UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MASTER

Design and development of a data analysis framework based on users' interaction from Stackoverflow and Reddit

ADRIÁN GONZÁLEZ HERNANDO

2023

TRABAJO DE FIN DE MASTER

Título:	Diseño y desarrollo de un análisis de datos basado en la interacción de los usuarios de Stackoverflow y Reddit
Título (inglés):	Design and development of a data analysis framework based on users' interaction from Stackoverflow and Reddit
Autor:	Adrián González Hernando
Tutor:	Álvaro Carrera Barroso
Ponente:	
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 DE FIN DE MASTER

Design and development of a data analysis framework based on users' interaction from Stackoverflow and Reddit

Julio 2023

Resumen

El auge de las redes sociales en nuestra sociedad ha dejado de ser una moda para implantarse como una manera de comunicación entre nosotros. En estas plataformas compartimos nuestras experiencias con todo tipo de detalles, dando nuestra opinión de forma libre y segura. Sin embargo, existen herramientas capaces de obtener y procesar esta información, los llamadas sistemas de monitorización. Los sistemas de monitorización son cada vez más frecuentes en las empresas que quieren mejorar su plan de negocio. Además, estas herramientas si las acompañamos de otras tecnologías, como análisis de sentimientos, son capaces de proporcionar una gran cantidad de datos valiosos a las compañías sobre sus producto, obteniendo las sensaciones de los usuarios al utilizar sus productos.

Bajo este pretexto, el objetivo de este trabajo fin de máster es la descarga, análisis y visualización de los comentarios de los usuarios en las redes sociales de Reddit y StackOverflow. Estas dos redes sociales han sido seleccionadas debido a que son dos grandes foros de internet en los que se pueden encontrar información sobre las empresas desarrolladoras de software.Nos centraremos en las tecnologías de DevOps, ya que estas tecnologías han tenido un importante incremento en todo tipo de empresas.

Con sistema de monitorización de comentarios de tecnologías de DevOps podremos analizar las tendencias de de una tecnología sobre la otra a través del análisis histórico. Además, podremos analizar el impacto positivo o negativo de una versión a través del análisis de sentimiento permitiendo detectar problemas o áreas de mejora para futuras versiones.

Para llevar a cabo este proyecto, en primer lugar, se deberá realizar un sistema capaz de obtener los comentarios. Una vez que se hayan descargado los comentarios, para poder sacarle el máximo partido a este sistema, serán analizados. Una vez que se haya realizado todo este análisis, los datos tendrán que ser guardados en una base de datos. Finalmanete, se configurado un panel para que poder analizar y visualizar los datos de la mejor manera.

Palabras clave: Reddit, StackOverflow, Elasticsearch, Kibana, GSIcrawler, Senpy

Abstract

The rise of social networks in our society has ceased to be a fad to certainly become a way of communication among people. On these platforms, we can share our experiences with all kinds of details as well as state our opinion freely and safely. However, there are tools capable of obtaining this information, the so-called monitoring systems. Monitoring systems are becoming more and more frequent in companies that want to improve their business plan. These tools are able to provide a large amount of valuable data to companies about their goods in all sorts of products.

In this context, the aim of this master's thesis is to download, analyze and visualize user comments on the social networks Reddit and StackOverflow. These two social networks have been selected because they are two major internet forums where you can find information about software development companies. We will focus on DevOps technologies since these technologies have had a significant increase in all types of companies.

With the DevOps comments monitoring system, we can analyze the trends of one technology over the other through historical analysis. In addition, we will be able to analyze the positive or negative impact of a release through sentiment analysis allowing us to detect problems or areas of improvement for future releases.

In order to carry out this project, first of all, a system capable of obtaining comments will have to be developed. Once the comments have been downloaded they will be analyzed in order to optimize this system. Once all this analysis has been done, the data will have to be stored in a database. Finally, a dashboard is configured so that the data can be analyzed and visualized in the best way.

Keywords: Reddit, StackOverflow, Elasticsearch, Kibana, GSIcrawler, Senpy

Agradecimientos

Este Trabajo Fin de Máster supone el cierre de un ciclo, que sin el apoyo de las siguientes personas no habría podido finalizar.

En primer lugar, quisiera dedicárselo a mis padres, Esther y Luciano. En segundo lugar, gracias a Andrea por su valiosa paciencia y apoyo durante la ejecución de este TFM. En tercer lugar a la memoria de mi abuelo Emilio. Por último, y no menos importante, quiero expresar mi más sincero agradecimiento a mi tutor, Álvaro Carrera, por su dedicación y apoyo a lo largo de este TFM.

Contents

List of Figures

List of Tables

Listings

CHAPTER 1

Introduction

This chapter will show an overview of monitoring systems, explaining the importance of monitoring systems nowadays and some of the use cases of these systems that have been reported in the media. In addition, we will explain why Reddit and StackOverflow have been chosen as data sources, highlighting their relevance among developers and professionals in the sector. Finally, the objectives of this project and the structure of the document will be explained.

1.1 Context

Extracting information from Internet has become a task that we perform almost every day for any matter, whether it is looking for a customer service phone number of a company, searching for information about a movie or even looking for reviews of a certain item. These are tasks that we have integrated into our daily tasks. However, if we want to go a step further, as for example, companies can do for the viability of their business, we can also perform tasks that provide an incalculable value for our business plan, for example obtaining real information from our customers.

Consumers are able to look for a hashtag or a specific word in almost social networks in which users ask questions, give their opinions and look for information. Therefore, in these hashtags, there are real opinions of consumers that can be used by companies to have feedback from customers in real time in an almost unlimited way. This information can be used, for example, to fix a product defect in the assembly line, improve customer service or analyze customer satisfaction with our service. This gives place to the creation of monitoring tools. Monitoring tools can not only be used to track a certain product, they are also used to analyze the competitors or as a way to search for a business opportunity. Through the Internet, you can observe total brand mentions in a specific region, the number of positive or negative mentions, and monitor the social media sentiment score over time [30].

The fact that it is impossible to analyze the comments manually, at least, on the most famous social networks due to the large amount of data in them, has turned into the need of developing automated social monitoring tools. The just mention monitoring software is composed of different modules such as sentiment analysis, real-time analysis, and automated reporting, making the combination of all of these a very valuable tool. At this point, what this thesis proposes is to create a tool to capture the information and analyze it.

Given the reasons in the previous paragraphs, it is clear that web scraping and its analysis are important. However, those who are still not convinced of the usefulness of these techniques can be convinced with the following examples.

The headline of the following article shows Uber's striking strategy, "Uber's Massive Scraping Program Collected Data About Competitors Around The World" [19]. Uber set up a system (called "Hell") through its market analysis team to periodically obtain data from transportation companies around the world by obtaining information about the technology they used and also about their drivers and executives. This information allowed Uber to identify drivers who also worked for its competitors. Uber obtained this data through public access websites and applications. This data was of such importance that they even took shifts to verify that the automated system was working all the time.

Another example is that price comparison portals, such as Trivago, use web scraping

techniques to obtain data from other websites in order to compare all of them and present it in their service [52].

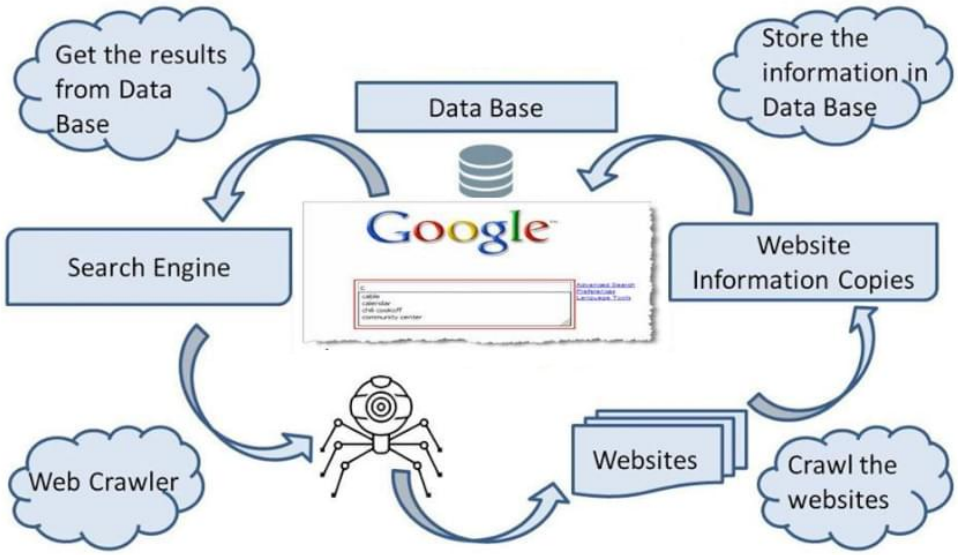


Figure 1.1: How Google search works. [21]

Another example of another large company using web scraping services is Google [21]. In the following image 1.1 we have a high-level design of how Google's webscraping system works.

Google searches are divided into three steps.

- Crawling: Thanks to web scraping Google is able to download the data (text, images, videos, and more) found on the websites we want to visit.
- Indexing: Once it has downloaded the web pages it parses them and stores the information in an index. The reason why Google stores the data on its servers is to provide a faster search service to users.
- Serving: Finally, the result presents the user with the results closest to the search that the user wanted to insert.

These three examples show the importance of being able to process the data displayed on the Internet as fast as possible. In addition, downloading the data with an accurate analysis of the data obtained can be very relevant information for our possible business or study. If we apply this useful tool to the monitoring of social network comments, we can obtain a lot of information that at first may not seem to be available to us.

By applying this case to software companies, several problems can be solved quickly and efficiently. To begin with, monitoring systems enhance early detection and diagnosis of

problems in real time. They also enable customer service by monitoring comments on social networks by locating users' questions quickly. In addition, they provide valuable information on how users interact with the software. Two forums where software companies can use this monitoring are Reddit and StackOverflow.

On the one hand, Reddit is an internet platform in which any user, previously registered, can comment on a particular topic, what is known as an internet forum. Reddit is a free forum and you can access it through *https://www.reddit.com/*. Reddit was founded on June 23, 2005, in the United States by two young students from the University of Virginia, Alexis Ohanian, and Steve Huffman. Although after a year they sold the company to Conde Nast, in 2014 they returned as executive chairman and CEO, respectively. Reddit is a platform that is valued at more than US$10 billion after a new round of funding [44]. According to Forbes magazine [16], since 2019, the company has increased its revenue by $700 million (595.16 € million). Its name comes from the phrase "I already read it". In this first graph, we can see the amount of data generated by the Reddit platform (in orange below LinkedIn). It generates one post, 13 comments, and 212 user interactions in the threads.

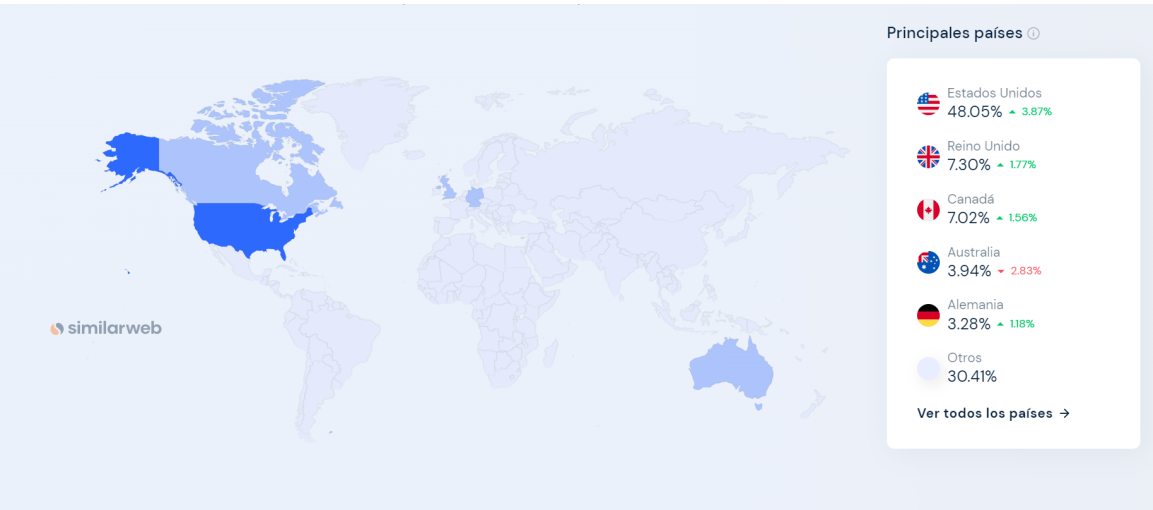


Figure 1.2: Geographic and country segmentation. [47]

In the map 1.2, we can see the strong impact that it has in the United States, where it has almost 50% of its share. It is also very powerful in other English-speaking countries such as the United Kingdom, Canada and Australia. Furthermore, similarweb provides also the gender segmentation of its users which is 72.23% male and 27.77% female. In terms of age, the group of users who use Reddit the most is the group of 25-34 years old, which occupies 31.23% of Reddit users, followed closely by the group of 18-24 with 30.25% [47].

There are large companies that have used the forum to advertise, for example, Audi used this tool for one of its campaigns (it was called Think Faster and it was in 2018). Audi

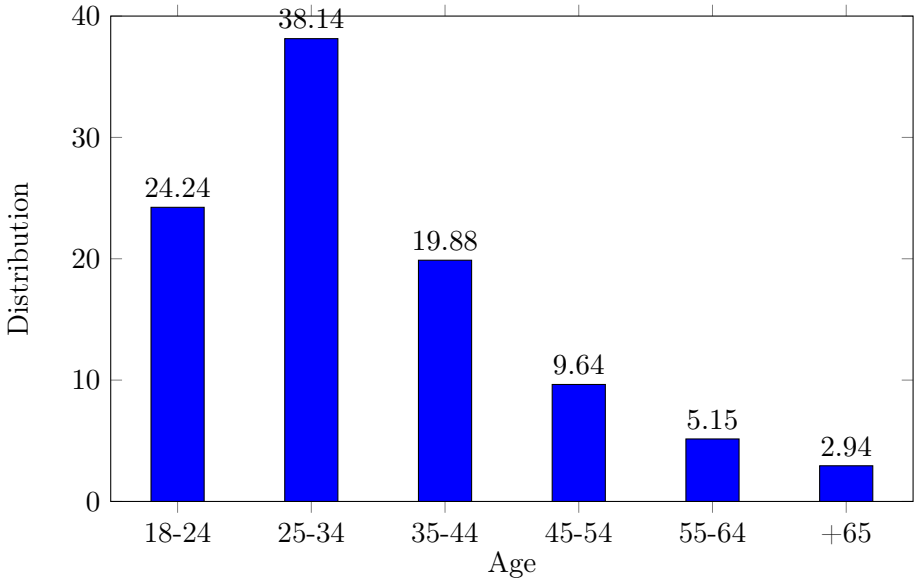
recorded a set of videos in which various celebrities answered questions from Reddit users while driving at high speed. [3]

On the other hand, StackOverflow is a very popular internet forum among developers. StackOverflow has become in recent years the reference place to answer programming questions. Through the questions and answers of the users, knowledge is shared among its users about different programming languages. They say they have over 100 million visitors a month. StackOverflow was founded in 2008 and it is the largest online community of developers. [60]

In 2021 Prosus [40], a huge telco company in the world, bought a part of StackOverflow. Prosus bought 46.5% of Tencent in 2001 and two decades later Tencent is one of the most powerful software and entertainment companies in the world. According to the official statement, the acquisition will be made for a total of 1.8 billion dollars.

