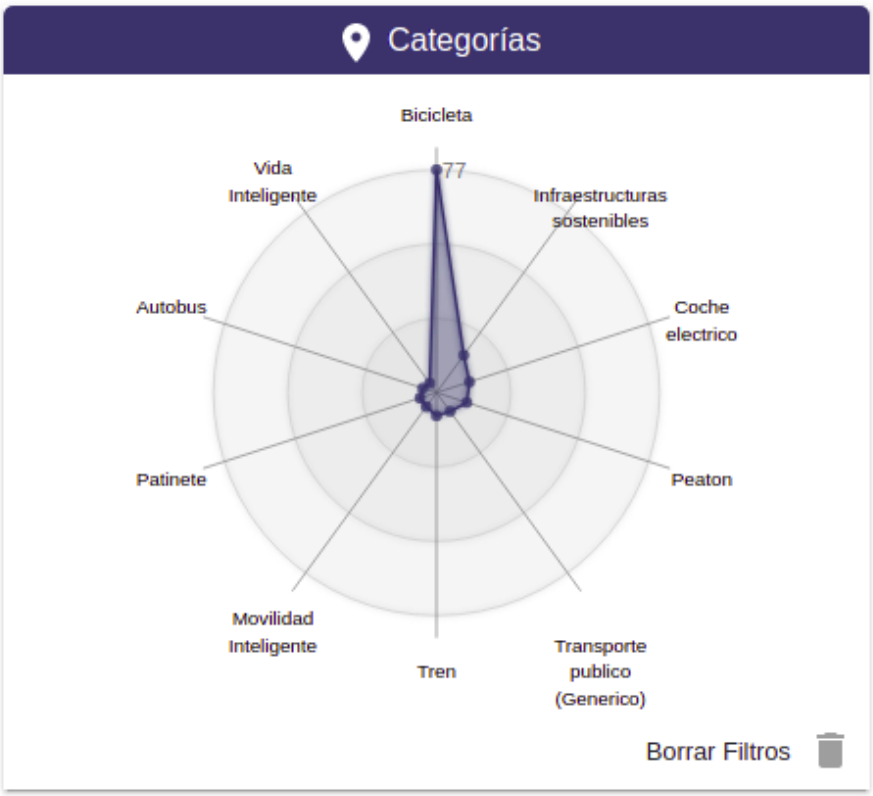


Figure 3.11: Radar chart component

3.7.6 Entities-chart component

As seen in Figure 3.12, this component aggregates the most repeated entities in a panel, classifying them by organization, place or person. The named entities data is provided by an already created model that is consumed through Senpy. The component also enables the user to filter through entities by clicking on them.

Figure 3.12: Entities chart component

3.7.7 Tweet-viewer component

This component shows all the tweets according to the filter criteria. In Figure 3.13 an example is provided. By default, it shows all tweets present in the Elasticsearch tweets index (Section 3.6). It is an scrollable component where the tweet (Figure 3.14), alongside creator data and date is showed. Information about entities and categories is shown too. It also lets the user to go to the original tweet by clicking the blue Twitter logo located in the top right corner of each tweet.

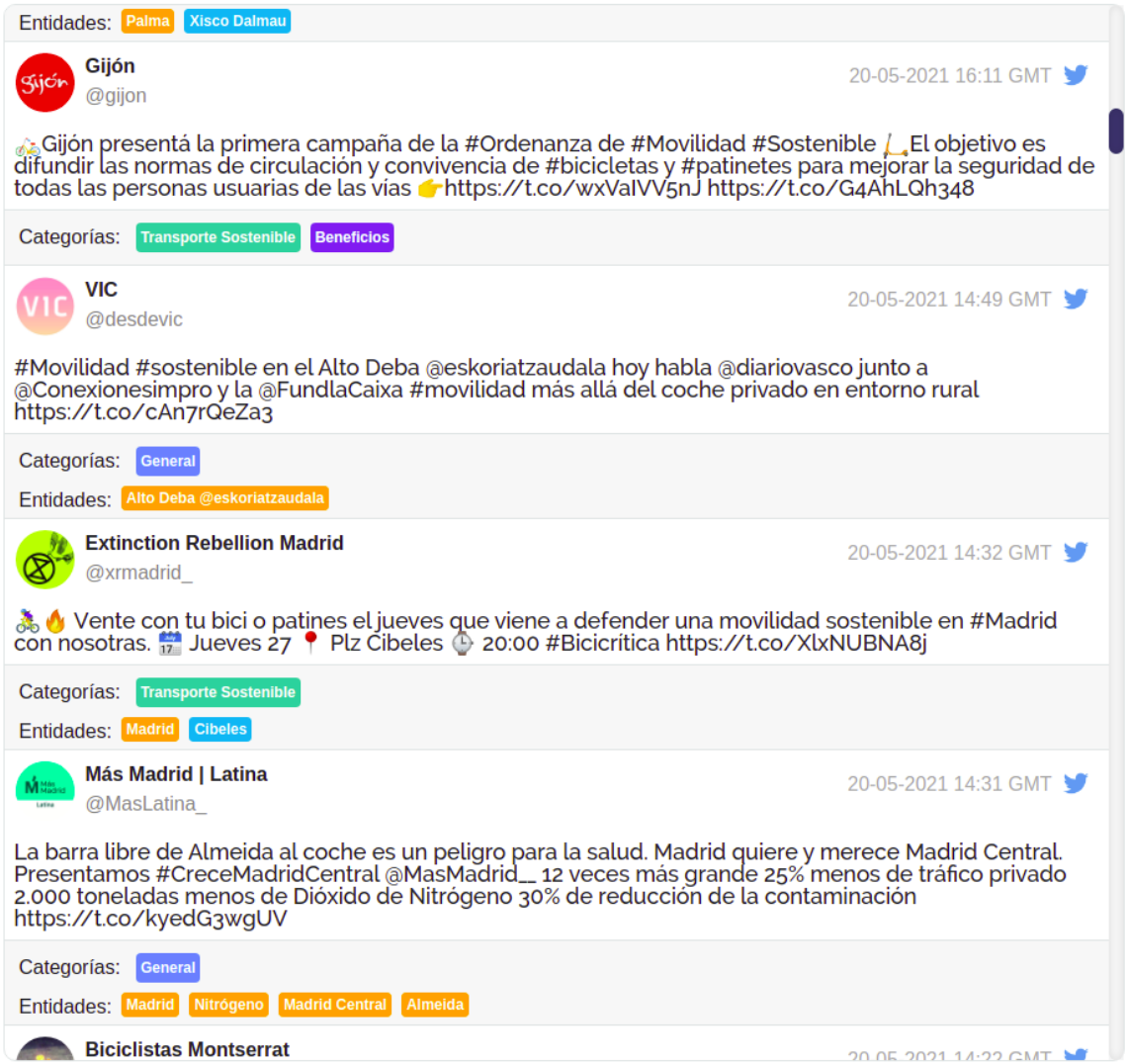


Figure 3.13: Tweets viewer component

Figure 3.14: Single Tweet

3.8 Docker Compose Architecture

To avoid interference between dependencies and libraries, each service is deployed in an individual Docker container. The whole set-up is deployed by Docker Compose.

There is a total of five containers, two volumes and one shared network so services can interact among each other and with the outside. In Figure 3.15 a diagram of the containers, exposed ports, volumes and network within the Docker Compose architecture is provided.

- **Containers:**
  - **luigid:** Luigi image to execute pipelines. It is exposed on port 8082 so it can be accessed from the outside.
  - **senpy:** Senpy image for the taxonomies and named entity recognition plugins. It is exposed on port 5000 so the GUI can be used independently from a browser.
  - **senpy-spacy3:** Senpy image for the Spanish tweets sentiment plugin. It is exposed on port 5000 so the GUI can be used independently from a browser. However it is forwarded to port 5001 since the port 5000 is already in use
  - **elasticsearch:** Elasticsearch image to deploy the database.
  - **dashboard:** Deploys the dashboard and exposes the port 8080.
  - **orchestrator:** Contains all the data that Luigi relays on, auxiliary functions and classes. Triggers the Luigi pipeline every 24 hours. Depends on all the previous containers to work.
- **Volumes:**
  - **observatorydata1:** Persistent volume used by the orchestrator to keep metadata and logs.
  - **esdata1:** Persistent volume used by Elasticsearch to store all the data
- **Network: dashboard-network**, the network shared by all containers in order to communicate with one another. Also the gateway for external access to the different services.

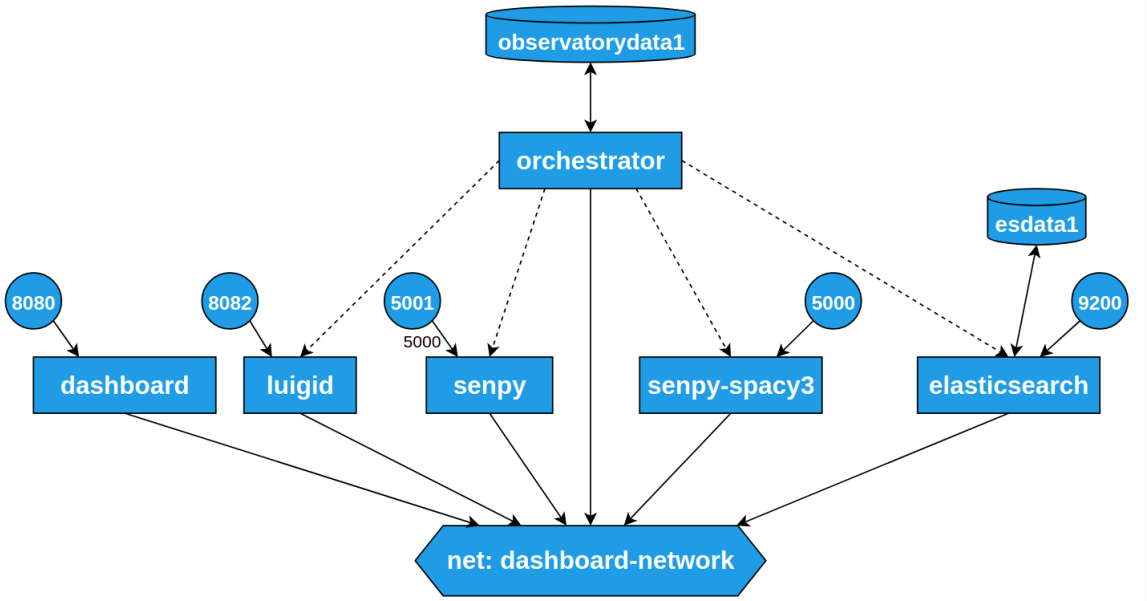


Figure 3.15: Docker Compose Architecture

CHAPTER 4

Sentiment Analysis Model

4.1 Introduction

In this chapter, a description of how the natural language processing Spanish sentiments model has been developed will be provided. The model analyzes Spanish tweets to determine whether the sentiment is positive, negative or neutral.

To create the model, NLP Library spaCy 3.0 has been used, this version was released in February 2021 and provided many features and changes on the use of the tool, alongside new architectures, that will be put in place in this development.

To train the model, the Spanish Society for Natural Language Processing's (SEPLN by its Spanish acronym) 2012 Semantic Analysis Workshop at SEPLN (TASS by its Spanish acronym) corpus [79] has been used.

To evaluate the different models created the library Sci-kit learn has been used, more specifically, the functions used to get the confussion matrix, and the accuracy, precission, recall & F1-score.

4.2 TASS Dataset

The SEPLN is a non-profit association created in 1983 to promote NLP in Spain. To achieve this goal, the TASS competition was created in 2012 with the aim of furthering the research on sentiment analysis in Spanish.

To do so, the competition provided a large corpus with thousands of tweets in Spanish indicating their sentiment. The corpus consists on an XML file with tweets, where each message is tagged with its sentiment. A total of five levels have been defined: strong positive (P+), positive (P), neutral (NEU), negative (N), strong negative (N+) and one additional no sentiment tag (NONE). On Figure 4.1 the XML structure is provided.

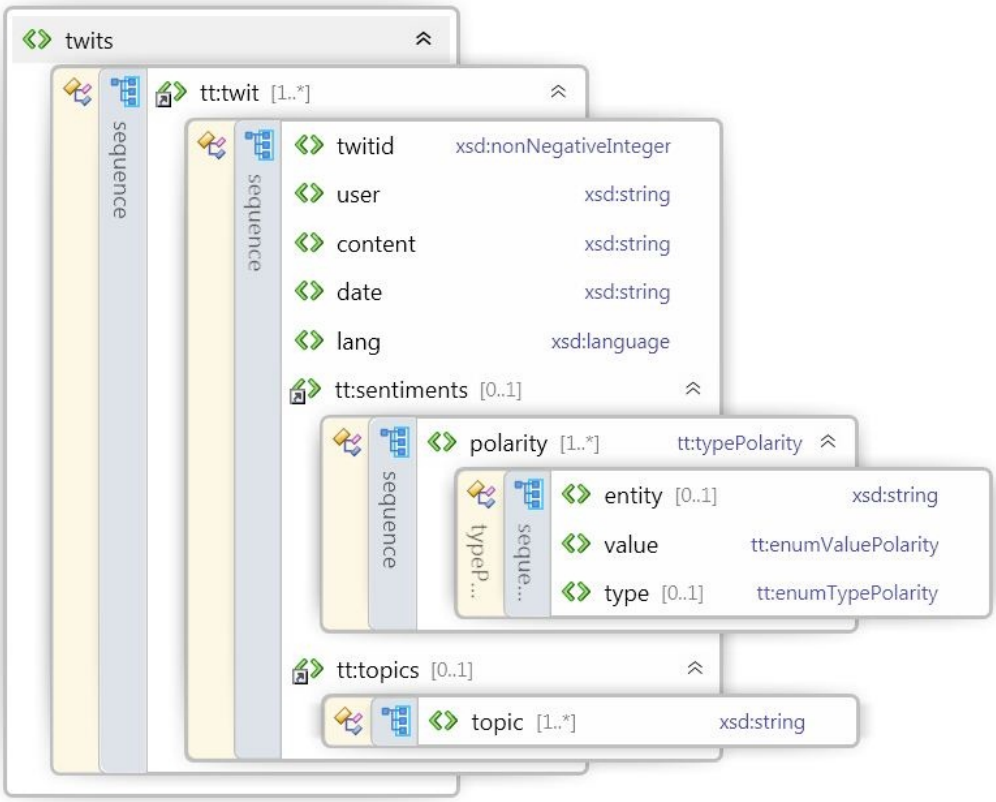


Figure 4.1: TASS XML Structure [13]

On Appendix D the code of XML Schema (XSD) is presented, alongside an example of a tweet for every sentiment.