Regarding the use of this platform, similarweb [48] provides us that 78.88% of the users are men compared to 21.56% who are women. It also tells us that 20% of StackOverflow users come from the United States, 10% from India, and 4% from the United Kingdom, showing a greater geographical diversity than what we will see in the next section with Reddit. Regarding the age distribution, we can see it in the following graph.

Therefore, we are talking about two leading Internet forums. Every day millions of users visit, interact and comment on these forums adding important value.



1.2 Motivation

According to the image below 1.3, taking into account that the photo is from 2021, we can see the number of tweets, Tiktok interactions, and photos uploaded to Snapchat, or the

number of searches on Linkedin. We can notice how much data is generated in one minute on any given day. Therefore, so as to realize a system that obtains data from the Internet, it is mandatory to do it through an automated process.

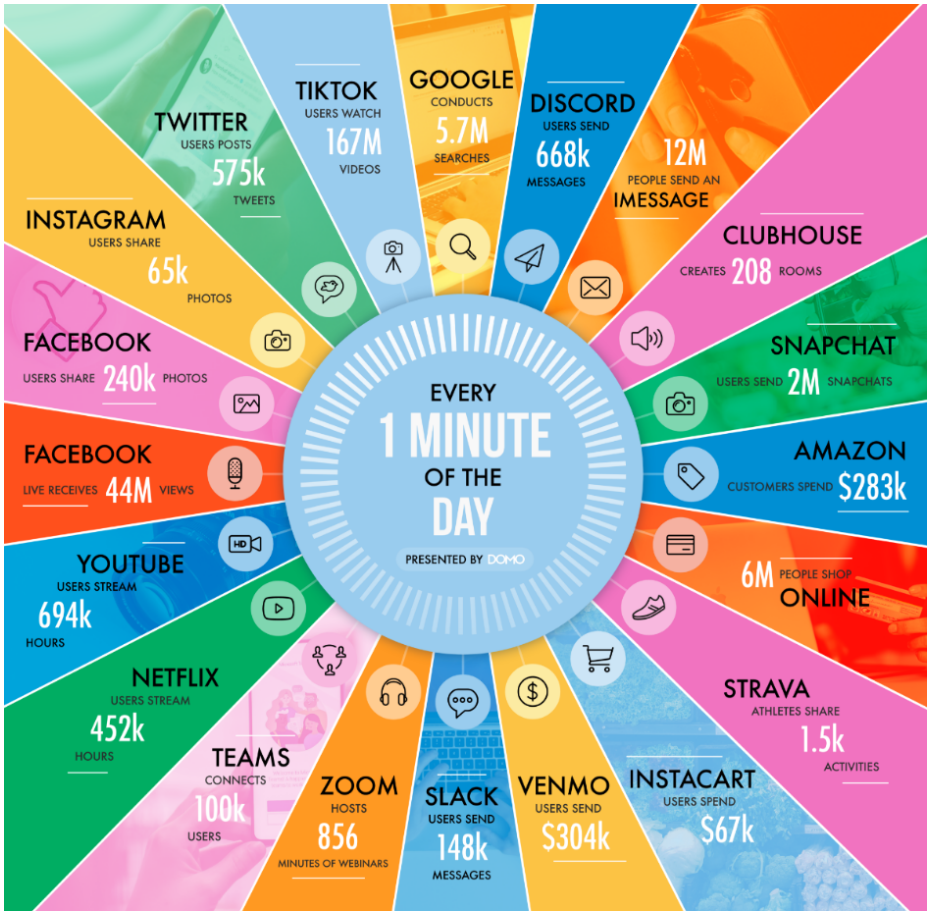


Figure 1.3: How quickly data is generated. [24]

This means that if we can build a system to monitor or analyze people's reactions to events we could understand market trends. For example, if we were a company monitoring users' reactions to an update of our product, we could gain a better understanding of market trends. Therefore, creating this system on two of the leading platforms in terms of software developers is a very valuable system. The two leading Internet forums are StackOverflow and Reddit. As in the companies just mentioned in the previous figure, these forums are visited by millions of users every day, they interact and comment on these forums providing important value, not only to the forum but also to those people or companies that want to go a step further.

Of course, companies probably have a private system for monitoring comments on multiple social networks. However, having visibility of a control panel similar to that of big companies, being able to analyze with our data the market trend, and having a dashboard

through open source systems, is a great way to have a better understanding of the market also to find answers to events that are in principle inexplicable.

1.3 Project goals

This Master's thesis proposes a system that allows the monitoring of user comments in two internet forums focused on programming. These two platforms are StackOverflow and Reddit. The comments will be extracted through the Reddit and StackOverflow API. Then, the comments will be processed using Natural Language Processing techniques. Once we have the comment analysis, a dashboard with different diagrams will be used so as to show the results.

Therefore, the objectives of this project are as follows:

1. Develop and implement in a scraper system a way to get the information This objective means that it is needed to develop new systems that allow making requests to StackOverflow and Reddit APIs to obtain the data needed for this thesis.

2. Add this information into a database. Once the API requests are made, collect the data and add it to the database.

3. Natural Language Processing techniques are used to analyze language. Natural Language Processing techniques are used to analyze language. Therefore, NLP must be used to process the comments extracted from StackOverflow and Reddit.

4. Create a dashboard so as to visualize the data analyzed. In order to correctly visualize the analyzed and extracted data, it should be presented in an interactive dashboard that allows the combination of several variables.

5. Analyze the results obtained through the analysis. Draw conclusions from the data provided by the analysis.

1.4 Structure of this document

In this section, we will introduce the structure of this project. This document is structured in five chapters. The content of the chapters can be summarized as follows:

Chapter 1: Introduction It will give perspective on the amount of data generated in social networks and the value of this data, and discuss the importance of social network monitoring and analytics systems.

Chapter 2: Enabling Technologies This chapter introduces the technologies that will be used in the development of the thesis. Data will be given on the size of the companies

that own the software we will use and we will explain why we have chosen the technologies we have chosen.

Chapter 3: Architecture and Methodology The overall system architecture is presented and will be divided into smaller modules. For these modules, their architecture will also be presented and their high-level operation will be detailed with diagrams.

Chapter 4: Case study This chapter explains the development of the prototype and each of the phases of the work. Every detail of the configuration will be explained and the reason, with examples, why certain decisions have been made regarding the design of the application will be explained. This chapter is the technical implementation of the previous chapter.

Chapter 5: Conclusions Finally, the conclusions obtained from the work are presented, as well as the problems encountered during the development of the work and the lines of improvement of the work.

CHAPTER 2

Enabling Technolgies

In this chapter, we will briefly explain the technologies that have been used to carry out this project. We will start by explaining the virtualization tools, which is the technology that has allowed us to deploy all the modules that have enabled the development of the project. We will continue talking about Natural Language Processing, how it has been applied in the project, and the impact it has. Then, we will discuss the web crawling and scraping tools and how they have been implemented in the project. Next, we will explain the technology that has been used as a database and then the frontend tool. Finally, we will finish by explaining the orchestrators and the advantages they have in this type of work.

2.1 Introduction

In this section, we are going to comment on all the technologies that have been used to develop this project from the installation of the environment to finally achieving their visualization. Each technology used will have its own section in which we will briefly comment on its function in the project, what use cases have that technology and which companies in the sector use it and also we will see the potential or the importance of that technology. In addition, in some cases, similar technologies will also be analyzed, and the advantages and disadvantages of the resources used will be presented compared to other options on the market in order to provide a complete overview and be able to evaluate them. This chapter is essential to understand and evaluate what kind of technologies can be used to carry out this project. In addition, it will help to have a global vision of the purpose of this work.

Finally, it should also be noted that in this chapter we will also explain some basic general concepts as an introduction to the project. This is important because it serves as a support to learn about the technology, for example in the second section we will begin to explain briefly what NLP is and then analyze the tool that we have selected for the analysis.

2.2 Virtualization Servicies

To virtualize means to be able to emulate something through a system. Virtualization now encompasses all kinds of concepts: you can virtualize an application or program, an operating system, data, a desktop, servers, networks, and much more. In this section, we will focus on applications. This virtualization, also called container-based virtualization, is a way of virtualization in which a virtualization layer runs on top of the operating system kernel, allowing multiple instances of the operating system to exist [57]. For this project, the tool used to virtualize these applications is Docker. We will now explain it.

Docker

In 2013, Docker released its first version as an open-source platform in March 2013, under the name of dotCloud as a Service (PaaS) written in the GO language. Docker began to gain popularity quickly, in fact in April 2015, two years after its release, the project had more than 20,700 GitHub stars, ranking it as one of the projects with the most GitHub stars, at 20th position) [55]

Docker allows developers to quickly build, run, and scale their applications by creating containers. Much of Docker's success is due to being Open Source and to the support of companies like IBM, Microsoft, RedHat and Google.

As previously stated, Docker [31] allows you to create containers so that software applications can run on any machine with Docker installed, regardless of what operating system the machine has underneath it, making deployments easier. Docker containers can be run locally in the customer data center, at an external service provider, or in the cloud. Therefore, it can be said that Docker is a kind of virtual machine that can be used on different devices through its containers. However, there are differences with respect to a virtual machine in practically all aspects. The following figure 2.1 shows a comparison between the two technologies.

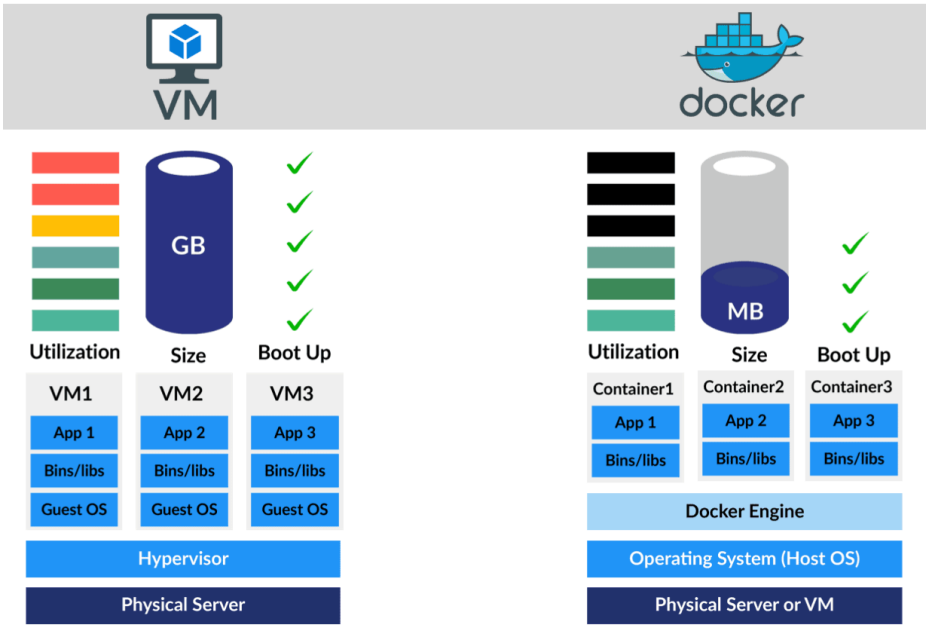


Figure 2.1: Docker vs VM. [4]

The main difference between the two technologies is that Docker shares the same operating system as the host, while a virtual machine mounts an operating system on top of the host operating system. By sharing Docker operating system avoids that we can encounter performance problems and hardware problems, for example, virtual machines require a lot of memory while Docker does not need them. Another aspect to be considered is also the storage capacity, while the virtual machine requires gigabytes to store the image being used, Docker only requires the download of the container.

The Docker architecture consists of three components:

- Docker client: The Docker client is the primary method of controlling the Docker server through a CLI. Allows you to create, manage and run containers.
- Docker service or docker daemon (dockerd) handles Docker API requests and Docker objects (images, containers, networks, and volumes)

- Docker images: These are the templates that, through their requirements, define the containers. Some images are public and can be downloaded, for example, through Dockerhub. Dockerhub is a public repository of container images. On the other hand, it is also possible to create your own image from a Dockerfile.

Docker compose

Docker compose is the tool used by Docker to be able to deploy several containers in a simulated way being able to interact with each other. Docker Compose uses a YAML file (under the name docker-compose.yml) that allows programming the configuration of multiple containers. Docker compose configuration involves not only defining different container images but also configuring the different ports and dependencies that the containers have on each other. This prevents the deployment of a container if it has a dependency on another one and this one has failed in its deployment.

2.3 Natural Lenguaje Processing

Natural Language Processing (NLP) [9] is a technology that allows computers the ability to interpret, manipulate and understand human language. As we have shown above, organizations have huge volumes of voice and text data through various communication channels such as emails, text messages, and audio in which they are often asked for almost immediate responses from consumers. In order to support this need, in most cases companies rely on NLP software.

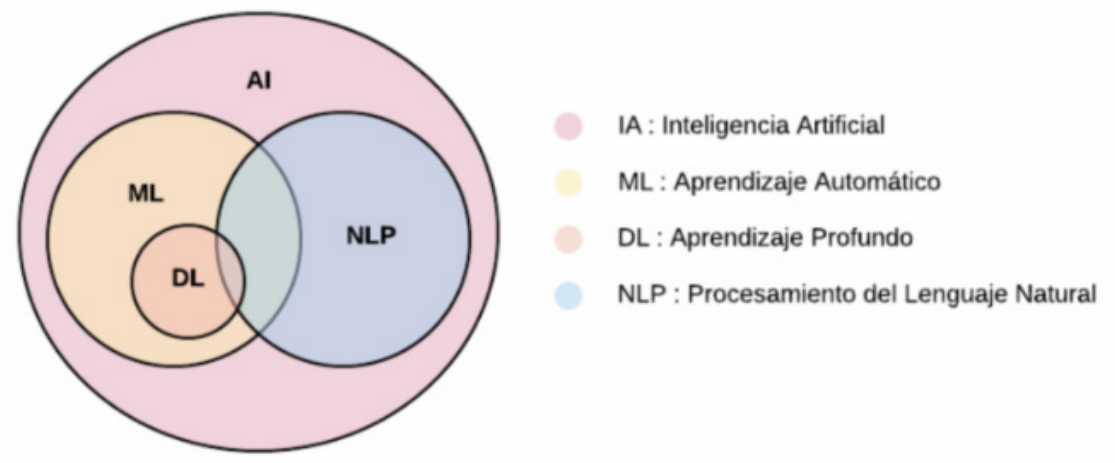


Figure 2.2: Relation between NLP, ML, AI, and DL. [41]

The applications that NLP software can have are the following: [53] [12]

- Virtual assistants or chatbots: Thanks to NLP they are able to understand messages and respond automatically. The first was ELIZA (1966), who used pattern matching and substitution to simulate a text conversation (in which it could neither hear nor speak). Currently, one of the most advanced on the market is Amazon ALEXA, a personal assistant that can perform all kinds of tasks, from setting alarms, timers or even shopping to home automation [8]. In addition to being a day-to-day assistant, there are also chatbots that are capable of answering questions. You can ask multiple kinds of questions: about definitions, cause/reason, yes/no, procedural, comparative, with examples, and opinion questions.

- Machine translations: The automatic translation of texts is a great support in society due to the large amount of information that users have at their disposal on the Internet. One of the most widely used Internet translators such as Google Translate [13] is a good example for this case study.

- Information retrieval: Information retrieval (IR) uses keywords to retrieve information and data, just like an Internet search engine. In these cases, no language is generated, but they can help users find what they are looking for or answer their questions using information they already have.

- Text classification: Texts can be sorted into user-defined categories or according to topics. This makes it possible to redirect the information, which is very useful when managing a large volume of data. This sorting function, for example, is used to detect spam in e-mails. [45]

- Sentiment detection: Sentiment analysis is a natural language processing technique used to determine whether data is positive, negative, or neutral. Sentiment analysis is often performed on textual data to help companies monitor brand and product sentiment in customer feedback and understand their needs. It is precisely this use case that will be developed in this paper. [18]

- Automatic Summarization: Through NLP we can summarize long texts, saving time. There are two models: extractive (extract sentences) and abstractive (plan the summary). [32]

The biggest challenges in NLP are linguistic variation, language ambiguity, and capturing irony or sarcasm. This is because the biggest challenges of NLP are related to human language understanding, in sentences where ambiguity exists. For example, many words have multiple meanings depending on the context, which can lead to misunderstandings in natural language processing tasks such as machine translation or sentiment analysis.

Furthermore, irony and sarcasm are literary figures of speech in which the meaning of the expressions is not literal and depends on the context. These factors settle the biggest challenges of Natural Language Processing.

Senpy

Senpy is a framework developed by GSI-ETSIT-UPM. It is a development that has been partially funded by the European Union through the MixedEmotions Project.

Senpy lets you create sentiment analysis web services easily. It uses a mix of web and semantic technologies such as JSON-LD, RDFlib, and Flask. Senpy provides a user-friendly interface.

Senpy's architecture is shown below, in which several clear phases can be distinguished, the first of which is the input. Senpy can be used through NIF HTTP Query or NIF Command Line. NIF (NLP Interchange Format) is an RDF (Resource Description Framework) / OWL-based format that aims to achieve interoperability between Natural Language Processing (NLP) tools, language resources, and annotations. After providing Senpy with the data, the data is validated, run with the selected plugin, and the response is created. After providing Senpy with the data the data is validated, run with the selected plugin, and validated, and the response is created. Senpy's output can be in JSON-LD, N3, or RDF format. JSON-LD, or JavaScript Object Notation for Linked Data, is a method of encoding linked data using JSON. Notation3 (N3) is a shorthand non-XML serialization of Resource Description Framework models.

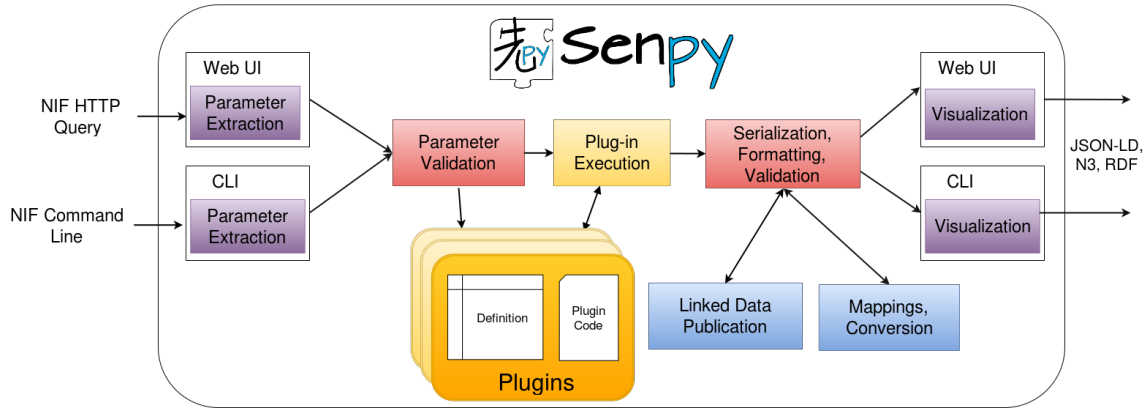


Figure 2.3: Senpy Architecture.

2.4 Web crawling and web scraping

As mentioned in the introduction to the project, more and more services on the Internet are using these technologies. Although these two concepts are sometimes similar, there are certain differences between them.

On the one hand, a web scraper has a defined purpose, which is to find specific information. It has certain components such as the downloader, the requests, the answers, and the crawlers. This last component of scraping is the main reason why it is confused with crawling.

On the other hand, a web crawler is an extraction method based on sending *spiders* to collect web pages related to a specific search. An example of trackers used by large companies is "Googlebot" used by Google, "Bingbot" used by Microsoft's Bing, and "Slurp Bot" used by Yahoo. [34]

The following table shows the fundamental aspects of each of them.

Web crawler	**Web Scraping**
collects hyperlinks from a website.	responds to any type of data found on the web pages.
indexes discovers and generates data	processes data and extracts structured data.

Table 2.1: Web Crawler vs Web Scraping

There are many online services that provide crawling and scraping. However the tool used to obtain the source data was GSIcralwer.

GSIcrawler

GSIcrawler is an API crawling application, this means that its goal is to collect information in an automated way through the API of certain websites. There are several ways to obtain information through an API crawler, for example through a bot or a computer program. The open-source Python framework 'Scrapy' is one technology that suits the computer program case.

For this project, the GSIcrawler tool has been used. GSIcrawler is a project developed by the GSI department of the ETSIT UPM. GSICrawler has three available platforms: Twitter, Reddit, and News. The user interacts with the tool through a web interface, selecting the platform and the parameters to be examined for example, in the case of the Twitter platform you can provide the user or a date to the plugin. In order to perform queries through GSIcrawler, a series of parameters that are requested by the API must be

configured. For example, if we want to scrape the data of a specific user we will have to give the nickname of the user and some more data if we only want the data in a specific period of time or about a specific topic that the user has written about. Therefore, it is necessary to adapt the framework that is prepared for GSIcrawler based on the needs of the API to which these queries are going to be made. GSIcrawler's output format is also flexible. It allows to adapt to the programmer's needs, choosing only the data to be used in the analysis. It is a tool that fits perfectly to the needs of this project since it allows you to obtain the data in JSON format or directly upload it to Elasticsearch configuring the index where you want to save it. In this project, two more plugins have been developed in GSIcrawler so as to carry it out: Reddit and StackOverflow.

As I mentioned before, the GSIcrawler tool already had the Reddit scraper, however, as there are several ways to get the information from Reddit, a new method has been created in GSIcrawler as an alternative to the one that was already created. The technical details that have been carried out in order to carry out the development will be specified later.

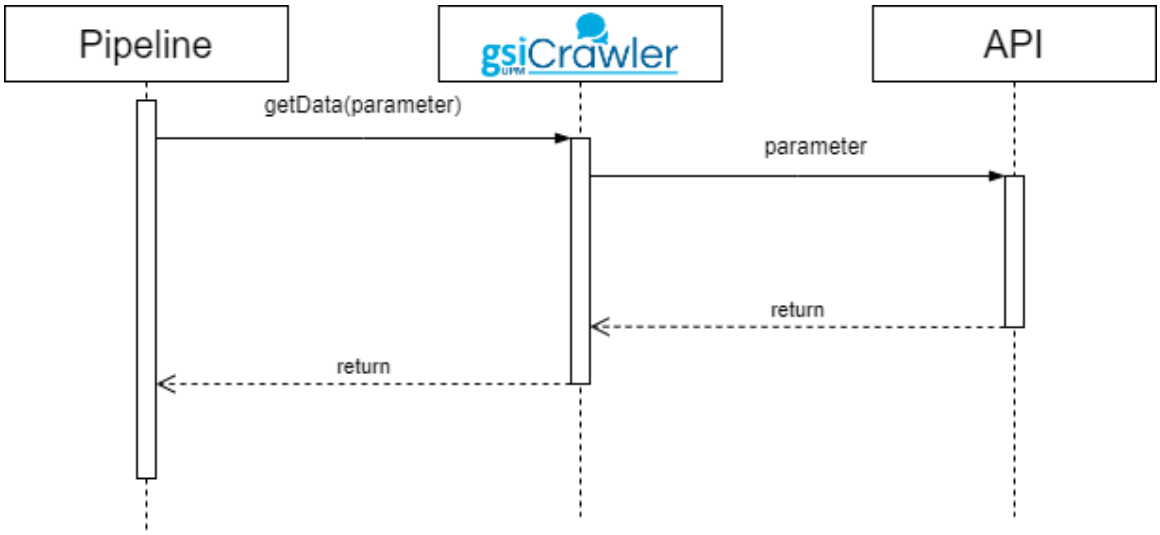


Figure 2.4: GSIcrawler Architecture.

2.5 Search engine and database

NoSQL technologies include search engines and allow for fast, reliable, and optimized search engines. With the new technologies of big data, cloud services, or real-time web application development, it becomes essential to have an efficient search engine.

Lucene (developed in 1999) has become the basis of two of the best open-source search engines: Solr (2004) and Elasticsearch (2010).

Although we always look for the best one when there are several technologies, in this

case, both Solr and Elasticsearch are designed for different use cases.

On the one hand, in the case of Solr it does not have a REST service, since it does not work with HTTP. As the entire Solr API is written in XML, it uses XML files. In addition, Solr is a search server that mainly works with. To put a downside to Solr is that it is difficult to configure due to its dependencies with Zookeeper.

On the other hand, Elasticsearch has a REST service. A REST (Representational State Transfer) API service enables the simple and standardized exchange of information in a secure, reliable, and efficient way. When a data request is sent to a REST API, it is usually done over HTTP. Therefore, the Elasticsearch requests are done through HTTP allowing us to perform the operations of this protocol such as GET, POST, PUT and DELETE. Data is sent in a structured format such as JSON and accessed via a URL. That is why Elasticsearch allows the use of JSONs, making API requests much simpler and more intuitive than in Solr. Elasticsearch has neither its own client nor its own internal protocol, since it uses HTTP it is not necessary to install anything additional.

Solr	**Elasticsearch**
No REST	REST
XML	JSON
Need client	Clientless
Need Zookeeper	Easier to install
	More popular
	High scalability

Table 2.2: Solr vs Elasticsearch

Therefore, Elasticsearch will be used for this project because the requests that will be made to the data sources use the JSON format, in addition to its high scalability. We will now proceed to explain Elasticsearch in more detail.

Elasticsearch

Elasticsearch [25] [46], based on Lucene and written in Java, is a powerful, distributed, open-source, and analytic tool that can store all types of structured (SQL) and unstructured (NoSQL) data. Elasticsearch is ideal for processing text but also stores numeric and geospatial data.

It uses QueryDSL (Query Domain Specific Language) to query the data stored in the indexes. An Elasticsearch index is a collection of documents that are stored in JSON-type documents. This point may be one of its major disadvantages, since it only supports JSON, discarding other languages such as CSV or XML.

Elasticsearch allows access to your data in real time thanks to its API. In addition, it supports different programming languages through its integrated libraries such as Java, Javascript, Python, and others.

Scalability is, without a doubt, the feature of Elasticsearch. It is highly scalable, which allows adding more resources when required. There are two ways to have scalability: vertical or horizontal. Due to its distributed architecture, Elasticsearch has horizontal scalability, which allows adding more nodes of the same characteristics at times of high workload. Meanwhile, vertical scalability involves enhancing the features and capabilities of a single server. In this way, one server is able to handle a larger workload, rather than distributing the load among multiple servers as it is done in horizontal scalability.

Regarding security, Elasticsearch allows login with user credentials. In addition, different systems can be integrated, allowing authentication services through Active Directory, LDAP, or Elasticsearch native domains. It uses single sign-on (SSO) options, such as certificates or Kerberos among others [27]. In addition, it offers Roll-Based Access Control (RBAC). Role-based access control or RBAC is a way that allows or restricts user access to data based only on the user's role in the organization.

This tool is used by a large number of leading companies in their sectors such as Tesco, Linkedin, Foursquare, Facebook, Netflix, Dell, Ebay, Wikipedia, The Guardian, New York Times, Salesforce, Docker, Orange, Groupon, Eventbrite, and many others. [11].

In the next photo, we can notice the different parts of Elasticsearch's architecture. There are basic parts of Elasticsearch that are used.

- Cluster: A cluster is a set of one or more nodes that keep all the information distributed and indexed.
- Node: A node is a server that is a component of a cluster, stores your information, and helps with the tasks of indexing and searching the cluster.
- Index: It is a collection of documents that has similar characteristics. The indexes will be used for indexing, searching, updating, and deleting information into Elasticsearch. Indexes are identified by a name.
- Shard: This tool is designed in order to make the indexing task easier.

Regarding the queries that can be made in Elasticsearch to obtain the information, different types of queries can be performed, such as:

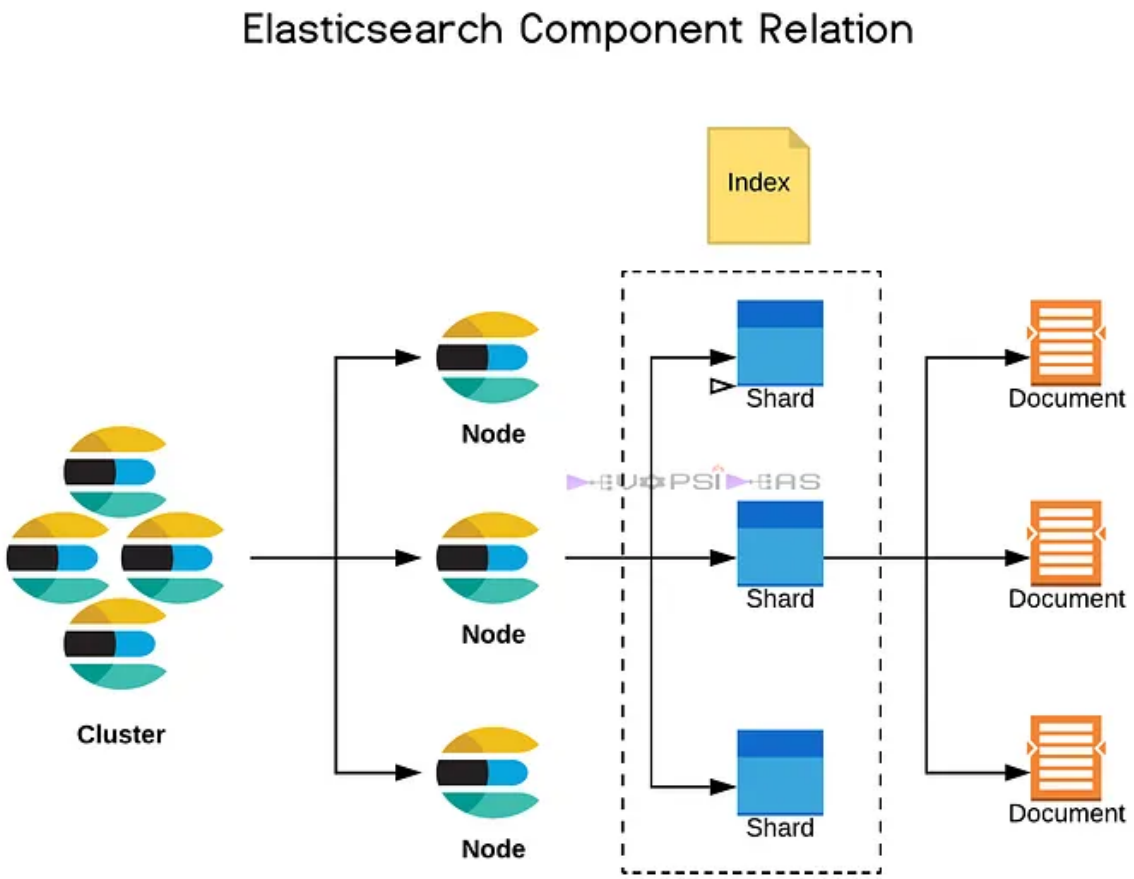


Figure 2.5: API Maps Integration

- Terms or complete phrases queries: Searches for data containing one or more specific terms or containing an exact phrase.
- Range queries: Searches the data that are between the values that are specified. These values can be for example numerical or date specific.
- Aggregation queries: Performs the requested calculations for specific requests, for example, it can calculate the average age of blond-haired people who have a house in Madrid and Malaga.