4.2.1 Data Conversion

To create the model, only the text of the tweet and the global sentiment is needed. Since the TASS corpus is provided in an XML format and provides extra information that is not needed, the data is converted to a CSV format where each row just contains the needed attributes.

In order to do so, the XML processor BeautifulSoup4 is used, which easily picks up the data between the tags and saves it in a Pandas Dataframe that is later saved to a CSV file. During this process, there are some texts containing the semicolon character (;), so to avoid any problems when saving the CSV, they are changed for a non-breaking space character during the XML processing.

The TASS corpus provides a train and a test set. During the conversion, they have been joined as one to ease the latter data munging process. The TASS corpus tags the sentiment of the tweet using six categories:

- Strong Positive (P+)
- Positive (P)
- Neutral (NEU)
- Negative (N)
- Strong Negative (N+)
- No Sentiment Tag (NONE)

Since the model will predict only three categories (positive, neutral and negative), the sentiment categorization of the tweets has been transformed following the schema provided below:

- Positive (P): Covers strong positive (P+) and positive (P).
- Neutral (NEU): Covers neutral (NEU) and no sentiment (NONE).
- Negative (N): Covers negative (N) and strong negative (N+).

4.2.2 Data Exploration

Once the CSV transformation is complete, the Tweets dataset will be explored with the help of Pandas and Seaborn. In the Tweets dataset (Table 4.1) there is a total of 68.017 tweets with a categorized sentiment.

tweet	value
Portada 'Público', viernes. Fabra al banquillo...	N
Grande! RT @veronicacalderon "El periodista es...	NEU
Gonzalo Altozano tras la presentación de su li...	P
Mañana en Gaceta: TVE, la que pagamos tú y yo,...	N
Qué envidia "@mfcastineiras: Pedro mañana x la...	NEU

Table 4.1: Tweets Dataset

In Figure 4.2 is shown that there is a total of 25.117 tweets with positive sentiment, 24.874 tweets with neutral sentiment, and 18.026 tweets with negative sentiment.

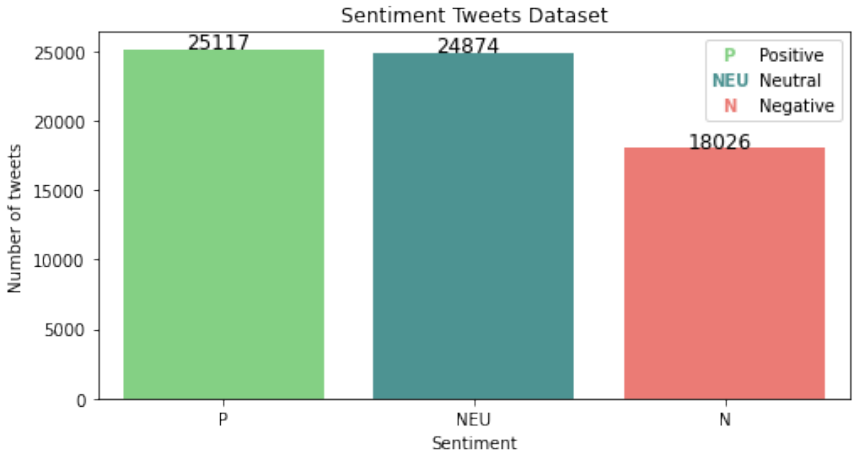


Figure 4.2: Tweets Dataset Sentiment Distribution

4.3 Data Pre-Processing

4.3.1 Text Cleaning

Before starting with proper NLP tasks, the tweets have gone through a cleaning to remove all URLs, emojis, hashtags (#something) and user handles (@user) present in the texts. This is done through a python script that batch processes the texts and returns them without those elements.

4.3.2 Data Split

To create a model three datasets are needed:

- **Train:** Data used to fit the model.
- **Validation:** Data to evaluate the fitness of the model to tune hyper-parameters while the fitting is taking place.
- **Test:** Data to evaluate final model fit.

From the original tweets, 25% will go to the test dataset. The rest, will be split 80% for the train dataset and 20% to the valid dataset. The final data distribution can be seen on Figure 4.3.

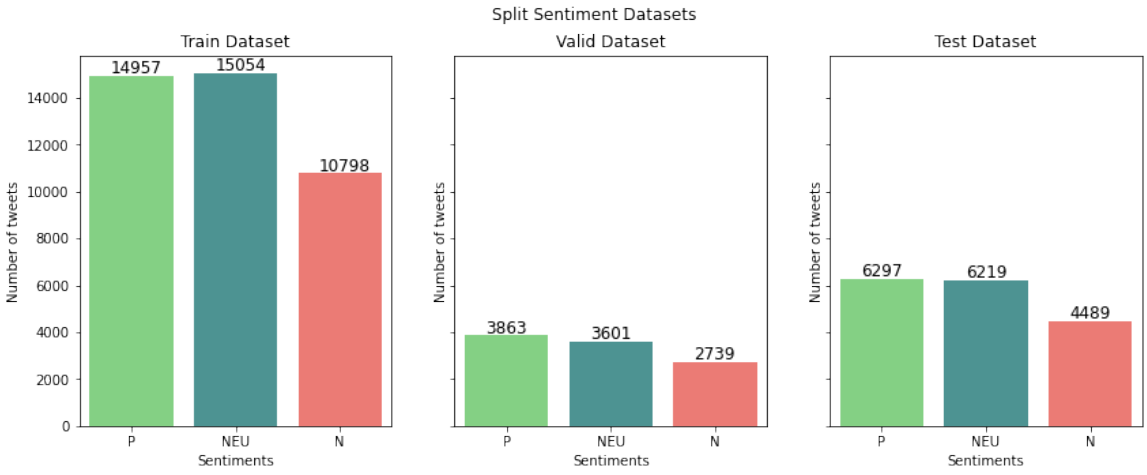


Figure 4.3: Data Distribution in Train, Valid & Test

4.4 Model Architectures

To create the models, NLP library spaCy has been used. Spacy offers a wide range of architectures to create models. In this section, a brief overview of the architectures used is given. A complete description of the architectures is provided in Appendix E.

4.4.1 Convolutional Neural Network Tok2Vec-based

A neural network model where token vectors are calculated using a Convolutional Neural Network (CNN). First, words are converted to their mathematical representation, this could be understood as a vector and is called word embedding.

These vectors are calculated through the use of spaCy's Tok2Vec token-to-vector model [80]. The vectors are mean pooled and used as features in a feed-forward network, that way, in function of those features, the category to which the tweet belongs to is determined.

4.4.2 Ensemble: Bag of Words + CNN

Stacked ensemble of a linear bag-of-words model and a CNN model. In the bag-of-words model, the sentences are represented as a set of the words that compose the sentence, only keeping the number of times that they are repeated and disregarding any other kind of information like the order. The neural network is built upon a Tok2Vec layer. This architecture is usually more accurate than the CNN, but runs slower.

4.4.3 Convolutional Neural Network Transformer-based

This architecture is somewhat similar to the simple Convolutional Neural Network. However, to produce tokens, instead of the Tok2Vec model, transformers are used. Transformers are deep learning models that use the attention mechanism. This means that it enhances the most important parts of the introduced data and ignores the rest.

For this project, the transformer used is BETO [81], a Spanish version of BERT [37]. BERT stands for Bidirectional Encoder Representations from Transformers, it has been developed by Google and provides great results. The pre-trained transformer BETO is consumed through Huggingface's transformers library [41].

4.5 Models Evaluation

4.5.1 Evaluation Metrics

In this section the different metrics to compare between classifiers will be explained. For a multi-class classifier like the one created, metrics differ a bit compared to binary classifiers. When evaluating a classifier, the following items will be calculated for each class:

- **True Positives (TP):** The classifier correctly predicts the item as belonging to its true class.
- **True Negatives (TN):** The classifier correctly predicts the item as not belonging to the class.
- **False Positives (FP):** The classifier wrongly predicts the item as belonging to the class.
- **False Negatives (FN):** The classifier wrongly predicts the item as not belonging to the class when it really does.

Once this items are calculated, the confusion matrix can be calculated, alongside the following metrics, which are calculated for every class:

- **Precision:** Percentage of predicted labels that actually belong to that class.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall:** Percentage of actual labels that are correctly classified.

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score:** Harmonic mean of precision and recall.

$$F1 = \frac{2 * Recall * Precision}{Recall + Precision}$$

Lastly, the macro average for each metric is calculated. This is simply the mean of each metric, taking into account the number of elements per class. This process is repeated for every classifier.

4.5.2 Evaluation

4.5.2.1 CNN Tok2Vec-based

The precision, recall and F1-score for the negative, neutral and positive class alongside the macro average for the CNN model based on Tok2Vec word embeddings are shown in Table 4.2. It can be seen that overall, the class with the poorer performance is the neutral class and the one with the best performance is the positive sentiment class.

	Precision	**Recall**	**F1-Score**
Negative	0.72	0.80	0.76
Neutral	0.75	0.67	0.71
Positive	0.81	0.83	0.82
Macro Average	0.76	0.77	0.76

Table 4.2: CNN Tok2Vec-based Architecture Classification Report

In Figure 4.4 the poorer performance of the neutral class is confirmed, with only 67% of neutral tweets being identified as such. However, in the negative and positive class an 80% and 83% recall has been obtained respectively. Furthermore, only 7% of negative sentiment tweets are identified as having a positive sentiment and 5% for the reverse case.

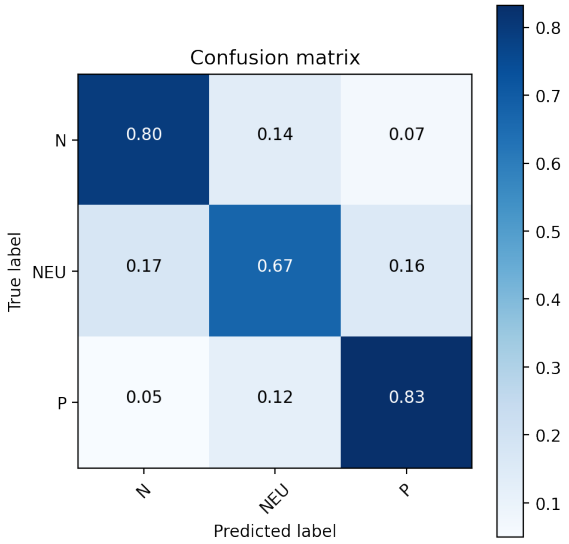


Figure 4.4: CNN Tok2Vec-based Architecture Normalized Confusion Matrix

4.5.2.2 Ensemble

The different metrics for the bag-of-words and CNN ensemble model are shown in Table 4.3. It can be seen that again, the class with the worst performance is the neutral class. However, recall has gone from a 67% to a 73% while the precision has dropped a 3%. Recall in the Negative class has dropped a 4%, while the precision has risen a 5%, leading to the same F1-Score. Overall, the class with the best performance is the positive sentiment class.

	Precision	**Recall**	**F1-Score**
Negative	0.77	0.76	0.76
Neutral	0.72	0.73	0.73
Positive	0.84	0.83	0.83
Macro Average	0.78	0.77	0.77

Table 4.3: Ensemble Architecture Classification Report

In Figure 4.5 the poorer performance of the neutral class can be seen again. The recall in the neutral class has gone up, while in the negative class has gone down. In the positive class it is maintained. It is also important to note that less positive sentiment tweets are mistaken as being negative and vice-versa

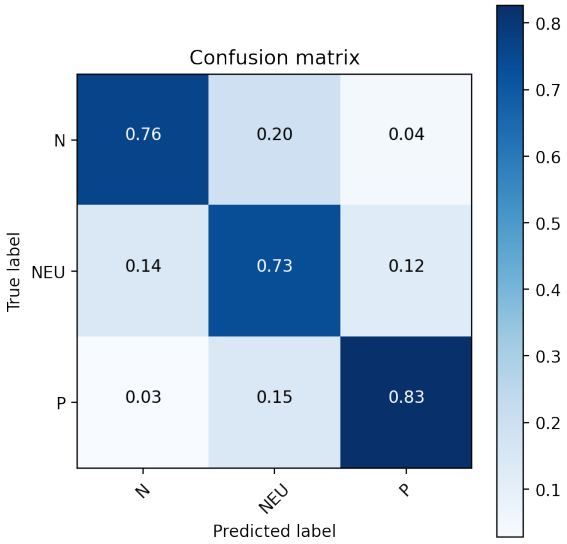


Figure 4.5: Ensemble Architecture Normalized Confusion Matrix

4.5.2.3 CNN Transformers-based

The different metrics for the CNN model based on transformer word embeddings are shown in Table 4.4. Overall, all metrics for all classes have dropped compared to the CNN Tok2Vec & ensemble architectures.

	Precision	**Recall**	**F1-Score**
Negative	0.63	0.75	0.68
Neutral	0.68	0.52	0.59
Positive	0.71	0.78	0.74
Macro Average	0.67	0.68	0.67

Table 4.4: CNN Transformers-based Architecture Classification Report

In Figure 4.6, the general poorer performance is confirmed, with a 52% recall for the neutral class, while just droppping a 5% for the negative and positive class compared to the CNN Tok2Vec based architecture. Also, this architecture is where more positive sentiment tweets are mistaken as negative and vice-versa.

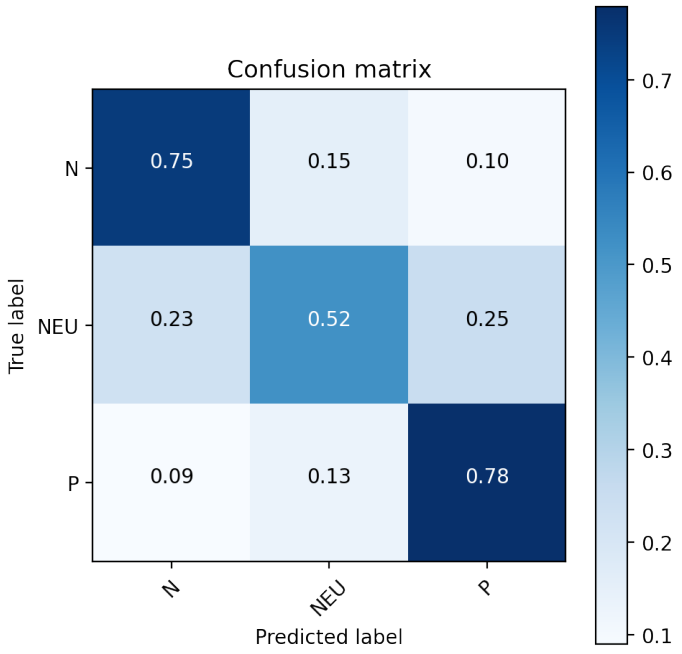


Figure 4.6: CNN Transformers-based Architecture Normalized Confusion Matrix

4.6 Model Selection

In Table 4.5 a comparison between the macro average F1-Score, the processing time and each of the architectures' requisites is provided.