In the following listing, we have an example of a query, which will search for all documents that contain in the field 'cloud' the word 'Azure', whose RTO is greater than or equal to 4 hours and the region is Northern Europe although this last parameter is optional.

Listing 2.1: Elasticsearch example response

```
GET /_search
{
    "query": {
        "bool": {
            "must": [
                { "match": { "cloud": "Azure" }},
                { "range": { "rto": { "gte": 4 }}}
            ],
            "should": [
                { "match": { "region": "NEU" }}
            ]
        }
    }
}
```

2.6 Visualization tool

The next step is to discuss the tool that has been used to visualize the data. The objective of monitoring is to have a control panel in which all data can be represented in an easy, simple, and interactive way. Tools such as Sefarad, Kibana, or Grafana are available for this purpose. There are several data visualization tools such as Grafana, Kibana, or Sefarad. We will now briefly discuss each of them.

- Kibana uses Elasticsearch as the search engine for its analysis, and integrates the Logstash data processing tool to complement its functions. Kibana is the tool that will finally be used.
- Grafana is free software that offers a customized visualization system. Grafana allows you to create customized dashboards and graphs.
- Sefarad is a tool developed by the GSI of the ETSIT UPM. Sefarad allows to visualization of the data through Polymer components.

Kibana

Kibana, also from Elastic, is a frontend, open-source application that provides data visualization and search capabilities for data in Elasticsearch. [28]

Through the different options, we can create different interactive graphs for the representation of the data indexed in Elasticsearch. Kibana allows us to represent the data through:

Grafana	Kibana	Sefarad
High compatibility	Data visualization	Easy graph creation
Supports various data sources	Integration with Elasticsearch	
Easier installation/deployment	Interactive charts	

Table 2.3: Grafana vs Kibana vs Sefarad

- Charts: Kibana has a wide variety of charts from bar charts, pie charts, geographical maps, or time charts.

- Dashboards: The collection of graphs in a single screen is called Dashboards. Kibana allows to add on the same page a multitude graphs. It allows mixing different types of charts allowing the developer to add the type of chart that he considers most appropriate from an aesthetic or analytical point of view.

- Canvas: It is another option similar to the Dashboard. However, it allows a bit more customization and detail. It allows you to edit the images and text. Furthermore, Canvas allows you to add your own CSS code.

From a security point of view, Kibana offers the same features that I discussed above for Elasticsearch in terms of authentication and authorization systems.

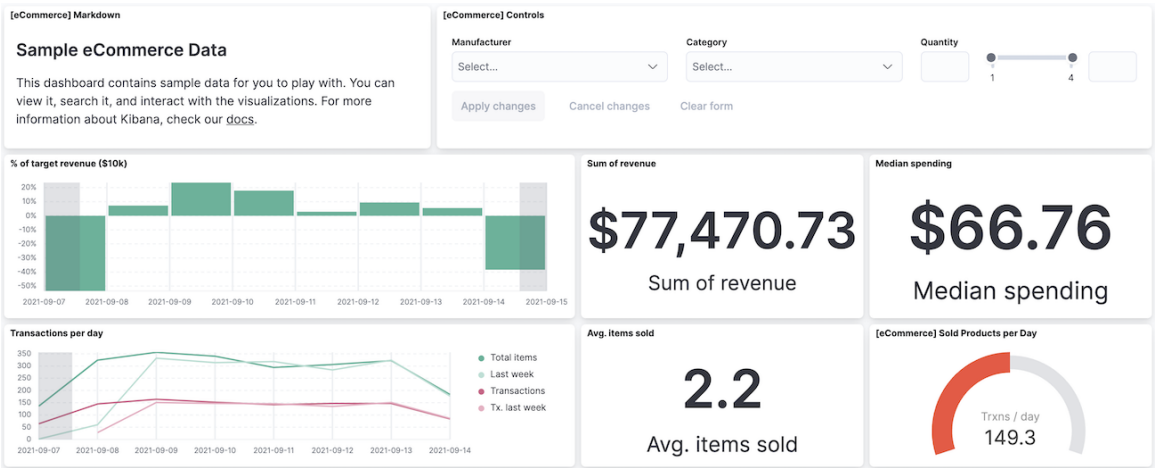


Figure 2.6: Example of a Kibana Dashboard. [41]

As in Elasticsearch, we have to highlight the scalability of Kibana. It is highly scalable and can be used to use large amounts of data and simultaneous queries.

Kibana, like Elasticsearch, integrates Machine Learning options that can be trained to find anomalies. It can also notify us by the channel that is configured, for example, an email.

In the following figure 2.6 we can notice a Kibana Dashboard example. We can appreciate different kinds of graphs. We can also configure a filtered chart, bar chart, time-based, pie chart, or operation (Median and Sum).

2.7 Orchestrator technologies

Python orchestrators have become very popular due to the complexity of programs written in Python. A data orchestrator is a program that is responsible for following the steps necessary to execute the flow we have defined in our diagram. Among these tasks are tasks such as execution, cleaning, validation, or transformation of data to make it suitable for what we need. Data pipelines can be simple and consist of a few steps, or they can be complex and involve multiple steps and tools.

Python offers several frameworks for creating data pipelines, such as Apache Airflow, Luigi, and Prefect.

- Apache Airflow: a powerful open-source platform for creating, scheduling, and monitoring workflows in Python. It is the one to be used in this project.
- Luigi: a Python module developed by Spotify that simplifies the construction of complex data pipelines.
- Prefect: a modern data pipeline framework focused on simplicity, flexibility, and scalability.

Now we will compare and discuss the advantages of Luigi and Airflow. On the one hand, Luigi is written in Python and is used only in Python. On the other hand, Airflow is also written in Python, but it allows us to define tasks using DSL. Regarding administrative tasks, Airflow has a web interface through which you can view the execution of the pipeline and manage workflows, while Luigi does not have this interface. Regarding task automation, one advantage Luigi has is that you can schedule tasks, while Airflow does not. The following table summarizes the comparison of the two technologies:

Therefore, due to the complexity of the web interface, the execution of tasks in parallel has been decided to use Airflow. The following is an explanation of Airflow technology.

Apache Airflow

Apache Airflow is an open-source workflow manager tool written in Python. The project was created in October 2014 at Airbnb and this company managed to more easily manage

Luigi	**Apache Airflow**
Simplicity	Programming by flows
Support schedule tasks	Web interface
Used only through Python	Scalability

Table 2.4: Luigi vs Apache Airflow

their workflows and control them thanks to the user interface included in Airflow [1]. Airflow has garnered so much interest that it has also been integrated into the Google Cloud stack as the tool to orchestrate its services in 2018. Nowadays, it is used in production by more than 200 companies worldwide, such as Paypal, Spotify and Twitter.

Airflow [23] is used to automate jobs programmatically by dividing them into subtasks. The most common use cases are automation with data processing related, time-periodic, and administration tasks. This is possible because Airflow allows you to schedule tasks and also to execute them on demand. Therefore, one of the advantages of Airflow is that it is highly scalable. It can be integrated with multiple third-party systems such as databases (It is compatible with popular databases such as MySQL, Postgres, SQLite, or with JDBC.), cloud services (AWS, Google Cloud Platform, or Microsoft Azure), monitoring tools, etc. The integrations with cloud environments are very powerful. Previously we discussed the integration of Airflow with GCP and Azure has developed Azure Data Factory Managed Airflow. This service is a simple and efficient way to create and manage Apache Airflow environments [36]. AWS also has a similar service with Amazon Managed Workflows for Apache Airflow. Airflow has a very good user interface for dynamic interaction. Another great advantage it has is its large online community which has great support with more than 30,000 stars on GitHub. Airflow has a very rich command line interface that allows a lot of different kinds of operation on a DAG, starting services, and supporting development and testing. [7]

Apache Airflow has five components: Web Server (Flask Server with Gunicorn), Scheduler (Daemon in charge of workflows), Metastore (Database where metadata is stored), Worker (Process that executes your task) and Executor (It is responsible for executing the tasks and defines how tasks are executed). Everything is executed in the scheduler in a development or local environment with the exception of the production tasks. Workers are in charge of the execution of the production environment.

Airflow uses Directed Acyclic Graphs, or DAGs, for the creation of tasks and workflows. A DAG is a data loop defined in Python. Each DAG represents a sequence of tasks to be

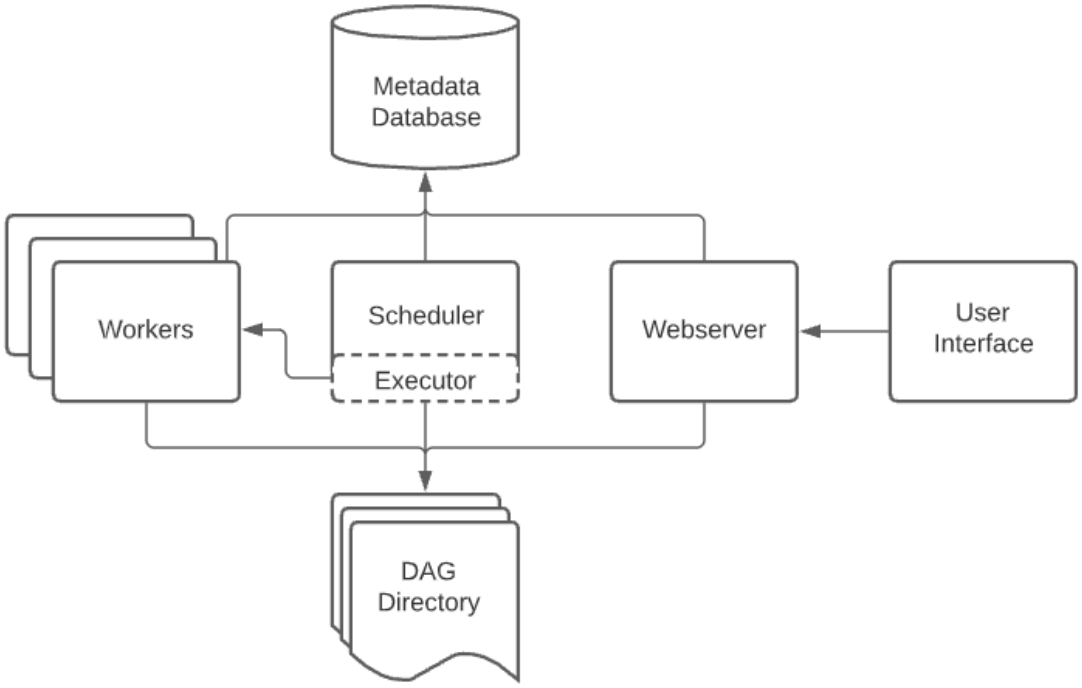


Figure 2.7: Apache Airflow Architecture. [6]

executed and organized with Airflow task relationships. The graphical interface allows you to monitor the execution status of tasks but does not allow you to create new flows.

In order to accomplish the above tasks, the workflow must have two basic conditions. They must be directed and acyclic. To be directed means that the relationships between the different nodes have only one direction. Being acyclic means that no loops can be formed in the workflow execution, i.e. the execution cannot return to a node that has already been executed previously.

To sum up, Apache Airflow is a simple tool for the implementation of workflows, which allows us to automate pipelines. Its user interface makes it easy to interact with and monitor workflows represented by DAGs.

CHAPTER 3

Architecture and Methodology

This chapter presents the methodology and architecture that have been carried out in this work. The order of the implementation of the modules and why will be explained. The general architecture of the project is also described, with the connections between the different components involved in the development of the project. In addition, we will go into detail in each of the modules that are exposed in the general architecture, explaining each one in detail. This chapter aims to show the project at a high level, without going into the technical details of the implementation.

3.1 Methodology

This section will detail the steps that have been followed to carry out the project. As we can see in the image below 3.1, the project consists of four fundamental modules and a fifth that cannot be represented graphically (which is the orchestrator module and it is represented by the black arrow, which will be explained in this chapter). To carry out the project implementation, a waterfall or sequential methodology has not been used because it has not been considered the most ideal.

The work methodology has been Scrum. Scrum is an agile work methodology for project management and development, especially focused on software development. Scrum divides activities into different sprints, some of which are very common. Of the most common in this project, the following sprints have been carried out:

- Sprint Planning: Meeting to plan the work to be done during the next sprint.
- Sprint Review: Meeting at the end of the sprint in order to demonstrate and review the work done.
- Daily Scrum: This is a meeting that is held daily to see the evolution of the project. However, since it is not possible to carry out a daily follow-up, a follow-up has been carried out every two weeks to see the evolution of the work.

Furthermore, these particular Sprints have been also carried out:

- Scraper development sprint: The reason for starting with this module is to be able to start working with some data test
- Airflow development sprint: Once we had a module working, the next sprint was to integrate it into Airflow.
- Visualization module development sprint: When Airflow was already working, the next Sprint consisted of being able to visualize said data, even if they were not analyzed. Simply to have a first functional version of the system.
- Analysis module development sprint: This Sprint consisted of analyzing data from Reddit and StackOverflow. In such a way that with this analysis we already had a first functional version of the complete system.
- Data development Sprint Once the development was completed, data collection phase was carried out in order to have a good number of comments. In order to be able to collect the data and not reach the limit of daily requests, this sprint had to be

carried out over several days. Once a sufficiently large number of comments have been downloaded to be able to carry out an analysis, this sprint has been finalized.