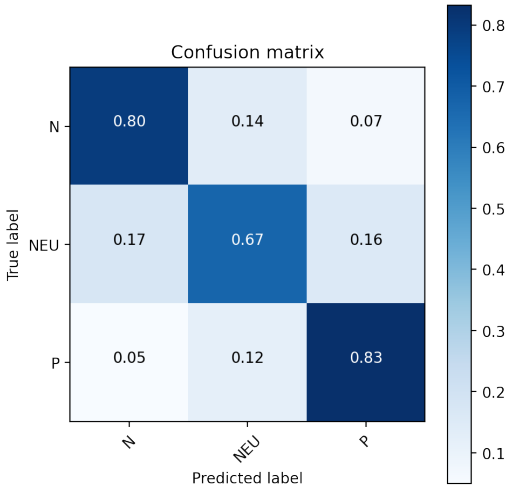
Model Architecture	Macro F1-Score	Processing Time	Requisites
CNN - Tok2Vec	0.76	220 tweets/second	Word Vectors
Ensemble	0.77	40 tweets/second	Word Vectors & Lexicon
CNN - Transformer	0.67	2.5 tweets/second	Word Vectors & Transformers

Table 4.5: Comparison of architectures

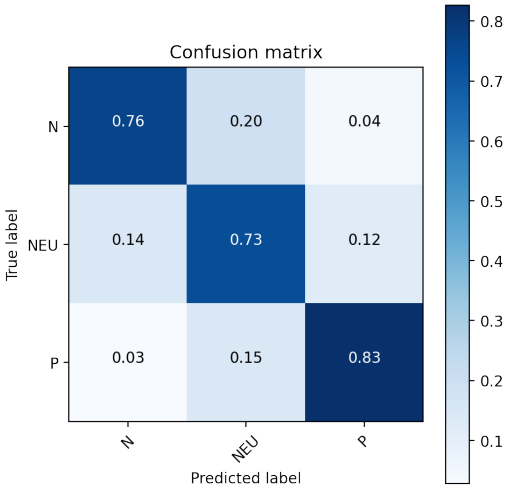
In addition, a comparison between the three confusion matrices is shown in Figure 4.7. The transformers-based model is automatically discarded because of its poor performance with the neutral class. Also, the processing time it offers won't be useful in a real-life production environment.

Although it's true that the ensemble model mistakes less positive tweets as negative and vice-versa, the recall in the negative class is worse than in the CNN Tok2Vec based model. Furthermore, the macro average F1-Score only differs by 1% between these two models. Nevertheless, the recall for the neutral class is worse in the CNN Tok2Vec based model, but it is preferred a worse performance in the neutral class than in the extremes, positive and negative sentiment.

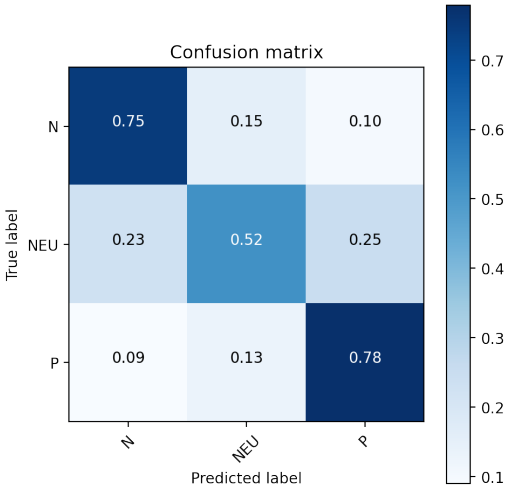
Because of this, and due to having a processing time 5.5 times better, the CNN model based on Tok2Vec word embeddings has been chosen as the model to deploy over the ensemble and the CNN based in transformers.



(a) CNN - Tok2Vec



(b) Ensemble



(c) CNN - Transformers

Figure 4.7: Normalized Confusion Matrices Comparison

CHAPTER 5

Case study

5.1 Introduction

In this chapter, the functionalities created and described on the previous chapters are shown with different examples of actual use. The case studied is the general use of the dashboard, which provides a section of recent tweets about sustainable mobility. These tweets are accompanied by their category within the sustainable mobility space and recognized named entities.

Alongside the list of tweets, a conglomerate of metrics is shown, including the sentiment of the tweets in a pie chart, a word cloud to represent the most used words, a radar chart showing the categories of the tweets and a chart showing the most repeated named entities. These charts enable the user to filter through the different insights by clicking on them one by one. The user can also make use of a search bar to filter through terms. These filters can be concatenated to look for very specific tweets, as long as their topic is sustainable mobility.

In the following sections, examples of actual use of the dashboard will be provided, next to different demonstrations of the charts functionalities as filters.

5.2 Dashboard

In Figure 5.1 the dashboard with all of the components that make it up is shown. There are three differentiated parts:

- **Navbar:** Located at the top. Shows the logo at the left and a lens icon at the right. When the lens icon is clicked, a search bar to look for tweets containing specific terms shows up.

- **Timeline:** Located at the right half. Presents a scrollable list of tweets about sustainable mobility. The tweet element contains the text of the tweet itself alongside author data, time published, and a link to the original tweet (This is the blue Twitter logo located at the top-right corner of each tweet). It also contains the named entities found alongside the category of the tweet. This list will change depending on the filters applied.

- **Charts:** Located at the left half. At the top, it shows a count of the tweets selected at the moment. If any filters are active they are also shown (See Figure 5.2). Here, the word cloud, sentiment pie chart, categories radar chart and entities chart are shown. These graphs will be further explained in the following sections.

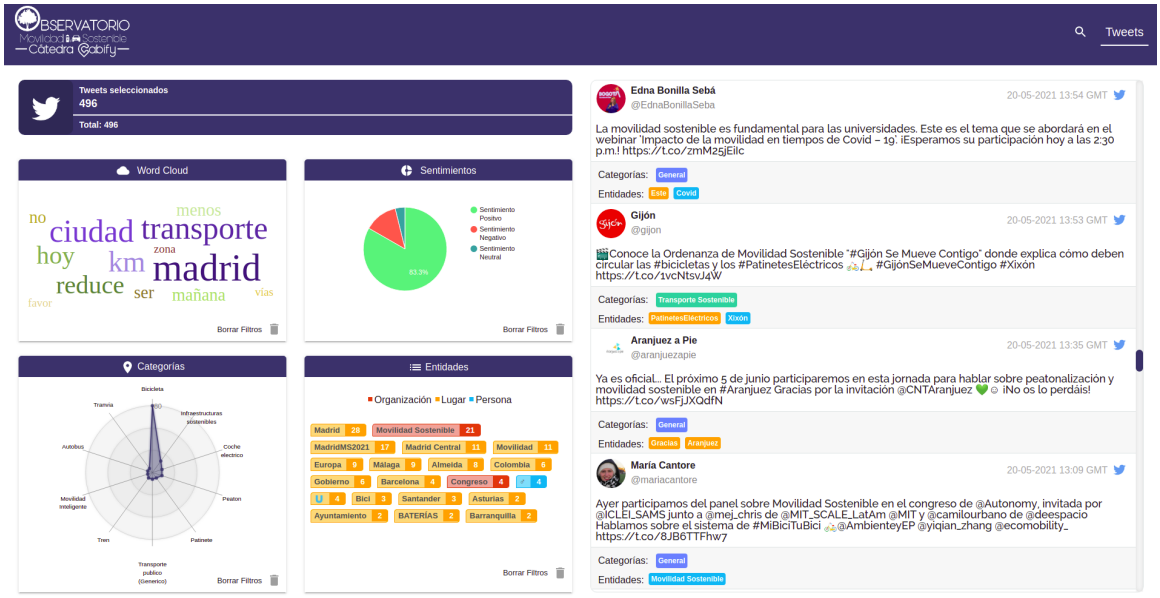


Figure 5.1: Dashboard

5.3 Sentiments Chart

The sentiment pie chart is located at the top-right corner of the charts section. It shows the percentage of tweets by sentiment and changes dynamically according to the filters applied. The sections of the pie chart are color-coded following this schema:

- **Green:** Positive Sentiment
- **Blue:** Neutral Sentiment
- **Red:** Negative Sentiment

Figure 5.1 shows that more than 80% of the tweets have a positive sentiment. This is due to most of them belonging to institutional accounts, which wouldn't show a negative sentiment.

This component also enables the user to filter by the sentiment of the tweet. From Figure 5.2 to Figure 5.4 tweets are filtered by positive, negative and neutral sentiment. This is done by clicking the section of the pie chart belonging to the sentiment desired or the text located at the right where the sentiment is described.

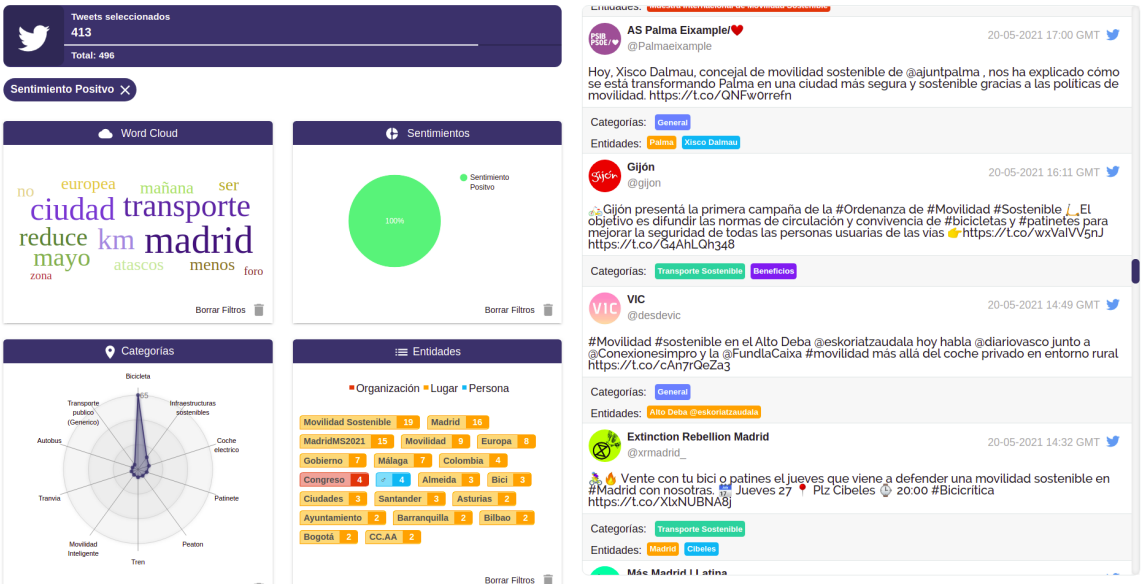


Figure 5.2: Filtering tweets by positive sentiment

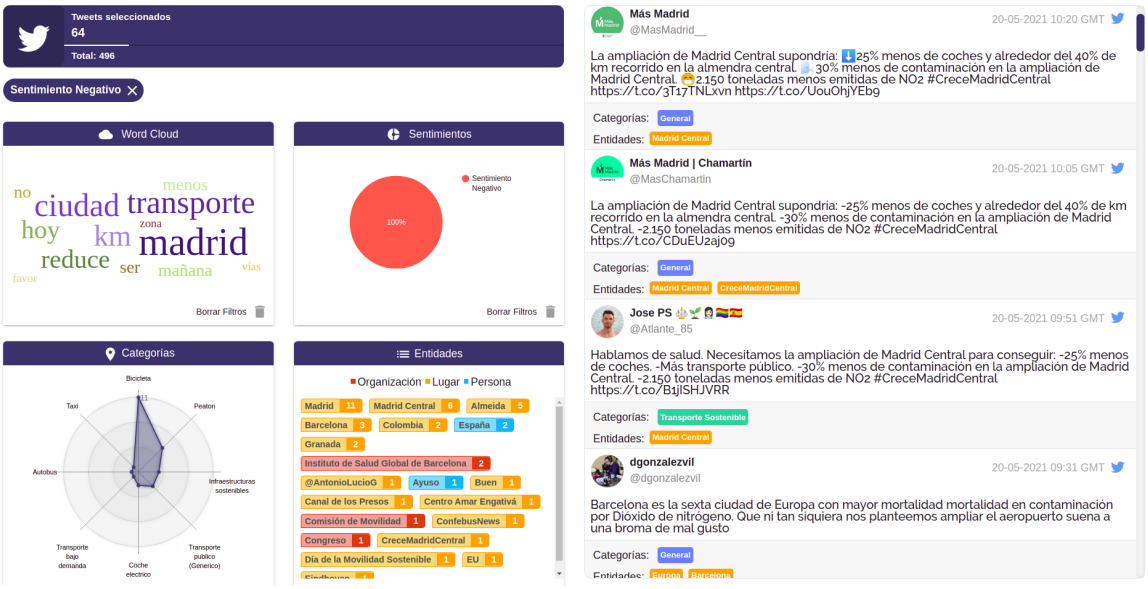


Figure 5.3: Filtering tweets by negative sentiment

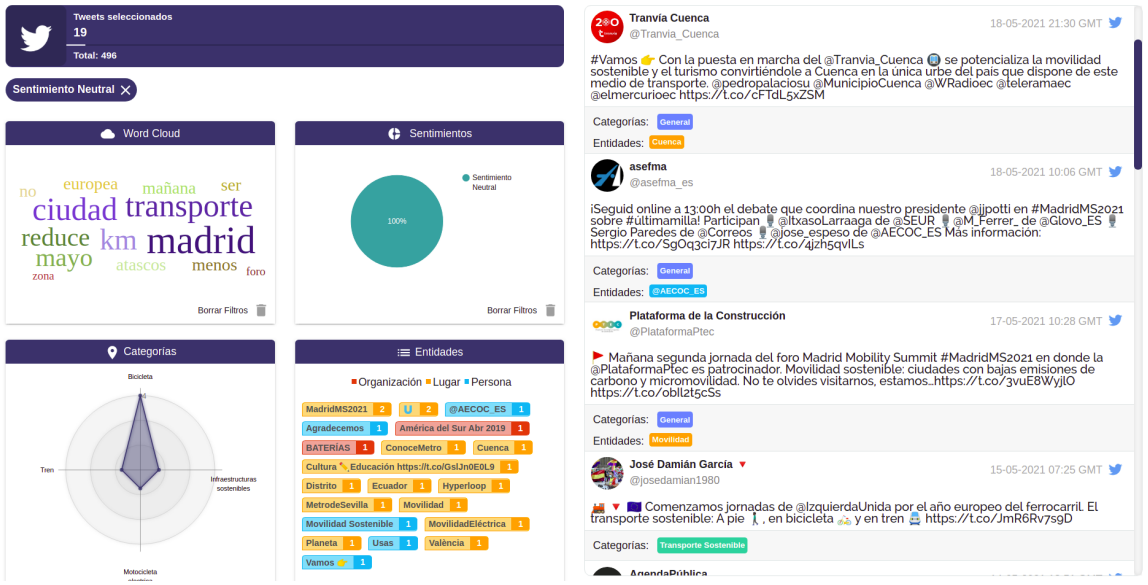


Figure 5.4: Filtering tweets by neutral sentiment

5.4 Word Cloud

The word cloud is located at the top-left corner of the dashboard's charts section. It shows the most repeated words among all the tweets. It does not change dynamically accordingly to the filters applied. However, it enables the user to apply those words as filters by clicking them (Figure 5.5). As seen on Figure 5.6, more than one word can be selected.