3.2 Architecture

In this chapter, we will briefly discuss the overall architecture and then we will analyze module by module. Therefore, the whole architecture of the system is as follows:

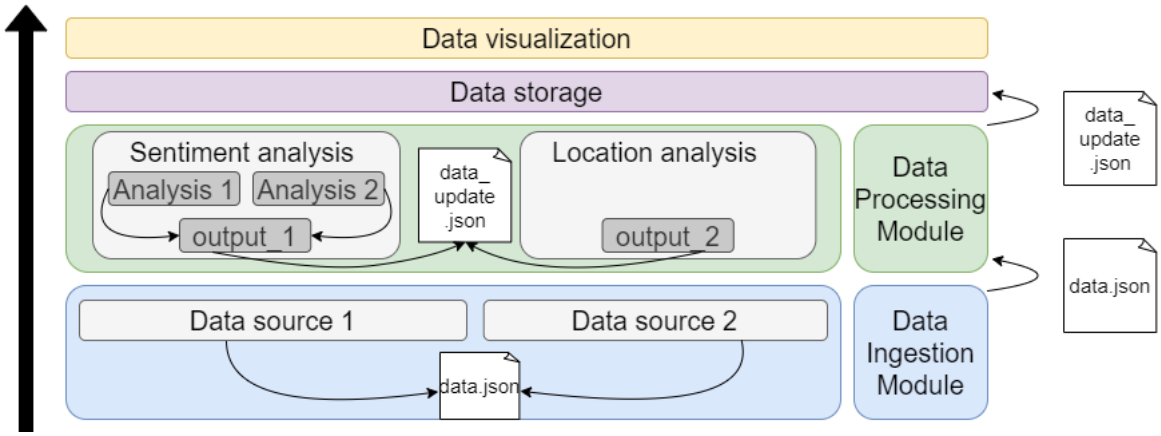


Figure 3.1: Architecture diagram.

We can see that it is composed of four modules: Data Ingestion Module, Data Processing Module, Storage Module, and Visualization Module. The Data Ingestion Module is in charge of obtaining the information from the Reddit and StackOverflow APIs. Once the target threads are obtained from the comments, the information is sent to Data Processing Module so as to analyze it and get the result. Both data, Data Processing Module together and the original data from the APIs, are stored in the Storage Module and then it will be sent to the Visualization Module. There is one more module that has not been added to the graph so as not to make the visualization of the diagram more difficult. This fifth module is the pipeline manager which is in charge of executing all the pipeline tasks and communicating the previous modules with each other. This module will be the so-called "orchestrator module".

Next, we will explain what is the pipeline flow through the state diagram. As we can see in the diagram there are two possible starts in the flow. The beginning of the pipeline is at point A, where we will start to get the comments we want.

If we exceed the daily limits of the StackOverflow platform we will receive a 502 error message, which will terminate the state flow. This is represented in the diagram by the point B.

Once it has been verified that all the tasks related to this state have been completed

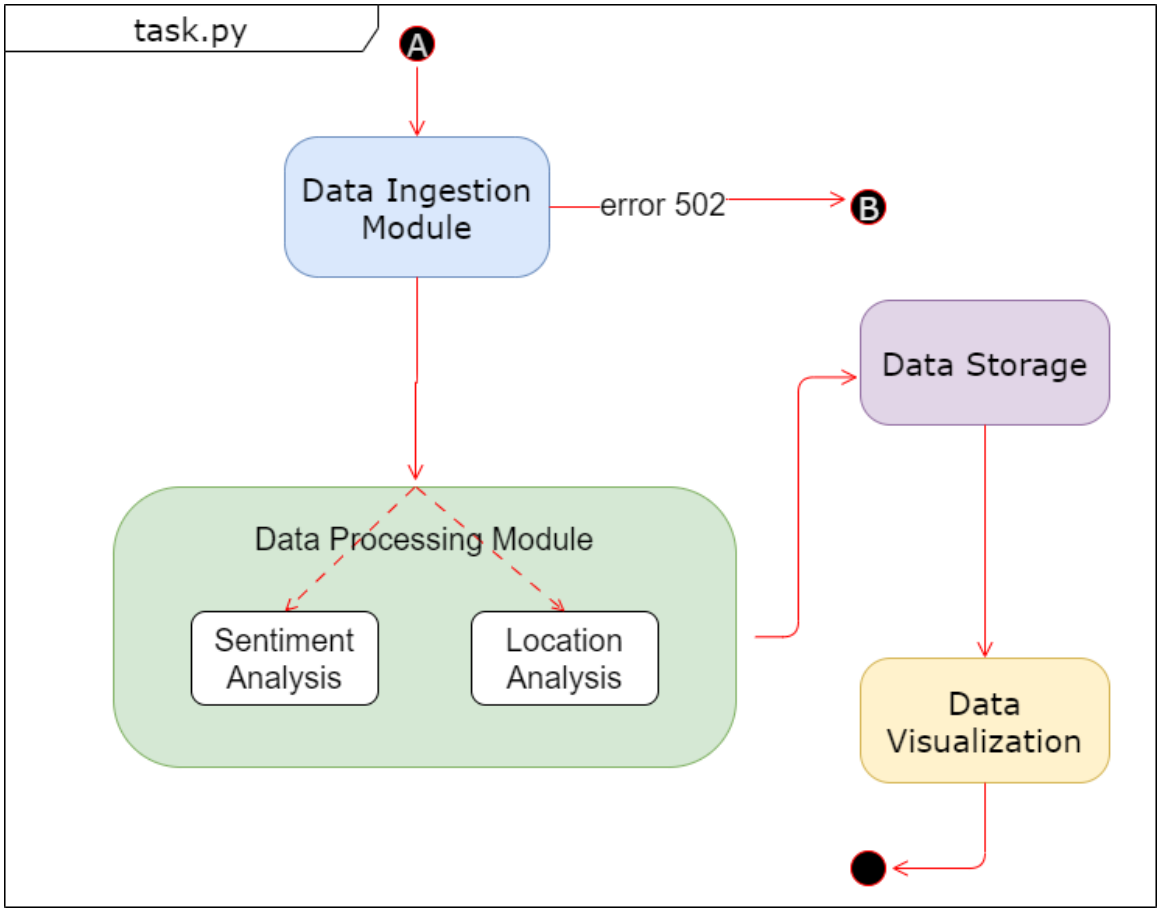


Figure 3.2: State diagram

and the necessary resources have been released, the following transition is made in the state diagram. Once the Data Ingestion Module is finished, we move on to the Data Processing Module. This transition occurs when we get a 200 response from the APIs and we get all the data without any error.

Once all the comments are parsed in Sentiment Analysis, we analyze the geolocation field through Location Analysis. The geographic analyzing state is quite similar to the previous state.

The next state is to upload all the information collected during the previous steps through Storage Module.

In order to better understand the pipeline workflow, we will now discuss the sequence diagram. We can see that the four main modules that we have discussed above are defined with the same colors as the previous diagram.

Starting with the first lifeline of the process which Data Ingestion Module. Data Ingestion Module is the longest process and has to make multiple queries to the data sources API through the scraper so as to get all the information. The result of this process does not

always return a 200 response, which would be the satisfactory one as shown in the diagram, but sometimes we get an erroneous 502 response. This error is due to the multiple requests made to the API in a period of time, which without registering as a developer the limit is about 300 requests per IP per day.

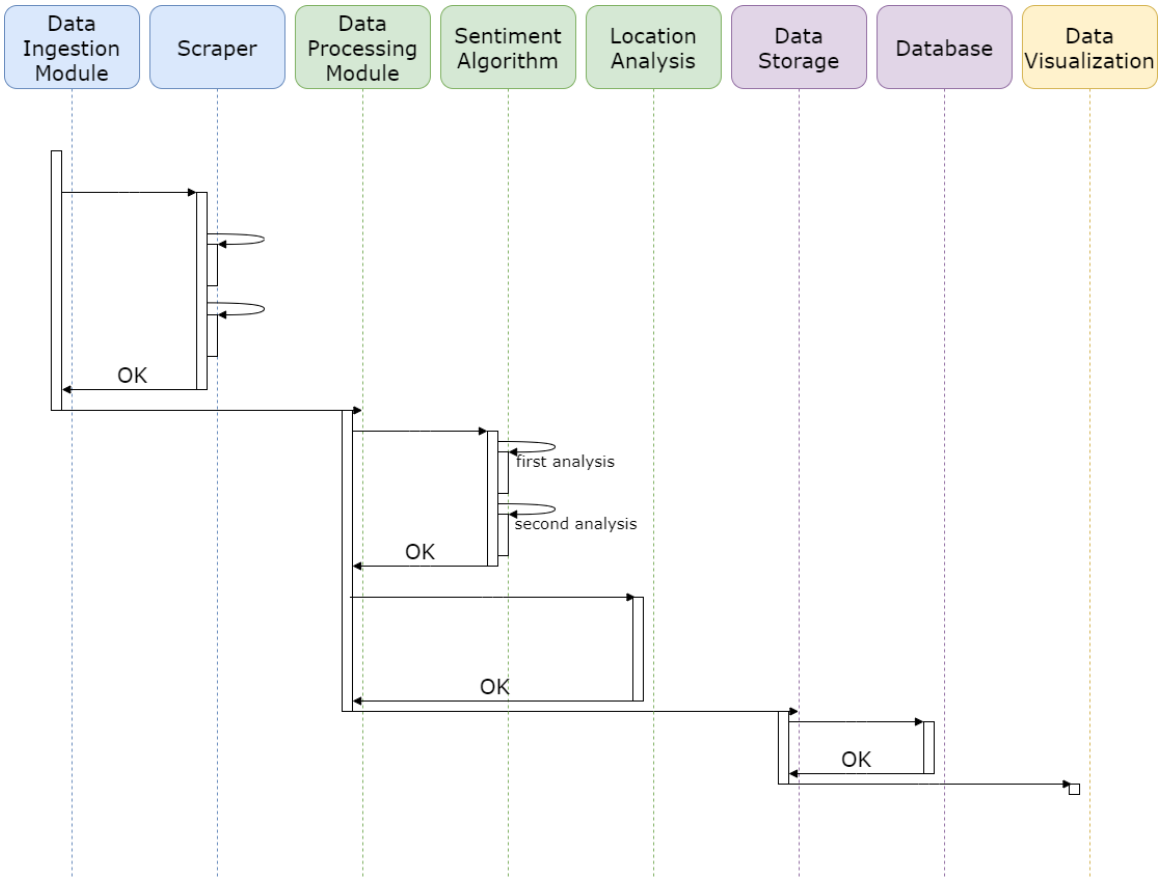


Figure 3.3: Sequence Diagram

Once the interaction between the data sources and Data Ingestion Module has taken place, we move on to the next phase of analysis. As we can see this phase is the other most time-consuming in the diagram. Data Processing Module makes calls to the analyzer API. In our case, the sentiment analysis is performed twice because we perform two types of analysis: analysis through sentiment140 and LIWC. However, if it were performed only once, there would only be one iteration in the Sentiment Algorithm. In addition, after the text analysis phase is finished, the analysis of the geographic location must be carried out. In our case, this is why this is the second longest pipeline, since, per request approximately 140 comments are obtained, having to query Senpy twice to different methods makes about 300 requests each time the pipeline is executed in addition to location analysis.

Next, once Data Processing Module finishes, it notifies Data Storage. This task takes care of uploading the data to the database. This task is not too long.

Finally, once the data is uploaded to the database, it is updated almost immediately on Data visualization.

I will now explain the architecture module by module. I will start with Data Ingestion Module.

3.2.1 Data Ingestion Module

This section will explain the architecture of the Ingestion Module, in other words, how data is collected from the web applications in order to be analyzed. Conceptually, this module is simple and is limited to requesting and getting information from the APIs. In the following image 3.4 we see that the information is obtained from Reddit and StackOverflow, as well as from some other social network if desired, in order to have a document in which to save all this information. Later, this document will be used to build the dashboard.

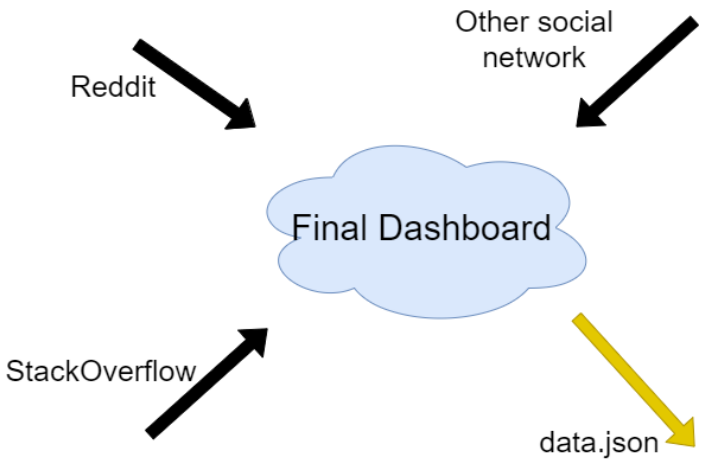


Figure 3.4: Ingestion Module Architecture.

We are going to use StackOverflow as an example, however, Reddit or any other website with a public API would be exactly the same. Therefore, so as to obtain the data we make requests to the API. In order to get a large number of threads on the same topic, it is required to provide as an argument a specific tag. This way we get a large number of comments on the same topic as we see below. In the following example, the owner field has been removed to ensure the privacy of the user's data.

Listing 3.1: StackOverflow example response

```
1 {
2 "tags": [
3 "azure",
4 "azure-logic-apps",
```

```
"azure-eventgrid"
],
"is_answered": false,
"view_count": 29,
"answer_count": 1,
"score": 0,
"last_activity_date": 1686657943,
"creation_date": 1686605324,
"last_edit_date": 1686606827,
"question_id": 76460439,
"content_license": "CC BY-SA 4.0",
"link": "https://stackoverflow.com/questions/76460439/unable-to-process-multiple-events-in-azure-logic-app-with-for-each-loop-and-i",
"title": "Unable to process multiple events in Azure Logic App with &#39;For each&#39; loop and &#39;items()&#39; function"
}
```


It should also be noted that API queries have been configured with a time range. The time field in the StackOverflow API, and also in the Reddit API, is normalized to Unix time stamp, which is the number of seconds elapsed since midnight UTC on January 1, 1970. Finally, as requests are made, the data that is useful or analyzable is saved in the JSON file.

3.2.2 Data Processing Module

In this chapter, we are going to deal with the data processing module. For this, the data processing comes after all the data has been collected from the source data and we already know the fields that need to be analyzed. In this case, we need to analyze two fields, the first one will be the comments from the StackOverflow and Reddit users, although it is obvious that it could be done for any other social network and the second one is the location of the users. Therefore, we are going to split this chapter into two sections. The first one will be the comment processing section, i.e. the sentiment analysis part and the second part will be the location processing.

3.2.2.1 Sentiment processing

As previously stated, we are going to explain the architecture of sentiment analysis of Reddit and StackOverflow comments. The architecture of this module could be summarized in the following image.

We can notice in the image three different analyses as an example, although it can be others. The ones that are shown in the image are the ones that have been used in this analysis. However, Senpy is the project tool that has been used, any other sentiment analysis tool or even a custom algorithm can be used. We will now discuss the output architecture provided by the analyses shown in the image through Senpy. The result of the following

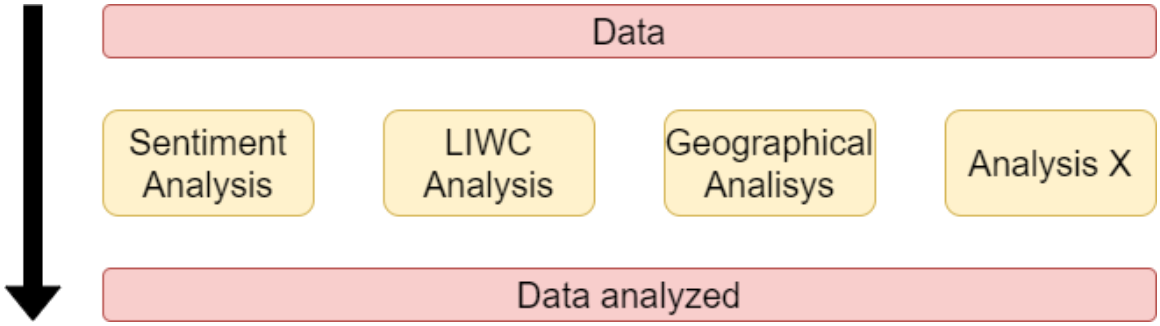


Figure 3.5: Processing Module Architecture.

analysis will be shown as an example, and even if the analysis is performed with the same plugin in a different tool, the output structure may be different, but the result will be the same.

- Sentiment140 is an online sentiment analysis resource that was created by Stanford University. Sentiment140 offers a data mining tool to extract details about Twitter users' points of view in real-time. In order to determine whether a tweet expresses positive, negative, or neutral sentiment. It has datasets with 1,600,000 tweets extracted using the Twitter API [33]. The tool uses NLP and machine learning techniques. This tool has been applied in numerous studies and projects in the artificial intelligence areas, opinion mining, and sentiment analysis.

- LIWC To analyze natural language using linguistic categories, using a computerized dictionary called LIWC (Linguistic Inquiry and Word Count) is a good option. It was developed by James W. Pennebaker and his co-workers at the University of Texas at Austin. The LIWC dictionary includes more than 90 linguistic categories, including emotions, motivation, certainty, negation, tense, and social relations. The dictionary is frequently used to analyze language in a variety of contexts, including newspapers, speeches, interviews, social networks, psychology texts, social science, linguistics, and related fields. The analysis shows the times that different kinds of words have been used in the text, based on the linguistic categories defined by LIWC. For example, the word 'cried' is part of four word categories: sadness, negative emotion, overall affect, and a past tense verb. Hence, whether this word (cried) is found in a text that is going to be analyzed, each of these four categories will be incremented in one point. The number next to each category indicates the number of words in the text that belong to that category. [42].

- emotion-anew: This plugin consists of an emotion classifier using ANEW lexicon dictionary. The Affective Norms for English Words (ANEW) provide a standardized set

of emotional assessments for more than 2,400 English words. It was developed by psychologists at the University of Florida and it is widely used in psychological and social science research to study how words affect our emotions and behavior. [15]. This analysis quantifies emotions by a number. The pleasure dimension is positive at one end and negative at the other end. For example, if the pleasure score is 6.026666666666666666 it would be an emotion with a mainly positive emotional tone. Furthermore, it also measures the arousal level from relaxation to excitement. For instance, if the arousal score is 5.4766666666666666667 it shows a moderate level.

3.2.2.2 Localization processing

Next, we will discuss the geolocation module part. This module is really simple and consists of only three parts: input, call, and output. On the one hand, as previously stated, StackOverflow does not support API maps. This means that StackOverflow allows the user to input the location as he wants. We also need to process the location field so as to get the value in a standardized way. On the other hand, Reddit does not provide users with a field to enter their location, so this module is exclusive to StackOverflow user data.

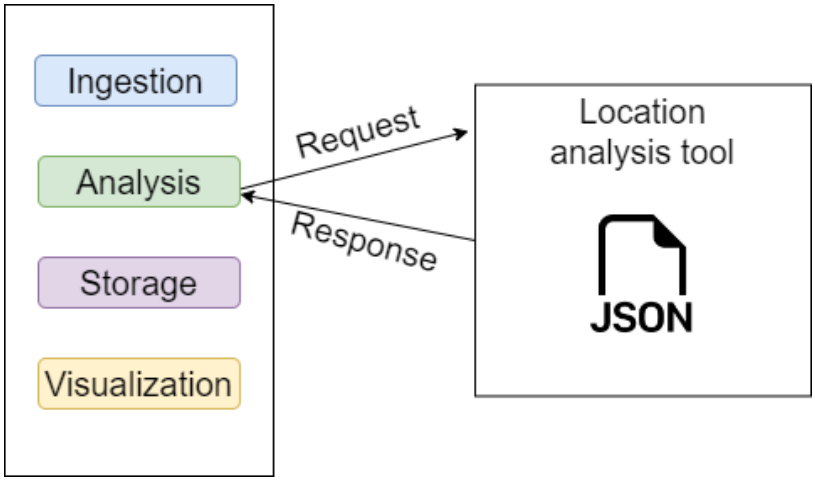


Figure 3.6: API Maps Integration

As mentioned before, the fact that StackOverflow allows to enter the location of the users as a free text field, allows the user to enter unreal locations, based on coordinates or whatever they want. To understand this module we have to be clear about the purpose of this module, and we have to transform the location of the users into latitude and longitude coordinates, country, and city they belong to. The reason why this data has been obtained is for later visualization. on the one hand, the latitude and longitude are necessary for the representation of these locations on a world map. On the other hand, the country and city are necessary to represent them in a Tag Cloud, which later will show what it is and how it

works 4.5

The application that has been used to 'standardize' the data has been OpenStreetMap due to the simplicity provided by its API response although it can be any other that offers the data we are going to see below. However any other can be used. For instance, if we send the "London" location to the OpenStreetMap API we would get the following result. We would get the coordinates in the fields "lat": "51.5073359" "lon": "-0.12765" and the name of the city and country in the "display_name" field. Its value would be "London, Greater London, England, United Kingdom". We also would get the "importance" field, which indicates the relevance of the city we are looking for. The value of "importance" is a number between 0 and 1, where 1 is the highest value, indicating that the place is very important or relevant.

This way we can obtain the data in a homogeneous way for all the locations that users insert in their StackOverflow profile. Finally, the data is saved for later visualization.

3.2.3 Orchestrator Module

In this section, we are going to deal with the architecture of the orchestrator module, which in our project is realized with Apache Airflow. However, this same scheme could be carried out by other technologies such as Luigi or Apache NiFi.

We start by relying on the following architecture diagram 3.7. This diagram shows a PROV Diagram. A PROV-Diagram [59] [37] is a way of structured metadata designed to represent or define the origin or source of information. The PROV specifications are designed to promote the publication of provenance information on the Web, being a standard defined by the W3C PROV (Provenance Ontology). The PROV Diagram provenance model is generic, although there are certain restrictions to represent agents, entities, or activities. Agents are usually represented with a house diagram and orange color, activities are usually represented with a blue rectangle, and entities with a yellow circle. An entity is a real, digital, or even conceptual thing. An activity occurs over a period of time and has an impact on entities. Finally, an agent is something that has some kind of responsibility for an activity, being for example a computer program that causes the activation of another or even a person.

In the diagram, we see the four modules that we saw in the previous diagram (3.1) Now we notice how they are related to each other with the agents. The modules are represented as activities since it is something that happens in a period of time and it interacts with the agents that we see in the diagram. The agents are the software that causes such activity (OpenStreetMap - OSM, Elastic, Senpy) and the user who needs to include the topic he desires to have data. In addition, we see two entities in yellow color: data.json

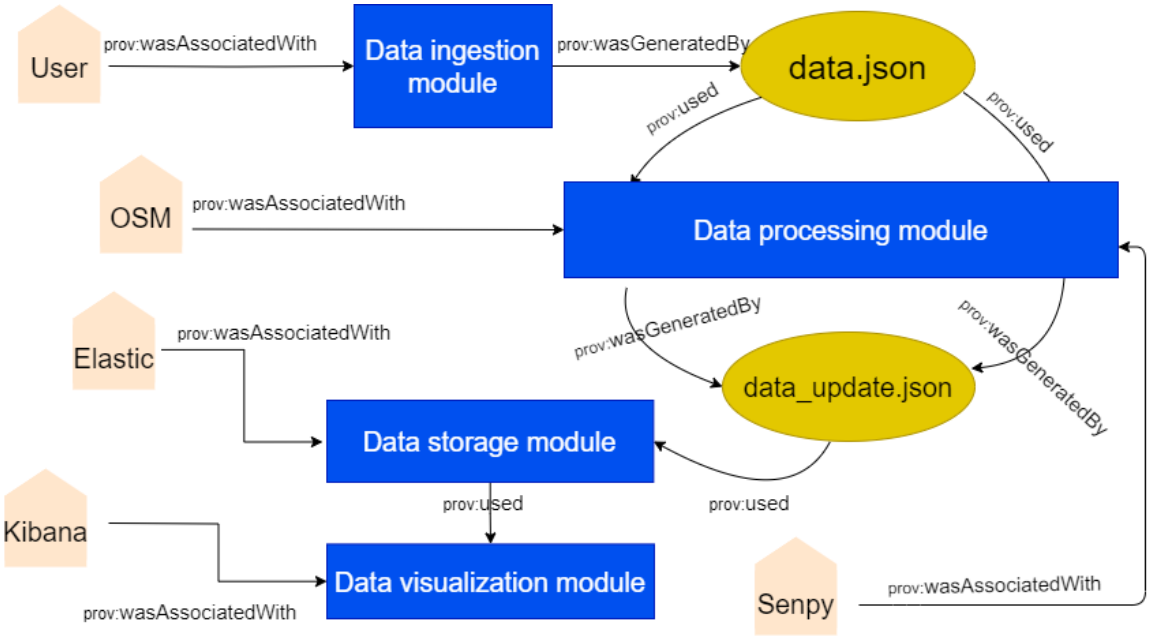


Figure 3.7: PROV-Diagram

and data_update.json. These two entities are the result of the execution of the previous modules in which all the information is stored. Later on, we will see another more specific PROV-Diagram 3.9. It should be noted that the agents that have been introduced in this diagram may be totally dispensable by other systems that produce a similar result. Next, we are going to explain the relationship between the entities, agents, and activities.

- prov:wasAssociatedWith is a relationship used to indicate the association between activities and entities or agents and activities/entities or between entities.
- prov:used is the beginning of the use of an entity by an activity. Before the entity uses the activity, the activity had not started using this entity. [59]
- prov:wasDerivedFrom is used to indicate that an entity is derived or generated from another entity. Before to the execution of this activity, the entity did not exist before the generation and can be used after this generation. [59]
- prov:wasGeneratedBy is the completion of an activity by generating an entity by the activity. The entity did not exist before the generation and becomes available for use after this generation. [59]

Next, we are going to explain the flow using the diagram. In our case study, the workflow of the orchestrator module has several tasks. The first one is an activity that obtains data from the APIs and generates an input file called "data.json", so the relationship it has with

the agent is prov:wasAssociatedWith and prov:wasGeneratedBy with the file. This file is used by the following activity, which is the data processing module. In this module, we see two arrows, one for each agent involved in this activity. The result of this activity is the generation of a new file called "data_update.json", so the relationship of this entity with the activity is through prov:wasGeneratedBy. In the penultimate step we see that once the entity of "data_update.json" is available, it is launched to the Data storage module activity. Finally, we see that the visualization module uses the previous module for data representation.

3.2.4 Storage Module

Finally, through all the data obtained, we have built in the repository a kind of Data Lake with a multitude of data sources as we can see in the following picture 3.8. A Data Lake is a storage repository that can store large amounts of data that has been ingested after an Extract/Transform/Load (ETL) process. This data can be either structured, semi-structured, or unstructured. However, although there is one point that by definition Data Lake does not meet, we are going to treat it as such. That point is that Data lake is a popular term for data stored in a Hadoop object store, while ElasticSearch stores data in Lucene indexes on a cluster of servers.

Therefore, as we can see in the image we have multiple sources of information. As we have mentioned in the previous point we have OpenStreetMap (OSM in the diagram), information coming from LIWC, onyx, and marl analysis and of course we have the one coming from StackOverflow APIs (SO in the diagram), and Reddit.

In the following diagram 3.9 we have the second part of the diagram mentioned in the previous section of this chapter 2.5, the Provenance (PROV) diagram. Going into detail on the structure of the data that is stored, we will rely on this diagram. Thus, we will only explain the new concepts that we can see in the figure.

- rdfs:Literal is a class of the RDF (Resource Description Framework) ontology language that is used for text and integers. These values can include text strings, numbers, dates, and any other values. [56]

- xsd:Integer is a data defined in the XML Schema Standard (XSD) that is used to represent integer values.

- xsd:Float The float data type accepts decimal values, which may be followed by the power of a value. For example a valid number for the representation of xsd:Float would be From -3.4028234663852886e+38 to -1.1754943508222875e-38. [56]

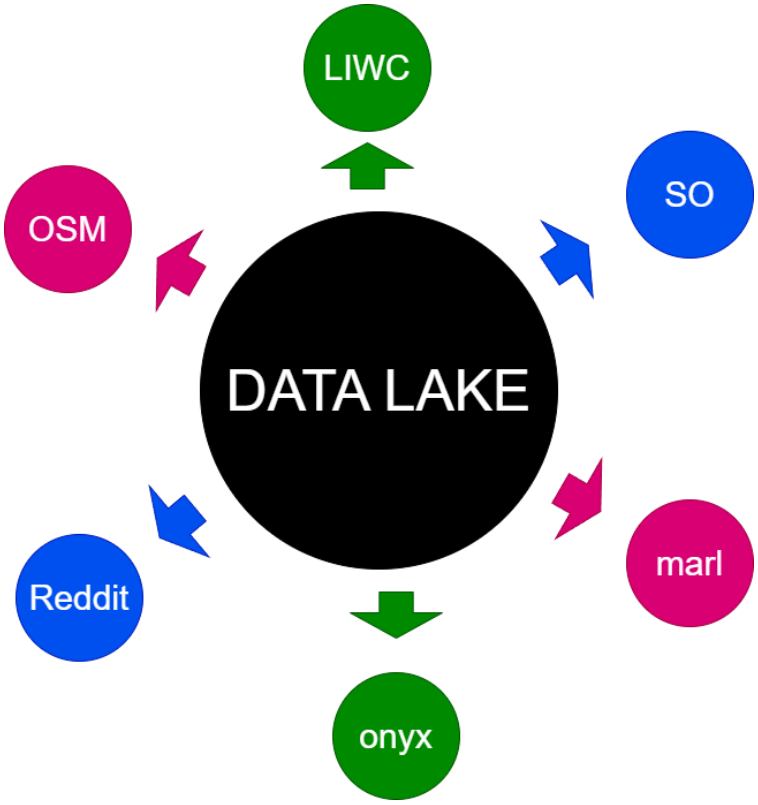


Figure 3.8: Data Lake.