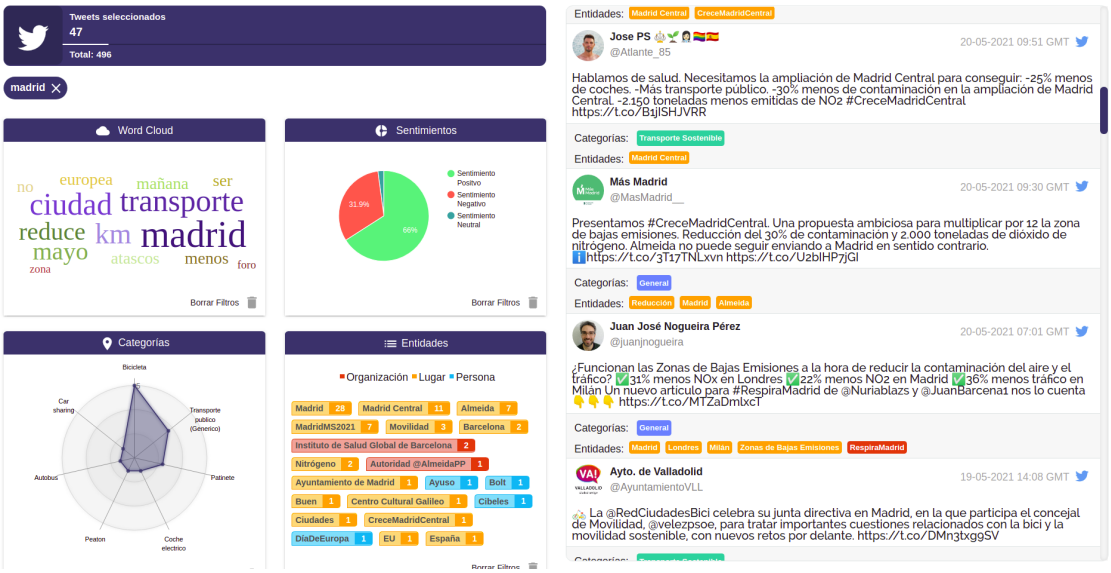


Figure 5.5: Filtering by clicking one word

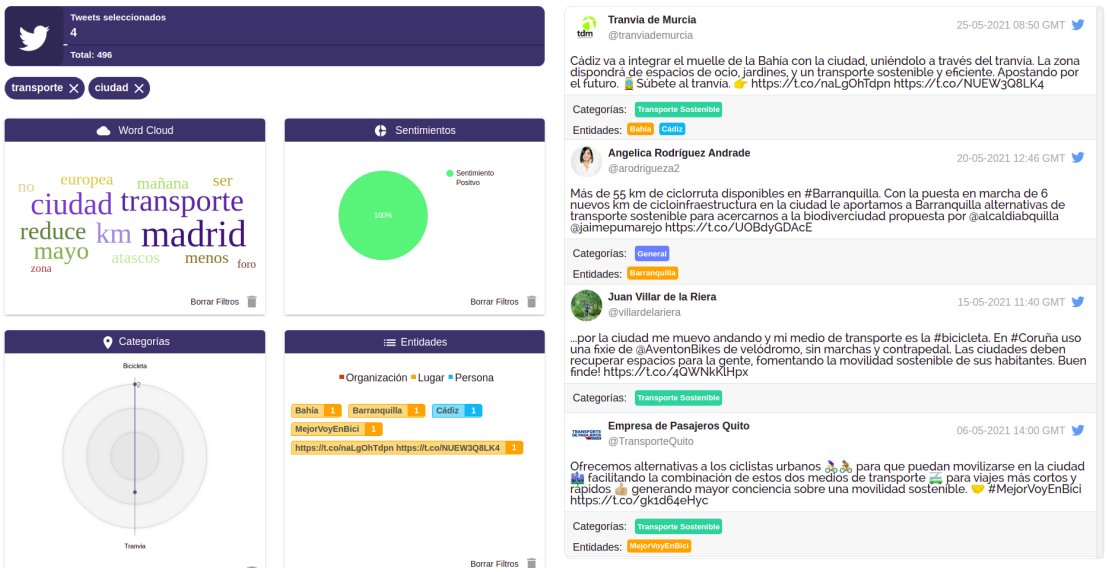


Figure 5.6: Filtering by clicking two words

5.5 Entities Chart

This graph shows the most repeated named entities among the tweets currently selected. It changes dynamically according to the filters in place. Also, by clicking the named entities in the component, the filter for tweets containing those entities is activated (Figure 5.7). As seen on Figure 5.8, more than one entity can be selected.

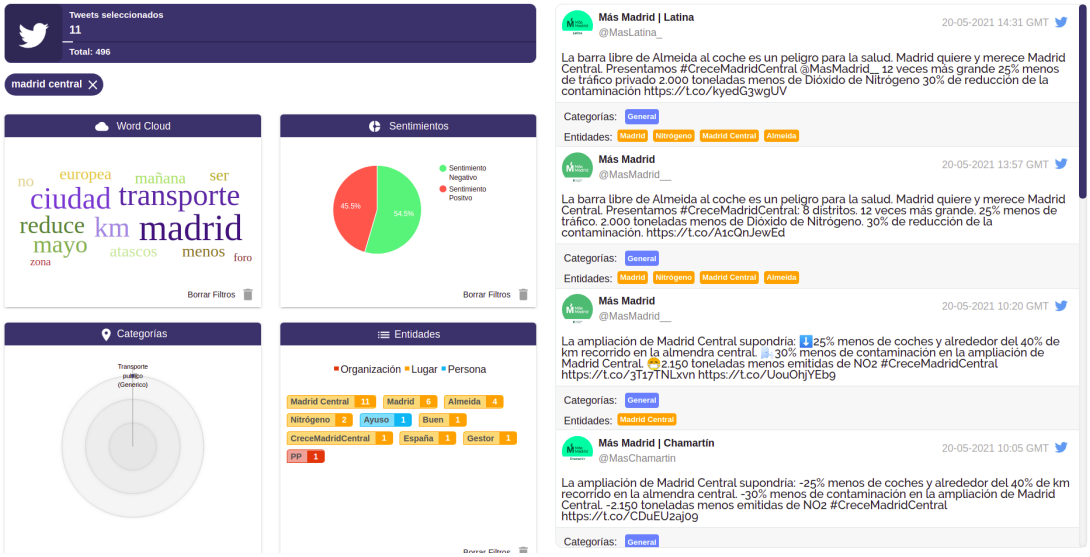


Figure 5.7: Filtering by clicking one entity

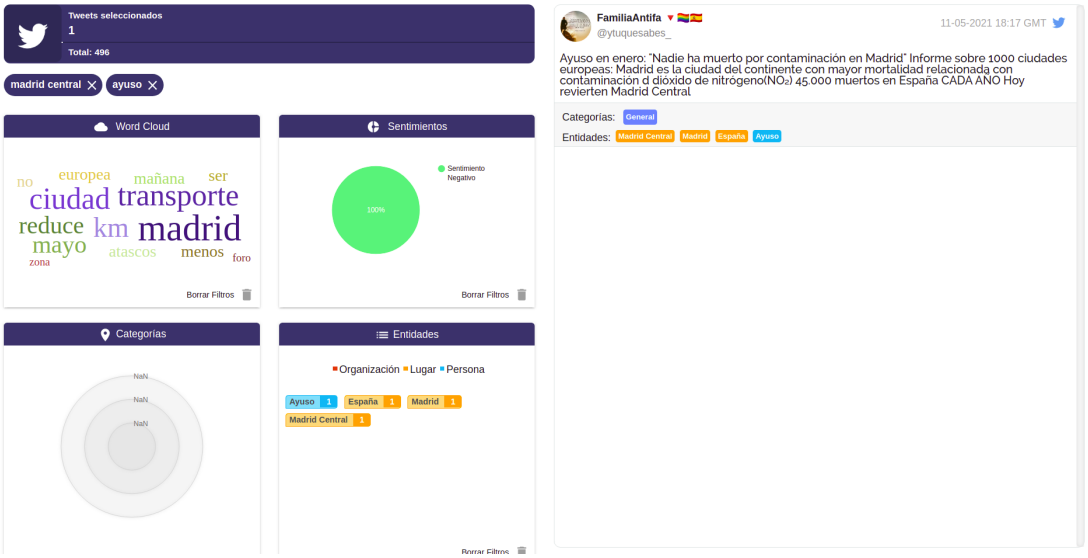


Figure 5.8: Filtering by clicking two entities

5.6 Categories Chart

The categories chart counts how many tweets belong to a given category within the sustainable mobility space. By clicking the categories in the component, the filter for tweets belonging to the selected category is activated (Figure 5.9). Figure 5.10 also shows how more than one category can be selected.

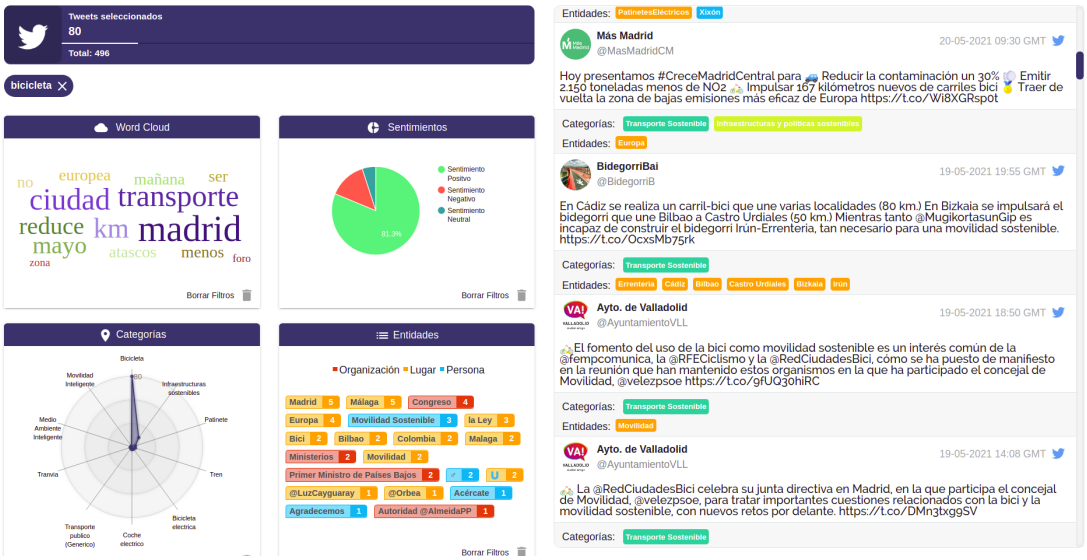


Figure 5.9: Filtering by clicking one category

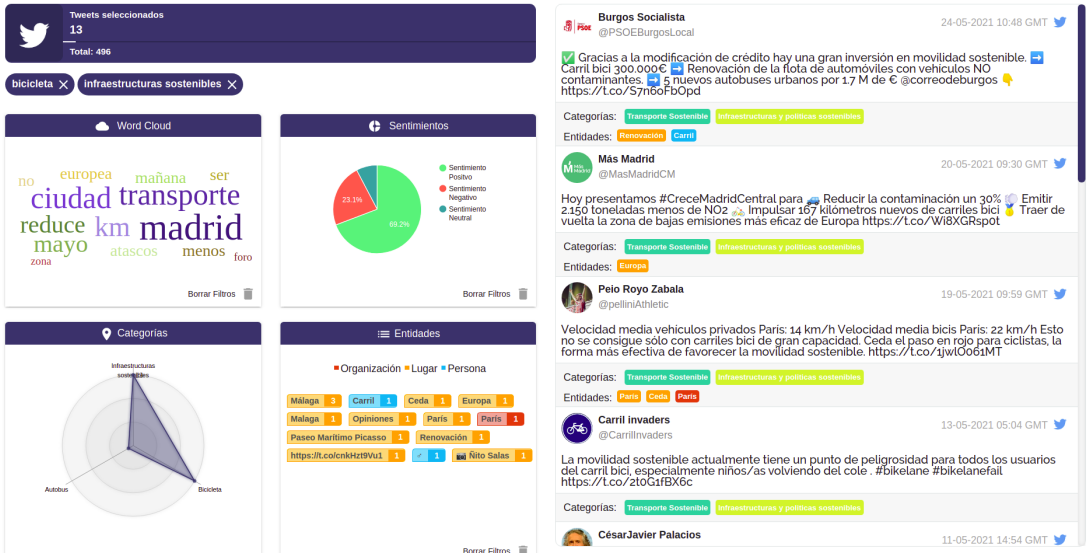


Figure 5.10: Filtering by clicking two categories

5.7 Search Bar

Furthermore, tweets containing specific terms can be looked for using the search bar introducing the term (Figure 5.11) or multiple terms (Figure 5.12).

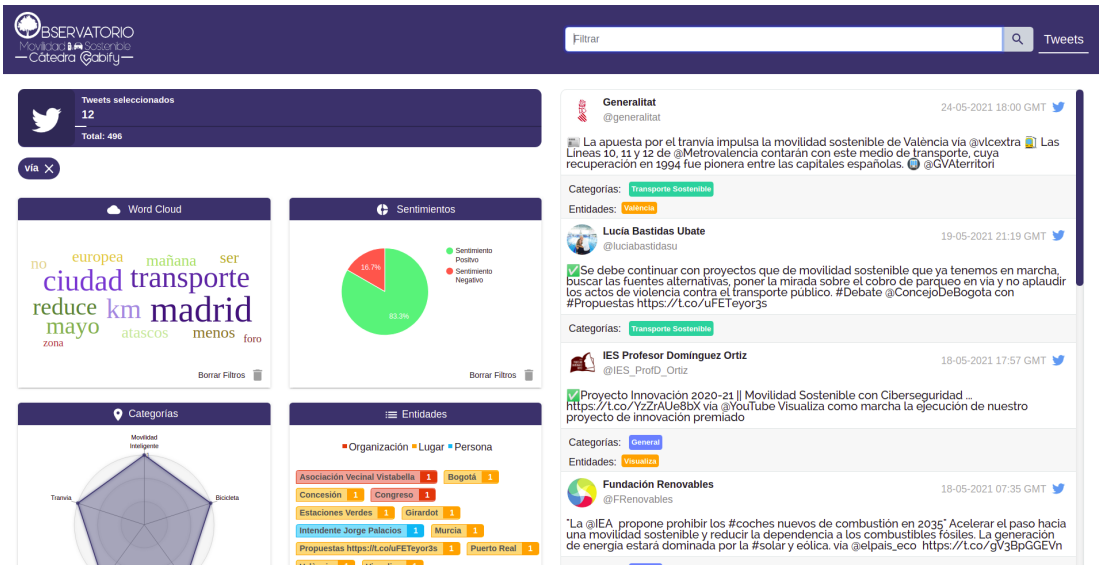


Figure 5.11: Filtering by searching one term

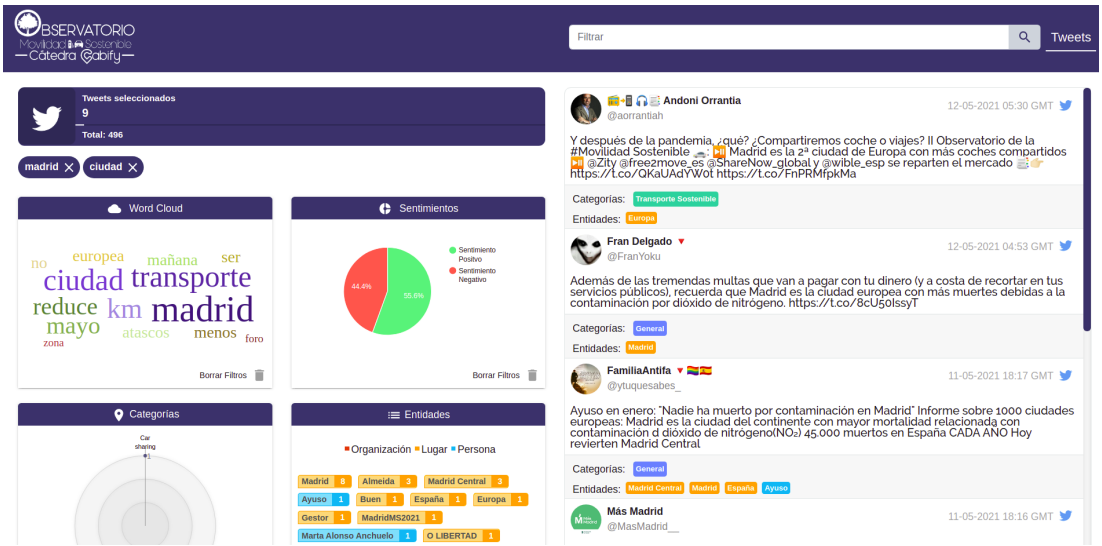


Figure 5.12: Filtering by searching two terms

5.8 Concatenating Filters

The dashboard also enables the user to mix different filters. On Figure 5.13 the sentiment filter and the category filter is used at the same time. On Figure 5.14 the search bar is used alongside the sentiment filters. All combinations are valid. However, concatenating too many filters could lead to no results, which is shown in Figure 5.15.

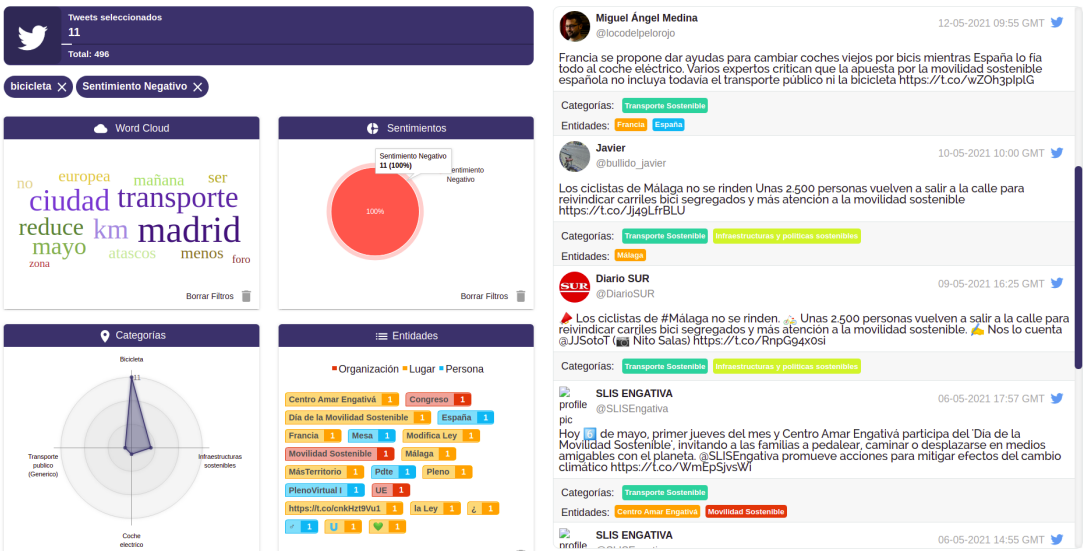


Figure 5.13: Filtering by sentiment and category

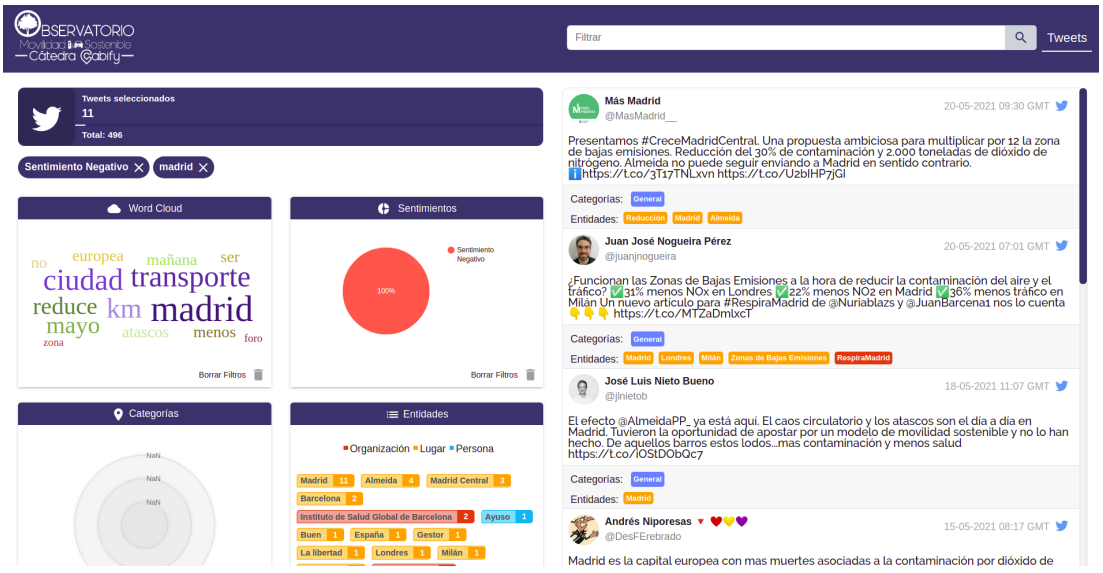


Figure 5.14: Filtering by sentiment and term

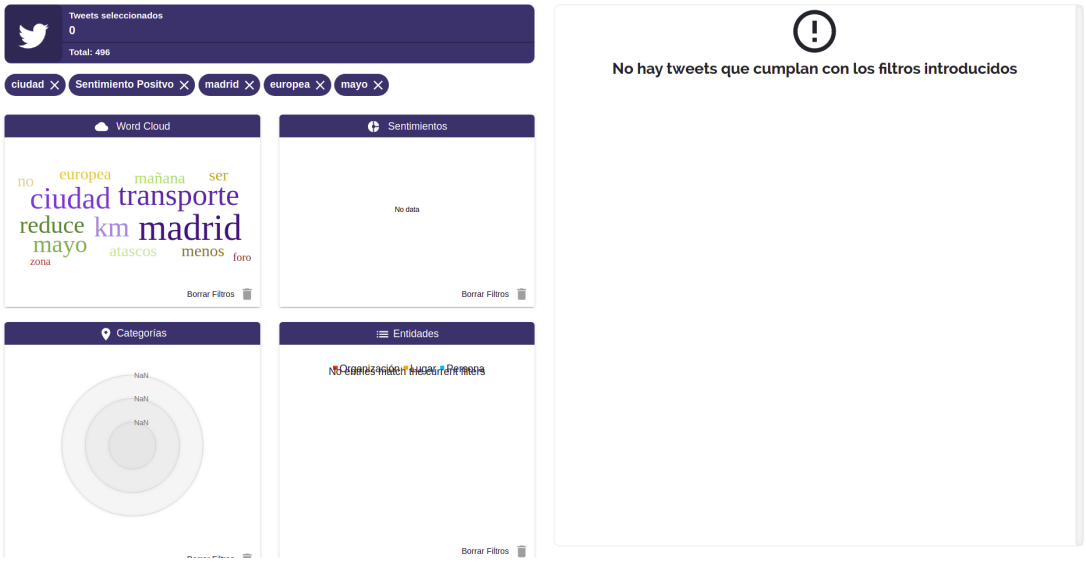


Figure 5.15: No results screen

CHAPTER 6

Conclusions and future work

In this chapter the conclusions extracted from this project, and the thoughts about future work will be described.

6.1 Conclusions

The main purpose of this project was to design and develop a sustainable mobility social trends tracker.

In order to do this, Twitter has been selected as the social network to study. Tweets have been retrieved though the Twitter API. Later, they are processed by three different NLP models through the use of Senpy and enriched following a Linked Data schema. To store these results, the document-oriented database Elasicsearch has been used.

One of the models that the data goes through is a sentiment analysis model created for the purpose of this project. It detects if the analyzed text has a positive, neutral or negative sentiment. It has scored an average F1-Score of 0.76. Finally, the tweets alongside charts of the analyzed data are shown in a dashboard developed thanks to the framework Polymer.

6.2 Achieved Goals

The goals achieved for this project are:

- **Creation of sentiment model**: An efficient NLP sentiment model to distinguish if a Spanish Tweet is positive, neutral or negative has been developed and successfully deployed.
- **Development of dashboard**: A complete, full-functioning dashboard to track sustainable mobility social trends on the social network Twitter has been created.

6.3 Problems Faced

During the course of this project, the main problems encountered have been:

- **Luigi**: The way Luigi functions is a little bit different from other orchestrators, since it has an "upside-down" structure. This means that tasks are defined in the reverse order that they are meant to be executed. Not understanding this at the beginning lead to constant system crashes and failed pipelines.
- **spaCy 3**: The version 3.0 of NLP library spaCy was released on February 2021, roughly one month before the model creation began. Because of this, few resources can be found on the web and getting around on how to use it has been hard.

  Although the use of previous versions was considered, spaCy 3 offered really interesting possibilities like the use of transformer-based token-to-vector models. At the end, the basic functioning mechanism was understood and a well-functioning sentiment model has been provided.
- **Polymer**: Polymer is an old web component framework that is in maintenance mode and no longer used by the community. Because of this, when some of the problems were encountered it was hard to find solutions right away and a lot of debugging was required any time errors appeared.

6.4 Future Work

The possible next steps to improve the dashboard are:

- **Migrate from Polymer:** Since 2019, Polymer is in maintenance mode by the Google development team. In fact, in the official web page [1], the use of lit-element is recommended. Lit-element is a simple base class for creating fast, lightweight web components that work in any web page with any framework.

  Although Polymer hasn't been officially deprecated yet, it is a matter of time until it happens, so migration of the web components to another source should be considered. Frameworks like React might be an option because of their wide adoption and the possibilities they offer.

- **Migrate from Bower:** Bower is a package manager that has been deprecated. Nowadays, there are many alternatives that cover all Bower functionalities alongside new ones. Yarn or npm could be considered.

- **Make dynamic word cloud**: The word cloud component, unlike the rest of the components that make up the dashboard, is not dynamic. This is because all the data processing is done in the front-end and iterating through all the data, counting words and showing them was too resource-consuming. Because of this, it was opted to leave it as a static component. This could be fixed by providing a back-end side that handles and processes data, maybe by adding a new container to the actual structure.

- **Use Kubernetes as orchestration tool:** Docker compose serves its purposes. However, it has a big limitation which is that it runs on a single host. The Kubernetes engine can use multiple nodes that can be changed dynamically, which is a great advantage.

Aside from changing the underlying technologies of already existing features, two completely new functionalities are proposed to continue the development of the Cabify Observatory for Sustainable Mobility:

- **Scientific Articles**: A new and valuable source of data for the observatory could be scientific articles. These articles would provide further insights regarding the latest advancements in the sustainable mobility field. However, new models will have to be developed, since the already created models work for Spanish texts, and scientific articles are mostly written in English.

[1] `https://polymer-library.polymer-project.org/`

- **Event Detection:** To boost the features of the tweet and news trackers, a cross-referencing functionality could be added. This way, when important news about sustainable mobility come up, a system to track how people react to those news on Twitter would provide valuable information about their interactions and feelings about the news.