- xsd:Boolean is also a data type defined in XML Schema that represents Boolean values, i.e. values that can be true or false.

In the graphic, we can appreciate the structure of the file containing the values. To guide the reader, two parts have been highlighted in colors and the agent that produces the values has been added. The letters in red describe the name of the field and the letters in blue define the type of value the field has. In some cases, for simplicity of the diagram, the number of fields that we obtain from the agents has been reduced, for example, in the API fields and in the 'LIWC:result' fields, since it is not the purpose of the diagram to show all the fields but to show their architecture. All the information related to the fields will be shown in the first section of the chapter 4.

3.2.5 Visualization Module

Regarding visualization, it is necessary to focus on Kibana technology. Once the data is in Elasticsearch, the data is sent to Kibana thanks to the interaction between these two programs. Therefore, once Kibana is able to read the data it is important to define an index pattern. The index pattern is just the configuration that allows Kibana to interpret the data stored in Elasticsearch. The configuration of the index pattern allows Kibana to

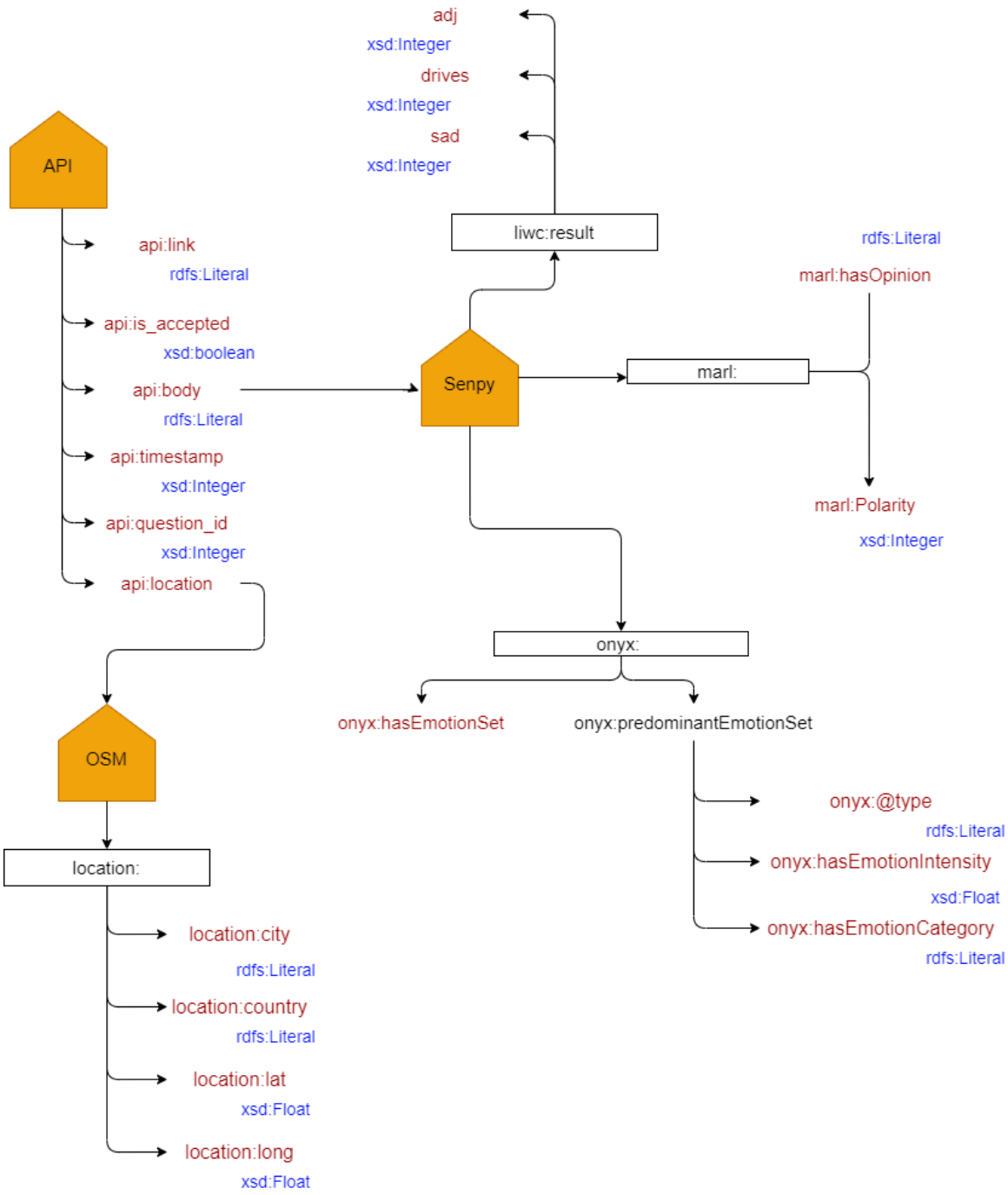


Figure 3.9: PROV-Diagram data.json and data_update.json

provide an interface with the user of the application so that the user can make the desired configurations.

The most common types given by Kibana to Elasticsearch variables are predefined and are as follows:

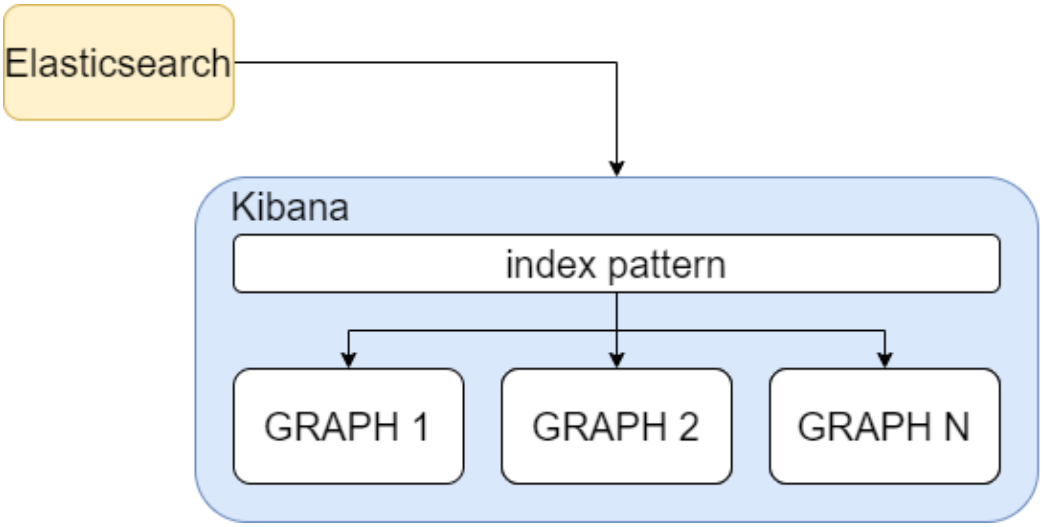


Figure 3.10: Kibana's architecture

- Keyword: This field is used for text data. The keyword data type is very useful when the user needs to perform exact matches. For example, if the user wants to search by the car brand field and inserts "Range", he will not get the brand "Range Rover" as it is not an exact search.
- Text: This type is also used for text data. However, if we continue with the example above, in this case, it would find the value "Range Rover". The text type is parsed at the time the question is asked and tokenized.
- Date: This type is used for date and time data. Kibana allows you to set a timestamp if it is a variable within the Elasticsearch index, which controls the Dashboard (or whatever is applicable). Therefore, it allows searches by time.
- Numeric: This type is used for numeric data and there is no difference between integers or decimals.
- Boolean: This type is used for Boolean values of true or false.
- geo_point: The geo_point field is a JSON made up of two values of latitude and longitude. It is a mandatory field if you want to use map diagrams.

Once the index pattern is configured, we will be able to generate the graphics or visualizations we desire. The configuration of the charts will be discussed in the next chapter, which will deal with the implementation of these modules.

CHAPTER 4

Case study

This chapter will detail how the implementation of the different modules that build the architecture presented in the previous chapter has been carried out. Finally, all the diagrams that have been made will be presented and explained. The explanation of the graphs, therefore, will be accompanied by a summary of the results obtained in each of the diagrams.

4.1 Implementation of the architecture

In this chapter, we will discuss how the modules of the previous chapter have been implemented. We will discuss specifically which technologies from Chapter 2 have been used for which modules, as well as the necessary configuration and their particularities. We will discuss what data have been chosen, what types of analysis have been carried out, and what configuration has been selected both in the dashboard and in the previous systems. In addition, the data obtained by the system will be provided: the number of comments, the ratio of comments obtained by the platform, and their geographical location. Finally, the datasets of the analysis will also be displayed, showing the overall results, the results by topic, and the results of all the analyses performed.

Before presenting the implementation of the modules we are going to expose the following picture. In this figure 4.1, we can see the first architecture graphic in which we implement the technologies we have used. In this chapter, we will first explain the orchestrator or Airflow module, in black color. Next, we will explain the module in blue color which is the data ingestion module. We will continue with the data processing module in which we will use Senpy and OpenStreetMap. Once the green module is finished we continue with the purple module which is where we use Elasticsearch to store the data in the database. Finally, we finish with the visualization module in which we use Kibana to represent the data we have stored in Elasticsearch.

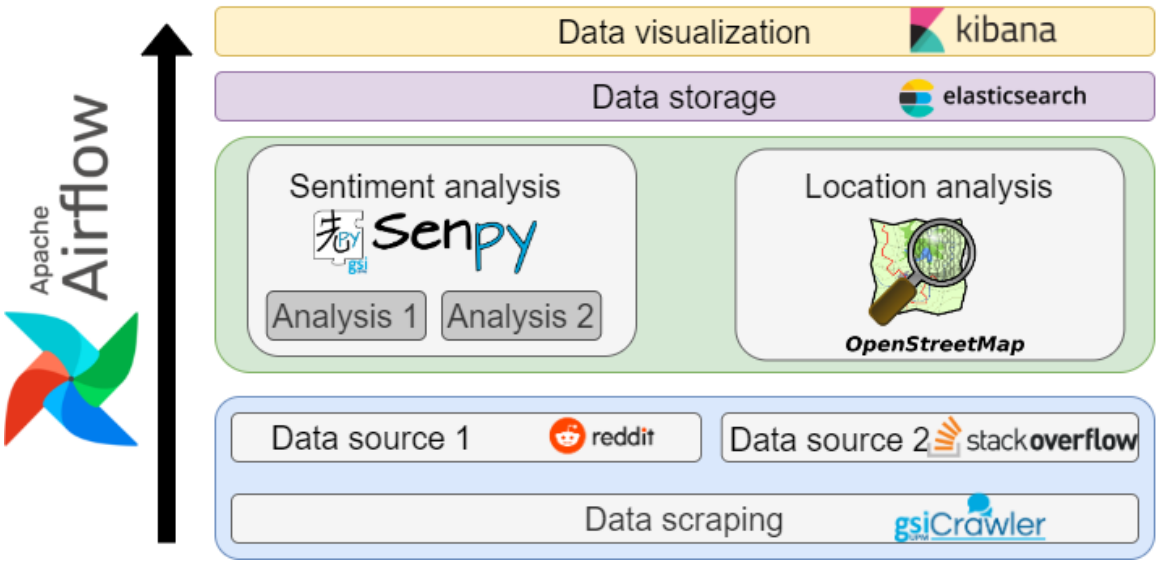


Figure 4.1: Architecture with the technologies used.

The implementation of these modules in the docker containers will be represented through the deployment diagram. The deployment diagram is responsible for showing the configu-

ration of the components and the physical layout during implementation [22]. Deployment diagrams are mainly composed of nodes, artifacts, devices, and communications between them. This diagram continues with the colors that we have been associating with each module: blue for the Data Ingestion Module, green for the Data Processing Module, purple for the Storage Data, yellow for the Visualization Module, and black for the Orchestrator Module.

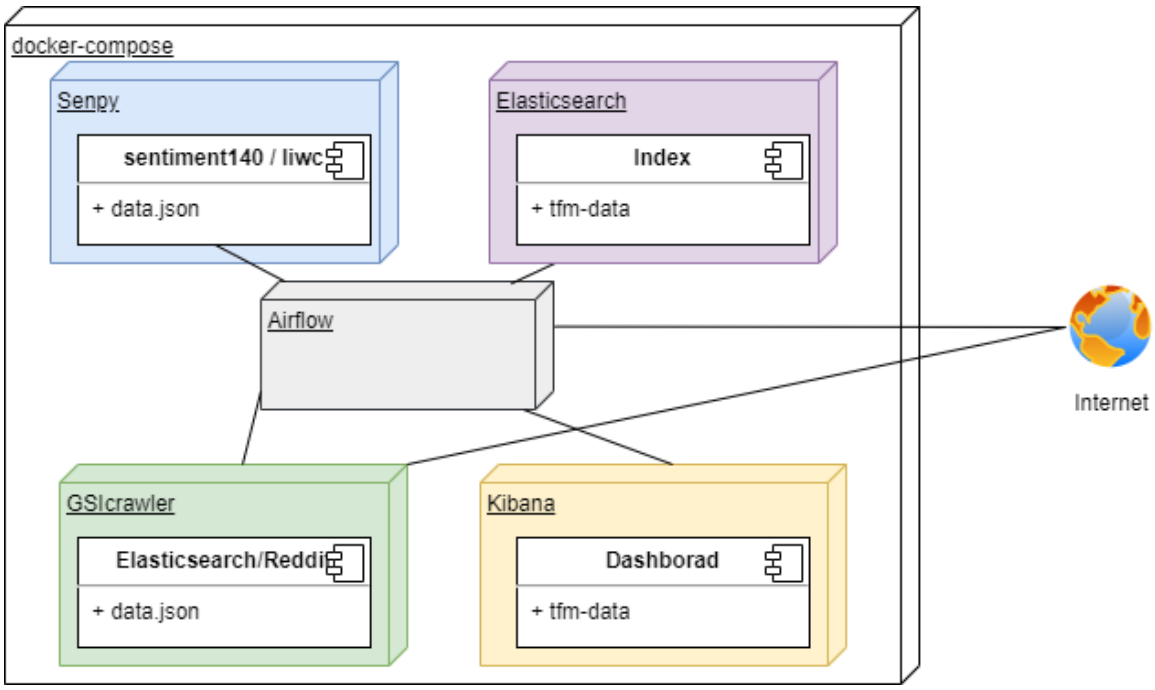


Figure 4.2: Deployment diagram

First, in deployment diagrams, we have the nodes. Nodes are represented as a cube and consist of physical entities running one or more components as we can see. A node can be either a hardware or software element, and in this case, it matches, since everything that is executed in the docker-compose is executed on the same machine except for the requests that are made to the internet. The docker-compose is composed of several instances that make up the nodes we see below.

On the one hand, we have the artifacts of the diagram. Artifacts are concrete elements that are built by a development process. The artifacts are represented by a square and they can be for example libraries or files that can be executed. In our case, we have three different artifacts: data.json and tfm-data repeated twice. Although tfm-data is duplicated it refers to two different artifacts with the same name. The first one is the Kibana dashboard, where the graphs are represented and we have the front end of the application, and the other one is the Elasticsearch index (tfm-data) where the information is stored.