APPENDIX A

Impact of this project

The purpose of this appendix is to explain the possible implications from a social, environmental and ethical point of view.

A.1 Environmental Impact

As of today, the environmental impact of the project is almost non-existent, since only the electrical consumption of the computer used to create the dashboard is to take into account. However, shall it be deployed as a Software as a Service, the total consumption of energy that the dashboard produces will vary according to the deployment performed. It is also important to take into account the provenance of said energy since it's not the same using electricity produced with renewable energies rather than electricity produced by the burning of coal or oil.

The more redundancy the different components that make up the dashboard service, from the necessary servers to databases, the more consumption it will have thus having a bigger environmental impact. This could also be countered developing activities to reach carbon neutrality such as planting trees.

A.2 Social Impact

From a social point of view, this project has a great impact, since its only task is to analyze the human behaviour related to sustainable mobility in the social network Twitter.

It can be used to keep track of how the Spanish-speaking society feels about new sustainable mobility laws and measures as well as how politicians and organizations promote those measures. Furthermore, the how and who is also a crucial task and the dashboard takes care of it so any user can use it.

A.3 Ethical Implications

The main concern regarding ethical implications was to carry the work being compliant to the European Union's General Data Protection Regulation (GDPR) [82]. Since Tweets can be traced to their creator, they are considered personal data. However, the GDPR enables personal data to be used for research & non-commercial purposes, being this project the case.

APPENDIX B

Economic budget

The purpose of this appendix is to detail the budget for the project, going from the structure followed to carry it out to all the resources used.

B.1 Project Structure

The project structure is shown in the following Gantt chart:

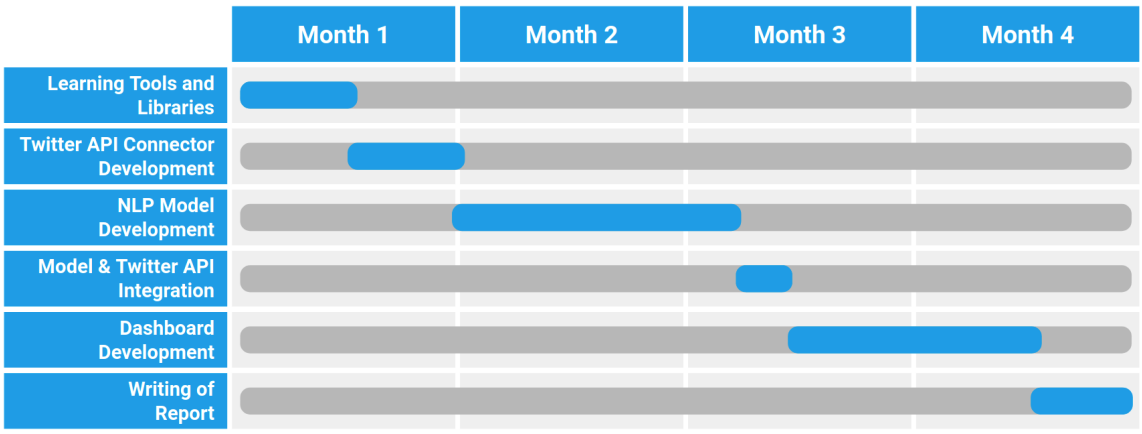


Figure B.1: Project's Structure Gantt's Chart

B.2 Physical resources

To carry out this project the resources that have been used are:

- **Software**: Regarding this matter, only open source software or free proprietary software has been used. Therefore, there is no cost associated to the software for this project.
- **Hardware**: The hardware used is:
  - Laptop ASUS F541U, with the following characteristics:
    - CPU: Intel Core i7-6500U
    - RAM: 16 GB
    - HDD: 1 TB
    - SDD: 500 GB
  - Custom Server, with the following characteristics:
    - CPU: Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz
    - RAM: 128 GB
    - NAS Node: 12 TB
    - SDD: 1 TB

The total cost of the laptop computer, since the SDD has been an enhancement after the purchase is an approximate of **750€**. The cost of the server is around 10.000€, but is shared among different people, so the total computed cost is around **300€**.

B.3 Human Resources

Regarding the cost related to human resources, only one person has been needed to complete the project during a 4 month period. Since it's been developed in an internship, the cost has been 550€ per month.

With this information the total cost related to human resources is: **2.500€**.

B.4 Conclusion

The total cost of the project is 3.550€ and has a total duration of four months.

APPENDIX C

Sustainable Mobility Taxonomy

In this appendix, the taxonomy used to name concepts within the sustainable mobility space is presented.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#">

<skos:ConceptScheme rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Taxonomia_Movilidad_Sostenible">
<skos:prefLabel xml:lang="es">Taxonomia sobre movilidad sostenible</skos:
    prefLabel>
<skos:narrower rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Transporte_Sostenible"/>
</skos:ConceptScheme>

<skos:ConceptScheme rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Transporte_Sostenible">
<skos:prefLabel xml:lang="es">Transporte Sostenible</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Taxonomia_Movilidad_Sostenible"/>
</skos:ConceptScheme>
```

```
<skos:ConceptScheme rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Transporte_Sostenible">
<skos:prefLabel xml:lang="es">Medios de Transporte Sostenible</skos:
    prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Transporte_Sostenible"/>
</skos:ConceptScheme>

<skos:ConceptScheme rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Privados">
<skos:prefLabel xml:lang="es">Medios de Transporte Privado</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Transporte_Sostenible"/>
</skos:ConceptScheme>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Patinete">
<skos:prefLabel xml:lang="es">Patinete</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Privados"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Coche_Electrico">
<skos:prefLabel xml:lang="es">Coche electrico</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Privados"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Bicicleta">
<skos:prefLabel xml:lang="es">Bicicleta</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Privados"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Bicicleta_Electrica">
<skos:prefLabel xml:lang="es">Bicicleta electrica</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Privados"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/Peaton
```

```
    ">
<skos:prefLabel xml:lang="es">Peaton</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Privados"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Avion_Electrico">
<skos:prefLabel xml:lang="es">Avion electrico</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Privados"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Motocicleta_Electrica">
<skos:prefLabel xml:lang="es">Motocicleta Electrica</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Privados"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Transporte_Bajo_Demanda">
<skos:prefLabel xml:lang="es">Transporte Bajo Demanda</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Privados"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Coche_Autonomo">
<skos:prefLabel xml:lang="es">Coche Autonomo</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Privados"/>
</skos:Concept>

<skos:ConceptScheme rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Publicos">
<skos:prefLabel xml:lang="es">Medios de Transporte Publico</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Transporte_Sostenible"/>
</skos:ConceptScheme>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Autobus">
<skos:prefLabel xml:lang="es">Autobus</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
```

```
    Medios_Publicos"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/Metro
    ">
<skos:prefLabel xml:lang="es">Metro</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Publicos"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Metro_Ligero">
<skos:prefLabel xml:lang="es">Metro Ligero</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Publicos"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/Tren">
<skos:prefLabel xml:lang="es">Tren</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Publicos"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/Taxi">
<skos:prefLabel xml:lang="es">Taxi</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Publicos"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Tranvia">
<skos:prefLabel xml:lang="es">Tranvia</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medios_Publicos"/>
</skos:Concept>

<skos:ConceptScheme rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Motores_Sostenibles_-_Alternativos">
<skos:prefLabel xml:lang="es">Motores Sostenibles - Alternativos</skos:
    prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Transporte_Sostenible"/>
</skos:ConceptScheme>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
```

```
    Motores_Hibridos">
<skos:prefLabel xml:lang="es">Motores Hibridos</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Motores_Sostenibles_-_Alternativos"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Motores_Electricos">
<skos:prefLabel xml:lang="es">Motores Electricos</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Motores_Sostenibles_-_Alternativos"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Motores_BiFuel">
<skos:prefLabel xml:lang="es">Motores bi-fuel</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Motores_Sostenibles_-_Alternativos"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Motores_Hidrogeno">
<skos:prefLabel xml:lang="es">Motores de Hidrogeno</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Motores_Sostenibles_-_Alternativos"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Biocombustible">
<skos:prefLabel xml:lang="es">Biocombustible</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Motores_Sostenibles_-_Alternativos"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Motores_Metano">
<skos:prefLabel xml:lang="es">Motores de Metano</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Motores_Sostenibles_-_Alternativos"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Motores_Gas_Natural">
<skos:prefLabel xml:lang="es">Motores de Gas Natural</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
```

```
    Motores_Sostenibles_-_Alternativos"/>
</skos:Concept>

<skos:ConceptScheme rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Movilidad_Compartida">
<skos:prefLabel xml:lang="es">Movilidad Compartida</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Transporte_Sostenible"/>
</skos:ConceptScheme>

<skos:ConceptScheme rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    RideSharing">
<skos:prefLabel xml:lang="es">Viaje compartido - Ridesharing</skos:prefLabel
    >
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Movilidad_Compartida"/>
</skos:ConceptScheme>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Alquiler_Vehiculos_(Generico)">
<skos:prefLabel xml:lang="es">Alquiler de vehiculos (Generico)</skos:
    prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    RideSharing"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Bus_Pooling">
<skos:prefLabel xml:lang="es">Bus pooling - Alquiler de autobuses</skos:
    prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    RideSharing"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Car_Pooling">
<skos:prefLabel xml:lang="es">Car pooling - Alquiler de coches</skos:
    prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    RideSharing"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Van_Pooling">
<skos:prefLabel xml:lang="es">Van pooling - Alquiler de furgonetas</skos:
```

```
    prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    RideSharing"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Moto_Pooling">
<skos:prefLabel xml:lang="es">Moto pooling - Alquiler de motos</skos:
    prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    RideSharing"/>
</skos:Concept>

<skos:ConceptScheme rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Vehiculo_Compartido">
<skos:prefLabel xml:lang="es">Vehiculo compartido</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Movilidad_Compartida"/>
</skos:ConceptScheme>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Car_Sharing">
<skos:prefLabel xml:lang="es">Car sharing</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Vehiculo_Compartido"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Bike_Sharing">
<skos:prefLabel xml:lang="es">Bike sharing</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Vehiculo_Compartido"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Moto_Sharing">
<skos:prefLabel xml:lang="es">Moto sharing</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Vehiculo_Compartido"/>
</skos:Concept>

<skos:ConceptScheme rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Beneficios">
<skos:prefLabel xml:lang="es">Beneficios</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
```

```
    Taxonomia_Movilidad_Sostenible"/>
</skos:ConceptScheme>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Economia_Inteligente">
<skos:prefLabel xml:lang="es">Economia Inteligente</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Beneficios"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Gobierno_Inteligente">
<skos:prefLabel xml:lang="es">Gobierno Inteligente</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Beneficios"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Movilidad_Inteligente">
<skos:prefLabel xml:lang="es">Movilidad Inteligente</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Beneficios"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Medio_Ambiente_Inteligente">
<skos:prefLabel xml:lang="es">Medio Ambiente Inteligente</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Beneficios"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Vida_Inteligente">
<skos:prefLabel xml:lang="es">Vida Inteligente</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Beneficios"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Poblacion_Inteligente">
<skos:prefLabel xml:lang="es">Poblacion Inteligente</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Beneficios"/>
</skos:Concept>
```

```
<skos:ConceptScheme rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Tecnologia">
<skos:prefLabel xml:lang="es">Tecnologia</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Taxonomia_Movilidad_Sostenible"/>
</skos:ConceptScheme>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Herramientas_TIC">
<skos:prefLabel xml:lang="es">Herramientas TIC"</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Tecnologia"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Tecnicas_Computacionales">
<skos:prefLabel xml:lang="es">Tecnicas Computacionales"</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Tecnologia"/>
</skos:Concept>

<skos:ConceptScheme rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Infraestructuras_Y_Politicas_Sostenibles">
<skos:prefLabel xml:lang="es">Infraestructuras y politicas sostenibles</skos
    :prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Taxonomia_Movilidad_Sostenible"/>
</skos:ConceptScheme>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Infraestructuras_Sostenibles">
<skos:prefLabel xml:lang="es">Infraestructuras Sostenibles</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Infraestructuras_Y_Politicas_Sostenibles"/>
</skos:Concept>

<skos:Concept rdf:about="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Politicas_Sostenibles">
<skos:prefLabel xml:lang="es">Politicas Sostenibles</skos:prefLabel>
<skos:broader rdf:resource="https://www.gsi.upm.es/es/ontologies/mobo/ns/
    Infraestructuras_Y_Politicas_Sostenibles"/>
</skos:Concept>

</rdf:RDF>
```

APPENDIX D

TASS Corpus

In this appendix, the complete XML Schema (XSD) of the TASS corpus is provided on Figure D.1. In addition, an example of a tweet with every sentiment categorization is provided:

- Strong Positive (P+) in Figure D.2
- Positive (P) in Figure D.3
- Neutral (NEU) in Figure D.4
- No sentiment (NONE) in Figure D.5
- Negative (N) in Figure D.6
- Strong Negative (N+) in Figure D.7

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tt="http://www.daedalus.es/TASS/twits"
 targetNamespace="http://www.daedalus.es/TASS/twits" elementFormDefault="qualified">
  <xsd:simpleType name="enumValuePolarity">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="NONE"/>
      <xsd:enumeration value="N+"/>
      <xsd:enumeration value="N"/>
      <xsd:enumeration value="NEU"/>
      <xsd:enumeration value="P"/>
      <xsd:enumeration value="P+"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="enumTypePolarity">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="AGREEMENT"/>
      <xsd:enumeration value="DISAGREEMENT"/>
    </xsd:restriction>
  </xsd:simpleType>
  <!--  typePolatity complexType definition  -->
  <xsd:complexType name="typePolarity">
    <xsd:sequence>
      <xsd:element name="entity" type="xsd:string" minOccurs="0"/>
      <xsd:element name="value" type="tt:enumValuePolarity"/>
      <xsd:element name="type" type="tt:enumTypePolarity" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <!--  sentiments definition -->
  <xsd:element name="sentiments">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="polarity" type="tt:typePolarity" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <!--  topics definition -->
  <xsd:element name="topics">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="topic" type="xsd:string" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="twit">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="twitid" type="xsd:nonNegativeInteger"/>
        <xsd:element name="user" type="xsd:string"/>
        <xsd:element name="content" type="xsd:string"/>
        <xsd:element name="date" type="xsd:string"/>
        <xsd:element name="lang" type="xsd:language"/>
        <xsd:element ref="tt:sentiments" minOccurs="0"/>
        <xsd:element ref="tt:topics" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <!--  users definition -->
  <xsd:element name="twits">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="tt:twit" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:key name="keyTwits">
      <xsd:selector xpath=".//tt:twit"/>
      <xsd:field xpath="tt:twitid"/>
    </xsd:key>
  </xsd:element>
</xsd:schema>
```

Figure D.1: TASS XML Structure [13]

```
<tweet>
  <tweetid>171563033330585601</tweetid>
  <user>SSantiagosegura</user>
  <content>
    <![CDATA[ Increíble audiencia para la gala de los Goya y un sobresaliente para su
    "pedazo" de presentadora mrs Eva H! Bravo!#hayquelidiaresetoro ]]>
  </content>
  <date>2012-02-20T12:53:09</date>
  <lang>es</lang>
  <sentiments>
    <polarity>
      <value>P+</value>
      <type>AGREEMENT</type>
    </polarity>
    <polarity>
      <entity>los_Goya</entity>
      <value>P+</value>
      <type>AGREEMENT</type>
    </polarity>
    <polarity>
      <entity>Eva_H</entity>
      <value>P+</value>
      <type>AGREEMENT</type>
    </polarity>
  </sentiments>
  <topics>
    <topic>cine</topic>
  </topics>
</tweet>
```

Figure D.2: Strong Positive (P+) Tweet [13]

```
<tweet>
  <tweetid>179473131809931264</tweetid>
  <user>Lissavetzky_M</user>
  <content>
    <![CDATA[ Hoy visitaremos la Nave Boetticher (Villaverde) para hablar sobre nuevas tecnologías en Madrid. ]]>
  </content>
  <date>2012-03-13T08:45:03</date>
  <lang>es</lang>
  <sentiments>
    <polarity>
      <value>P</value>
      <type>AGREEMENT</type>
    </polarity>
  </sentiments>
  <topics>
    <topic>otros</topic>
    <topic>tecnología</topic>
  </topics>
</tweet>
```

Figure D.3: Positive (P) Tweet [13]

```
<tweet>
  <tweetid>143002689222082560</tweetid>
  <user>Albert_Rivera</user>
  <content>
    <![CDATA[ RT@FranHervias:UPyD pacta con Cascos para tener grupo.A pesar d que en 2008 decían que no
    era lícito este tipo d pactos http://t.co/fG5cVwM9 ]]>
  </content>
  <date>2011-12-03T17:24:32</date>
  <lang>es</lang>
  <sentiments>
    <polarity>
      <value>NEU</value>
      <type>DISAGREEMENT</type>
    </polarity>
    <polarity>
      <entity>UPyD</entity>
      <value>P</value>
      <type>AGREEMENT</type>
    </polarity>
    <polarity>
      <entity>Cascos</entity>
      <value>P</value>
      <type>AGREEMENT</type>
    </polarity>
  </sentiments>
  <topics>
    <topic>política</topic>
  </topics>
</tweet>
```

Figure D.4: Neutral (NEU) Tweet [13]

```
<tweet>
  <tweetid>171516739316756480</tweetid>
  <user>LFRudi</user>
  <content>
    <![CDATA[ En Sesión Plenaria de las Cortes de Aragón. #Debate y #votación
    dictamen Comisión Hacienda sobre #Proyecto Ley #Presupuestos 2012 ]]>
  </content>
  <date>2012-02-20T09:49:11</date>
  <lang>es</lang>
  <sentiments>
    <polarity>
      <value>NONE</value>
      <type>AGREEMENT</type>
    </polarity>
  </sentiments>
  <topics>
    <topic>política</topic>
  </topics>
</tweet>
```

Figure D.5: No Sentiment (NONE) Tweet [13]

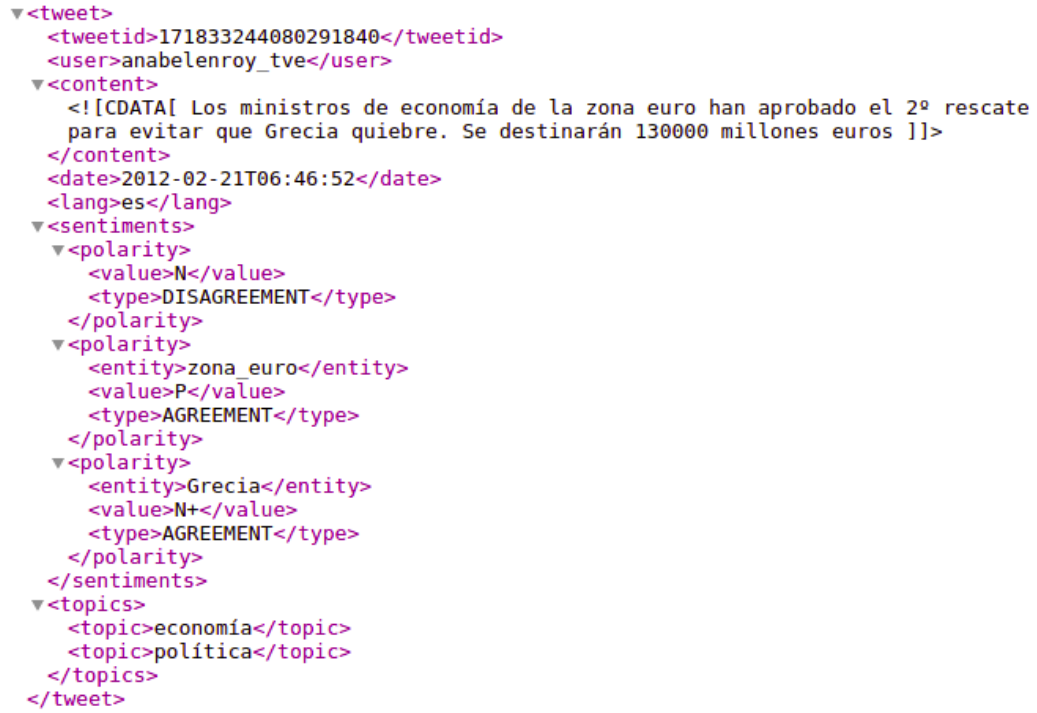


Figure D.6: Negative (N) Tweet [13]

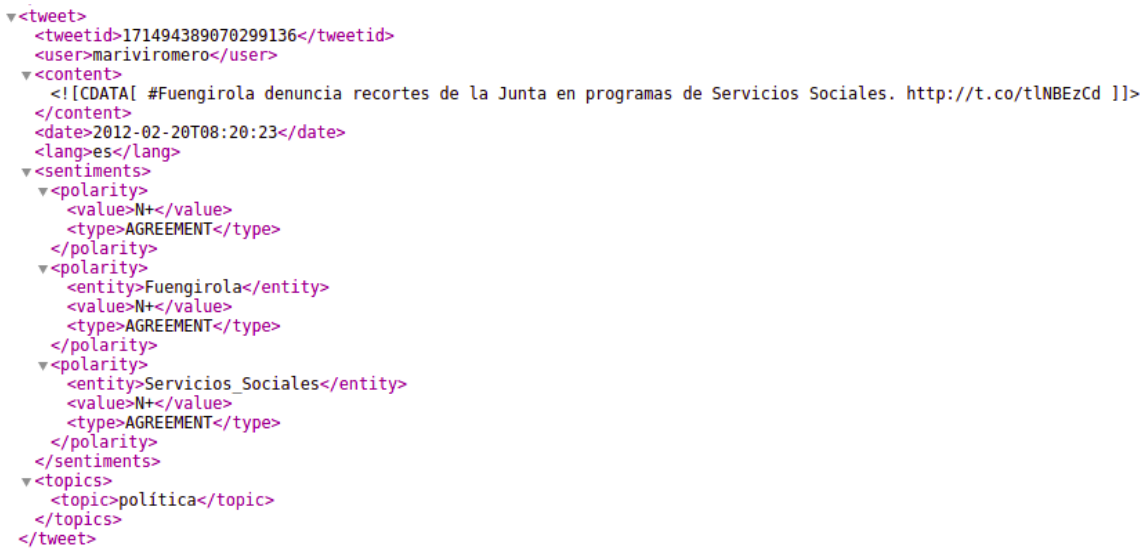


Figure D.7: Very Negative Tweet (N+) [13]

APPENDIX E

spaCy Configurations

In this appendix the configuration files for the three spaCy architectures are showed:

E.1 Ensemble Configuration

```
[paths]
train = "data/train.spacy"
dev = "data/valid.spacy"
vectors = null
init_tok2vec = null

[system]
gpu_allocator = null
seed = 0

[nlp]
lang = "es"
pipeline = ["tok2vec","textcat"]
batch_size = 1000
disabled = []
before_creation = null
```

```
after_creation = null
after_pipeline_creation = null
tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}

[components]

[components.textcat]
factory = "textcat"
threshold = 0.5

[components.textcat.model]
@architectures = "spacy.TextCatEnsemble.v2"
nO = null

[components.textcat.model.linear_model]
@architectures = "spacy.TextCatBOW.v1"
exclusive_classes = true
ngram_size = 1
no_output_layer = false
nO = null

[components.textcat.model.tok2vec]
@architectures = "spacy.Tok2VecListener.v1"
width = ${components.tok2vec.model.encode.width}
upstream = "*"

[components.tok2vec]
factory = "tok2vec"

[components.tok2vec.model]
@architectures = "spacy.Tok2Vec.v2"

[components.tok2vec.model.embed]
@architectures = "spacy.MultiHashEmbed.v1"
width = ${components.tok2vec.model.encode.width}
attrs = ["ORTH","SHAPE"]
rows = [5000,2500]
include_static_vectors = true

[components.tok2vec.model.encode]
@architectures = "spacy.MaxoutWindowEncoder.v2"
width = 256
depth = 8
window_size = 1
maxout_pieces = 3
```

```
[corpora]

[corpora.dev]
@readers = "spacy.Corpus.v1"
path = ${paths.dev}
max_length = 0
gold_preproc = false
limit = 0
augmenter = null

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths.train}
max_length = 2000
gold_preproc = false
limit = 0
augmenter = null

[training]
dev_corpus = "corpora.dev"
train_corpus = "corpora.train"
seed = ${system.seed}
gpu_allocator = ${system.gpu_allocator}
dropout = 0.1
accumulate_gradient = 1
patience = 1600
max_epochs = 0
max_steps = 20000
eval_frequency = 200
frozen_components = []
before_to_disk = null

[training.batcher]
@batchers = "spacy.batch_by_words.v1"
discard_oversize = false
tolerance = 0.2
get_length = null

[training.batcher.size]
@schedules = "compounding.v1"
start = 100
stop = 1000
compound = 1.001
t = 0.0
```

```
[training.logger]
@loggers = "spacy.ConsoleLogger.v1"
progress_bar = false

[training.optimizer]
@optimizers = "Adam.v1"
beta1 = 0.9
beta2 = 0.999
L2_is_weight_decay = true
L2 = 0.01
grad_clip = 1.0
use_averages = false
eps = 0.00000001
learn_rate = 0.001

[training.score_weights]
cats_score_desc = null
cats_micro_p = null
cats_micro_r = null
cats_micro_f = null
cats_macro_p = null
cats_macro_r = null
cats_macro_f = null
cats_macro_auc = null
cats_f_per_type = null
cats_macro_auc_per_type = null
cats_score = 1.0

[pretraining]

[initialize]
vectors = ${paths.vectors}
init_tok2vec = ${paths.init_tok2vec}
vocab_data = null
lookups = null
before_init = null
after_init = null

[initialize.components]

[initialize.tokenizer]
```

E.2 Convolutional Neural Network Configuration

```
[paths]
train = "data/train.spacy"
dev = "data/valid.spacy"
raw = null
init_tok2vec = null
vectors = null

[system]
seed = 0
gpu_allocator = null

[nlp]
lang = "es"
pipeline = ["textcat"]
tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}
disabled = []
before_creation = null
after_creation = null
after_pipeline_creation = null
batch_size = 54413

[components]

[components.textcat]
factory = "textcat_multilabel"
threshold = 0.5

[components.textcat.model]
@architectures = "spacy.TextCatCNN.v1"
exclusive_classes = false
nO = null

[components.textcat.model.tok2vec]
@architectures = "spacy.Tok2Vec.v1"

[components.textcat.model.tok2vec.embed]
@architectures = "spacy.MultiHashEmbed.v1"
width = ${components.textcat.model.tok2vec.encode:width}
rows = [10000,5000,5000,5000]
attrs = ["NORM","PREFIX","SUFFIX","SHAPE"]
include_static_vectors = false
```

```
[components.textcat.model.tok2vec.encode]
@architectures = "spacy.MaxoutWindowEncoder.v1"
width = 96
depth = 4
window_size = 1
maxout_pieces = 3

[corpora]

[corpora.dev]
@readers = "spacy.Corpus.v1"
path = ${paths:dev}
gold_preproc = ${corpora.train.gold_preproc}
max_length = 0
limit = 0
augmenter = null

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths:train}
gold_preproc = false
max_length = 0
limit = 0
augmenter = null

[training]
train_corpus = "corpora.train"
dev_corpus = "corpora.dev"
seed = ${system.seed}
gpu_allocator = ${system.gpu_allocator}
dropout = 0.2
patience = 1600
max_epochs = 0
max_steps = 20000
eval_frequency = 200
accumulate_gradient = 1
frozen_components = []
before_to_disk = null

[training.batcher]
@batchers = "spacy.batch_by_sequence.v1"
size = 32
get_length = null

[training.logger]
```

```
@loggers = "spacy.ConsoleLogger.v1"
progress_bar = false

[training.optimizer]
@optimizers = "Adam.v1"
beta1 = 0.9
beta2 = 0.999
L2_is_weight_decay = true
L2 = 0.01
grad_clip = 1.0
eps = 0.00000001
learn_rate = 0.001
use_averages = true

[training.score_weights]
cats_score_desc = null
cats_micro_p = null
cats_micro_r = null
cats_micro_f = null
cats_macro_p = null
cats_macro_r = null
cats_macro_f = null
cats_macro_auc = null
cats_f_per_type = null
cats_macro_auc_per_type = null
cats_score = 1.0

[pretraining]

[initialize]
vectors = ${paths.vectors}
init_tok2vec = ${paths.init_tok2vec}
vocab_data = null
lookups = null
before_init = null
after_init = null

[initialize.components]

[initialize.tokenizer]
```

E.3 Transformer Configuration

```
[paths]
train = "data/train.spacy"
dev = "data/valid.spacy"
raw = null
init_tok2vec = null
vectors = null

[system]
seed = 0
gpu_allocator = null

[nlp]
lang = "es"
pipeline = ["transformer","textcat"]
tokenizer = {"@tokenizers":"spacy.Tokenizer.v1"}
disabled = []
before_creation = null
after_creation = null
after_pipeline_creation = null
batch_size = 1

[components]

[components.textcat]
factory = "textcat_multilabel"
threshold = 0.5

[components.textcat.model]
@architectures = "spacy.TextCatCNN.v1"
exclusive_classes = false
nO = null

[components.textcat.model.tok2vec]
@architectures = "spacy-transformers.TransformerListener.v1"
grad_factor = 1.0
pooling = {"@layers":"reduce_mean.v1"}
upstream = "*"

[components.transformer]
factory = "transformer"
max_batch_items = 4096
set_extra_annotations = {"@annotation_setters":"spacy-transformers.
```

```
    null_annotation_setter.v1"}

[components.transformer.model]
@architectures = "spacy-transformers.TransformerModel.v1"
name = "dccuchile/bert-base-spanish-wwm-cased"

[components.transformer.model.get_spans]
@span_getters = "spacy-transformers.strided_spans.v1"
window = 128
stride = 96

[components.transformer.model.tokenizer_config]
use_fast = true

[corpora]

[corpora.dev]
@readers = "spacy.Corpus.v1"
path = ${paths.dev}
gold_preproc = ${corpora.train.gold_preproc}
max_length = ${corpora.train.max_length}
limit = 0
augmenter = null

[corpora.train]
@readers = "spacy.Corpus.v1"
path = ${paths:train}
gold_preproc = false
max_length = 500
limit = 0
augmenter = null

[training]
train_corpus = "corpora.train"
dev_corpus = "corpora.dev"
seed = ${system.seed}
gpu_allocator = ${system.gpu_allocator}
patience = 10000
eval_frequency = 400
dropout = 0.1
max_epochs = 0
max_steps = 0
accumulate_gradient = 3
frozen_components = []
before_to_disk = null
```

```
[training.batcher]
@batchers = "spacy.batch_by_sequence.v1"
size = 256
get_length = null

[training.logger]
@loggers = "spacy.ConsoleLogger.v1"
progress_bar = false

[training.optimizer]
@optimizers = "Adam.v1"
beta1 = 0.9
beta2 = 0.999
eps = 0.00000001
L2_is_weight_decay = true
L2 = 0.01
grad_clip = 1.0
use_averages = false

[training.optimizer.learn_rate]
@schedules = "warmup_linear.v1"
warmup_steps = 250
total_steps = 20000
initial_rate = 0.00005

[training.score_weights]
cats_score_desc = null
cats_micro_p = null
cats_micro_r = null
cats_micro_f = null
cats_macro_p = null
cats_macro_r = null
cats_macro_auc = null
cats_f_per_type = null
cats_macro_auc_per_type = null
cats_score = 0.5
cats_macro_f = 0.5

[pretraining]

[initialize]
vectors = ${paths.vectors}
init_tok2vec = ${paths.init_tok2vec}
vocab_data = null
```

```
lookups = null
before_init = null
after_init = null

[initialize.components]

[initialize.tokenizer]
```

APPENDIX F

Acronyms and Abbreviations

The acronyms and abbreviations used throughout the document are presented in this section in order of appearance:

NLP: Natural Language Processing

RDF: Resource Description Framework

XML: Extensible Markup Language

JSON: JavaScript Object Notation

URI: Uniform Resource Identifiers

HTTP: Hypertext Transfer Protocol

SQL: Structured Query Language

CSV: Comma-Separated Values

ML: Machine Learning

API: Application Programming Interface

HTML: HyperText Markup Language

NIF: NLP Interchange Format

YML: YAML Ain't Markup Language

IP: Internet Protocol

REST: Representational State Transfer

URL: Uniform Resource Locator

SEPLN: Sociedad Española para el Procesamiento del Languaje Natural

TASS: Taller de Análisis Semántico en la SEPLN

XSD: XML Schema

CNN: Convolutional Neural Network

GDPR: General Data Protection Regulation

Bibliography

[1] Word embedding: New age text vectorization in nlp. https://medium.com/swlh/word-embedding-new-age-text-vectorization-in-nlp-3a2db1db2f5b. Accessed: 2021-05-24.

[2] A comprehensive guide to convolutional neural networks. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b116 Accessed: 2021-05-24.

[3] Convolutional neural network in natural language processing. https://towardsdatascience.com/convolutional-neural-network-in-natural-language-processi Accessed: 2021-05-24.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[5] Emmanuel Desmontils and Patricia Serrano-Alvarado. Personal linked data: A solution to manage user's privacy on the web. 06 2013.

[6] Wes McKinney et al. Pandas: A foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, 14(9):1–9, 2011.

[7] Python data visualization with matplotlib — part 1 — rizky maulana n — towards data science. https://towardsdatascience.com/visualizations-with-matplotlib-part-1-c9651008b6b8. Accessed: 2021-05-21.

[8] Visualizers · spacy usage documentation. https://spacy.io/usage/visualizers. Accessed: 2021-05-17.

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

[10] Kubernetes vs docker – which one should you use? https://www.nakivo.com/blog/docker-vs-kubernetes/. Accessed: 2021-05-17.

[11] Containerized python development - part 2 - docker blog. https://www.docker.com/blog/containerized-python-development-part-2/. Accessed: 2021-05-17.

[12] What is sefarad? — sefarad 1.0 documentation. https://sefarad.readthedocs.io/en/latest/sefarad.html. Accessed: 2021-05-18.

[13] Tass 2012 - sepln. http://tass.sepln.org/2012/corpus.php. Accessed: 2021-05-22.

[14] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[15] Derek Ruths and Jürgen Pfeffer. Social media for large studies of behavior. *Science*, 346(6213):1063–1064, 2014.

[16] Md Rafiqul Islam, Muhammad Ashad Kabir, Ashir Ahmed, Abu Raihan M Kamal, Hua Wang, and Anwaar Ulhaq. Depression detection from social network data using machine learning techniques. *Health information science and systems*, 6(1):1–12, 2018.

[17] Marco Pennacchiotti and Ana-Maria Popescu. A machine learning approach to twitter user classification. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 5, 2011.

[18] European investment bank (eib). 2020–2021 eib climate survey. https://www.eib.org/en/surveys/climate-survey/3rd-climate-survey/climate-change-and-covid-recovery.htm. Accessed: 2021-05-24.

[19] David Caldevilla-Domínguez, Almudena Barrientos-Báez, and Graciela Padilla-Castillo. Twitter as a tool for citizen education and sustainable cities after covid-19. *Sustainability*, 13(6):3514, 2021.

[20] Hao Wang, Doğan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan. A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In *Proceedings of the ACL 2012 system demonstrations*, pages 115–120, 2012.

[21] Dieter Fensel, Birgit Leiter, and Ioannis Stavrakantonakis. Social media monitoring. *Semantic Technology Institute, Innsbruck*, 16, 2012.

[22] Gobinda G. Chowdhury. Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89, 2003.

[23] Lluís Marquez and Jordi Girona Salgado. Machine learning and natural language processing, 2000.

[24] Thomas G. Dietterich. Machine-learning research. *AI Magazine*, 18(4):97, Dec. 1997.

[25] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Willey & Sons, New Yotk, 1973.

[26] Ronald Christensen. *Log-Linear Models and Logistic Regression*. Springer-Verlag New York, 1997.

[27] Dan H. Moore II. Classification and regression trees, by leo breiman, jerome h. friedman, richard a. olshen, and charles j. stone. brooks/cole publishing, monterey, 1984,358 pages, $27.95. *Cytometry*, 8(5):534–535, 1987.

[28] Ronald L Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987.

[29] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational linguistics*, 21(4):543–565, 1995.

[30] Roni Khardon, Dan Roth, and Leslie G Valiant. Relational learning for nlp using linear threshold elements. In *IJCAI*, volume 99, pages 911–919. Citeseer, 1999.

[31] Ramon Lopez De Mantaras and Enric Plaza. Case-based reasoning: an overview. *AI communications*, 10(1):21–29, 1997.

[32] Marc Moreno Lopez and Jugal Kalita. Deep learning applied to nlp. *arXiv preprint arXiv:1703.03091*, 2017.

[33] David E Golberg. Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, 1989(102):36, 1989.

[34] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.

[35] Grigori Sidorov, Francisco Velasquez, Efstathios Stamatatos, Alexander Gelbukh, and Liliana Chanona-Hernández. Syntactic n-grams as machine learning features for natural language processing. *Expert Systems with Applications*, 41(3):853–860, 2014.

[36] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, 2014.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[38] Google: Bert now used on almost every english query. `https://searchengineland.com/google-bert-used-on-almost-every-english-query-342193`. Accessed: 2021-05-24.

[39] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, August 2019. Association for Computational Linguistics.

[40] Denny Britz. Understanding convolutional neural networks for nlp. *URL: http://www. wildml. com/2015/11/understanding-convolutional-neuralnetworks-for-nlp/(visited on 11/07/2015)*, 2015.

[41] Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.

[42] Ora Lassila, Ralph R Swick, et al. Resource description framework (rdf) model and syntax specification. 1998.

[43] Dan Brickley, Ramanathan V Guha, and Andrew Layman. Resource description framework (rdf) schema specification. 1999.

[44] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

[45] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227. IGI global, 2011.

[46] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data: Principles and state of the art. In *World wide web conference*, volume 1, page 40, 2008.

[47] Ian Horrocks, Peter F Patel-Schneider, and Frank Van Harmelen. From shiq and rdf to owl: The making of a web ontology language. *Journal of web semantics*, 1(1):7–26, 2003.

[48] Ramanathan V Guha, Dan Brickley, and Steve Macbeth. Schema.org: Evolution of structured data on the web: Big data makes common schemas even more necessary. *Queue*, 13(9):10–37, 2015.

[49] John G Breslin, Stefan Decker, Andreas Harth, and Uldis Bojars. Sioc: an approach to connect web-based communities. *International Journal of Web Based Communities*, 2(2):133–142, 2006.

[50] Thomas Baker. Libraries, languages of description, and linked data: a dublin core perspective. *Library Hi Tech*, 2012.

[51] Jennifer Golbeck and Matthew Rothstein. Linking social networks on the web with foaf: A semantic web case study. In *AAAI*, volume 8, pages 1138–1143, 2008.

[52] Timothy Lebo, Satya Sahoo, Deborah McGuinness, Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. Prov-o: The prov ontology. 2013.

[53] Alistair Miles and José R Pérez-Agüera. Skos: Simple knowledge organisation for the web. *Cataloging & Classification Quarterly*, 43(3-4):69–83, 2007.

[54] Sebastian Hellmann, Jens Lehmann, and Sören Auer. Nif: An ontology-based and linked-data-aware nlp interchange format. *Working draft*, page 252, 2012.

[55] Paul Buitelaar, Mihael Arcan, Carlos A Iglesias, J Fernando Sánchez-Rada, and Carlo Strapparava. Linguistic linked data for sentiment analysis. In *Proceedings of the 2nd Workshop on Linked Data in Linguistics (LDL-2013): Representing and linking lexicons, terminologies and other language data*, pages 1–8, 2013.

[56] Mark Summerfield. *Programming in Python 3: a complete introduction to the Python language.* Addison-Wesley Professional, 2010.

[57] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.

[58] Wes McKinney. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython.* O'Reilly Media, 1 edition, February 2013.

[59] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[60] John D Hunter. Matplotlib: A 2d graphics environment. *IEEE Annals of the History of Computing*, 9(03):90–95, 2007.

[61] Michael L Waskom. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[62] Leonard Richardson. Beautiful soup documentation. *Dosegljivo: https://www. crummy. com/software/BeautifulSoup/bs4/doc/.[Dostopano: 7. 7. 2018]*, 2007.

[63] S Thivaharan, G Srivatsun, and S Sarathambekai. A survey on python libraries used for social media content scraping. In *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, pages 361–366. IEEE, 2020.

[64] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. *Jupyter Notebooks-a publishing format for reproducible computational workflows.*, volume 2016. 2016.

[65] Yuxing Yan and James Yan. *Hands-On Data Science with Anaconda: Utilize the right mix of tools to create high-performance data science applications*. Packt Publishing Ltd, 2018.

[66] Bhargav Srinivasa-Desikan. *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd, 2018.

[67] spacy · industrial-strength natural language processing in python. `https://spacy.io/`. Accessed: 2021-05-17.

[68] Tim O'Reilly and Sarah Milstein. *The twitter book*. " O'Reilly Media, Inc.", 2011.

[69] Kevin Makice. *Twitter API: Up and running: Learn how to build applications with the Twitter API*. O'Reilly Media, Inc., 2009.

[70] J Fernando Sánchez-Rada, Oscar Araque, and Carlos A Iglesias. Senpy: A framework for semantic sentiment and emotion analysis services. *Knowledge-Based Systems*, 190:105193, 2020.

[71] Babak Bashari Rad, Harrison John Bhatti, and Mohammad Ahmadi. An introduction to docker and analysis of its performance. *International Journal of Computer Science and Network Security (IJCSNS)*, 17(3):228, 2017.

[72] Joshua Cook. Docker hub. In *Docker for Data Science*, pages 103–118. Springer, 2017.

[73] Randall Smith. *Docker Orchestration*. Packt Publishing Ltd, 2017.

[74] Spotify. luigi documentation. `https://github.com/spotify/`. Accessed: 2021-05-17.

[75] Clinton Gormley and Zachary Tong. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*. O'Reilly Media, Inc., 2015.

[76] Arshak Khachatrian. *Getting Started with Polymer*. Packt Publishing Ltd, 2016.

[77] Jian Yang and Mike P Papazoglou. Web component: A substrate for web service reuse and composition. In *International Conference on Advanced Information Systems Engineering*, pages 21–36. Springer, 2002.

[78] Tim Ambler and Nicholas Cloud. Bower. In *JavaScript Frameworks for Modern Web Dev*, pages 1–9. Springer, 2015.

[79] Julio Villena Román, Sara Lana Serrano, Eugenio Martínez Cámara, and José Carlos González Cristóbal. Tass-workshop on sentiment analysis at sepln. 2013.

[80] Tok2vec · spacy api documentation. `https://spacy.io/api/tok2vec`. Accessed: 2021-05-24.

[81] José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*, 2020.

[82] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10:3152676, 2017.