On the other hand, we have communication between the nodes. This is represented by

the solid line between two nodes that shows which nodes have communication with each other.

Finally, we would have the devices, which in our case is only one node. Devices are used to represent a physical computational resource in a system. An example of a device is an application server. However, for this case, we would not have any represented in the diagram.

It should be noted that this diagram is specific to this development and has been made this way due to the simplicity of incorporating new dockers as the Sprints progressed. Due to the functionality of Docker, you could have had all containers running in a single container. In the following sections, we will explain in detail each of the modules we have presented.

4.1.1 Implementation of Orchestrator Module

The first module we are going to present is the orchestrator module. Explaining this module first is essential to understand the implementation of all the modules and also to understand how the Airflow pipeline works. However, before presenting Airflow's block diagram and its explanation, we need to configure some environment variables, for example Senpy, Elasticsearch or GSIcrawler. The configuration of these variables in the DAG would be as follows:

Listing 4.1: DAG environment variables

```
ES_ENDPOINT = Variable.get("es_endpoint", default_var='http://localhost
    :9200')
CRAWLER_ENDPOINT = Variable.get("crawler_endpoint", default_var="http://
    localhost:5000/api/v1/scrapers/")
SENPY_ENDPOINT = Variable.get("senpy_endpoint", default_var="http://
    localhost:8007/api/")
```

In this configuration, we configure the default ports needed by the services that we are going to use in the pipeline. However, the services could have been configured on other ports of your choice, in case those ports were already occupied by other services.

Once we have the environment variables configured we can proceed to integrate the different services through the Airflow modules.

In this photo of the Airflow diagram 4.3 we can see six different blocks in two different colors white and green: create_index (the only one in white), download_data_reddit, download_data_stackoverflow, geo_data, analyze_data, and storage_data. The Airflow is in charge of executing the pipeline and chaining one module after another. So when we start with the download modules, the next module to be executed is geo_data because analyze_data cannot be executed due to the dependency with geo_data. Finally, the pipeline

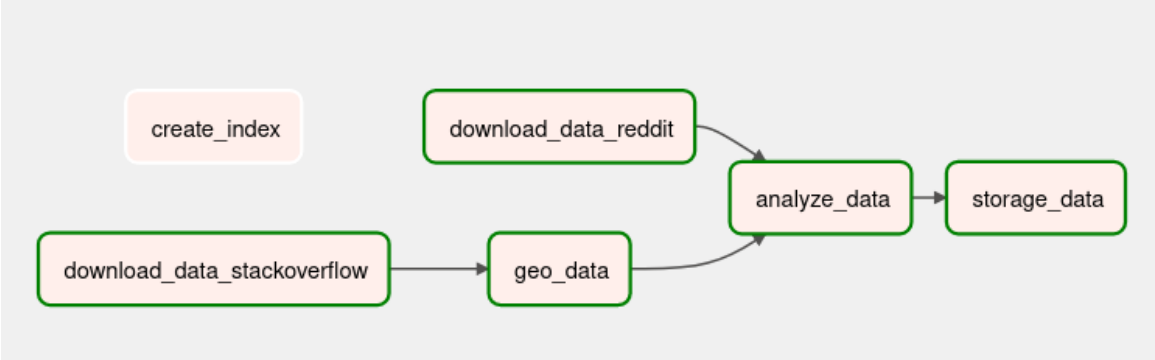


Figure 4.3: Airflow diagram.

ends when storage_data is executed.

create_index: This module is directly connected to Elasticsearch. In create_index the index structure to be used is defined and created. Therefore, when defining the index in this module, it would be enough to be executed only once. In order to give more functionality to this module, the option to delete the Elasticsearch index data has been introduced. Therefore, the execution of this module is optional and it will be executed whenever you want to create a new index or delete the content of the one already in use.

download_data_reddit and **download_data_stackoverflow**: We define the two modules simultaneously because they are analogous. Download data Reddit and StackOverflow is a module that makes the request to the Reddit/StackOverflow API through GSIcrawler. This will be explained in the section 4.1.2

geo_data In this module we get the location of the StackOverflow user, since this field is not possible to get it from Reddit because it is not available. It is worth remembering that the StackOverflow user location field is a free text field that is not related to any map API, so it can be filled with any data whether it is a real, fictitious or not real location. In this field you get the latitude and longitude of the entered city, the country, and the city, town or village. The latitude and longitude coordinates will be stored as a geo_type variable in Elasticsearch and will be reflected in the Kibana map. This implementation will be explained in the section 4.1.3

analyze_data The analyze_data module is responsible for analyzing the comment field. The comment to be indexed in the sentiment analysis tool has undergone 'cleaning' processing. Non-alphanumeric characters have been removed as they could alter the result. The comment to be indexed in the sentiment analysis tool has undergone 'cleaning' processing. This also will be explained in the section 4.1.3 because it has been considered part of the data analysis.

storage_data In this software module the upload to the database is performed. In this

module, the information that has been processed during the whole pipeline is uploaded to the Elasticsearch indexes. This section has been evaluated in three ways. The first option considered was to configure an index for questions and another index for answers, thus having a logical separation of questions and answers, since at first they would have different attributes as we will see later on. The second was to create a specific index for each platform (Reddit and StackOverflow). This way had the advantage of being able to differentiate the origin of each thread without the need to add a field to the data to establish it. Finally, the option chosen was creating a single index and a field that establishes that the data comes from Reddit or StackOverflow and also includes a field to differentiate whether it is a response to a thread or question. The data upload was performed through the Elasticsearch library for Python. This implementation will be explained in depth in the section 4.1.4

Regarding the dependencies of the pipeline with other methods and classes, we will reflect them in the class diagram. A class diagram describes the structure of a system by showing the classes of the system with all their characteristics (attributes, methods and requirements). It is also very useful to see the direct relationship between different classes that make the system work. In the diagram, you can notice the dependencies that our class has with other classes, the inheritance it has, interfaces, objects, attributes, and more.

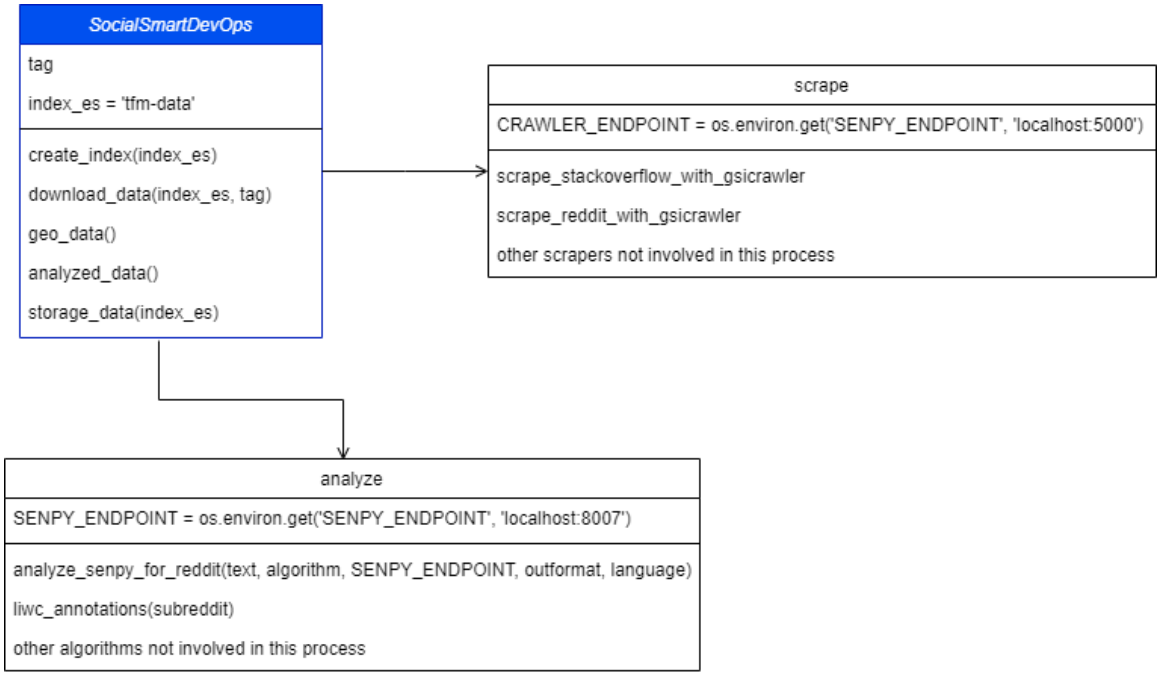


Figure 4.4: Class Diagram.

In our diagram, to simplify the reading of the graph our class is in blue. A class diagram, as we can see, always has 3 basic sections:

- Upper section: Contains the name of the class to be represented. In our case, it is

with the name of 'SocialSmartDevOps'.

- Middle section: Contains the attributes of the class. In our case we can appreciate the attributes under the name of tag and index_es. The tag attribute is empty since it depends on the technology that the user wants to analyze, although, for example, it could be Azure, Jenkins or Terraform.

- Lower section: Contains the methods of the class. In our case we can see five methods: create_index, download_data, geo_data and analyzed_data and storage_data

It is also worth mentioning the relationship that our class has with the 'scrape' and 'analyze' classes. They are related under a unidirectional association. The unidirectional relationship is that one class is and interacts with the other when needed. The one-way association is drawn with a straight connecting line pointing to an open arrowhead from the class that needs the other's dependency.

On the one hand, the GSIcrawler's scrape class is directly connected to the scraper that will be shown later 4.1.2. On the other hand, Analyze request is in charge of making the request within Senpy to query LIWC and Sentiment140 in order to obtain the result based on the algorithm parameters.

Now that we have explained the operation of each of the methods and we have explained the dependencies and how they work, we are going to explain the rest of the modules one by one in detail in this order: Implementation of Data Ingestion Module, Implementation of Data Processing Module, Implementation of Storage Module and Implementation of Visualization Module.

4.1.2 Implementation of Data Ingestion Module

This module will explain the adaptations that had to be made to make the GSIcrawler module work. First of all, in order to create a data ingestion process to obtain all the desired data, a module has to be created in GSIcrawler. This module is a script executed by GSIcrawler. It must have a unique name and is written in Python. In order to get the data from Reddit and StackOverflow, two different scripts have been created. As previously mentioned, the application GSIcrawler already had a module to obtain Reddit data, however, it was decided to create a parallel one in order to adapt it to the needs of this project. In the following section, we are going to discuss the StackOverflow script in detail and later we will discuss the particularities of the Reddit script.

StackOverflow Scraper

The StackOverflow API allows us to obtain almost all web data, such as questions, answers, question tags, user data (location, name, id...), thread comments and answer scores, and so on. The StackOverflow API allows developers to query the data in real-time. The StackOverflow API uses OAuth 2.0 for authentication and authorization, ensuring data security and user privacy. OAuth 2.0, which stands for "Open Authorization", is a standard designed to allow websites or applications to access resources hosted by other web applications on behalf of users. The StackOverflow API allows you to filter searches so as to get the results you are looking for. You can set the request using parameters such as date, order, keywords, and more. Therefore, the StackOverflow API becomes the ideal tool to obtain all the data we need for the analysis of the comments. We will give details of the specific data later, but you can already sense some of the data that will be key to the analysis.

The scripts that have been developed in GSIcrawler, consist of a series of requests to the StackOverflow's API. A very basic and summarized example of what the script would look like is the following:

Listing 4.2: Example StackOverflow Script

```
# Make HTTP request to Stack Overflow API
responseQuestion = requests.get(f"https://api.stackexchange.com/2.2/
    questions/{question_id}?site=stackoverflow&filter=withbody")

responseAnswer = requests.get(f"https://api.stackexchange.com/2.2/
    questions/{question_id}/answers?order=desc&sort=activity&site=
    stackoverflow&filter=withbody")

# Check the status of the response
if responseQuestion.status_code == 200:
    # We get the content of the response in JSON format
    dataQuestion = responseQuestion.json()
    questionJSON = {}

    questionJSON["link"] = dataQuestion["items"][0]["link"]
    questionJSON["title"] = dataQuestion["items"][0]["title"]
```

In these few lines, we can see the core of our script. To begin with, we can see that the request to the StackOverflow API is done by changing the question_id variable. This is due to the fact that in order to obtain the maximum number of comments on the same topic, the API query is made for all the threads that have a specific tag so that the scraper is only limited to obtaining the data of the same thread. This general request to the API

will be shown in this same section later on once the integration of the script in GSIcrawler has been explained. Next, we can see how we check the status of the response and, in case it is affirmative (status == 200) we save in a variable the response in JSON format and we obtain all the fields that we need, in this example, we can see that we receive the link and the title. We can also appreciate that in the request to the API, two requests are made. The reason why two requests are made is that the way the StackOverflow API is set up the answers to a given thread and the questions are in different URIs. This is why the process has to be performed twice.

Some particularities have been created inside the scraper in order to satisfy the structure we have created, for example with the localization. The location, as mentioned above, is not a required field in StackOverflow which has been a problem in the development as we will see now. Before obtaining the location data, we had to check if this data exists by consulting the user's profile since this field does not appear in the data of the answer or question of a thread, which implies that we have to make a new query to the user's profile. In addition, we have had to check that the location inserted by the user is correct. There have been cases in which the user's location contained non-alphanumeric digits and has meant different errors when converting this data to a JSON format (JSONDecodeError). It should also be taken into account whether the user who has written the question, has unsubscribed his account in StackOverflow. In this case, it is not possible to get the user's data because StackOverflow has removed the data from the platform. Therefore, the adaptation that has been made for the localization has been the following:

Listing 4.3: Location StackOverflow Script

```
    # We check if the user has unsubscribed.
    if "user_id" in dataQuestion["items"][0]["owner"]:
        userQuestion = dataQuestion["items"][0]["owner"]["user_id"]
        # We consult the profile to get the user's location.
        responseUserQuestion = requests.get(f"https://api.stackexchange.com
            /2.3/users/{userQuestion}?order=desc&sort=reputation&site=
            stackoverflow")
        dataUserQuestion = responseUserQuestion.json()
        if responseUserQuestion.status_code == 200 and "location" in
            dataUserQuestion["items"][0]:
            locationUserQuestion = dataUserQuestion["items"][0]["location"]
            print("The location in the question is " + str(
                locationUserQuestion))
        else:
            locationUserQuestion = ""
            print("No localization in the question")
```

In the chart above we can see how we check that the user of the question exists and then we make the request to the user of the question verifying that the field "location" exists in the API. If it does, it is saved in the "locationUserQuestion" variable and otherwise it is saved empty. For the user of the answer, the structure is analogous.

Finally, the way to join the data of the questions and the answers has been to create a JSON. We have created two variables that contain all the information, one under the name "answer" and the other under the name "question". Afterward, they have been saved in another variable with the following structure data = { "answer": answer, "question": question }. In this way, as we have question_id field, we are able to get all the data related to a question in single JSON.

Reddit Scraper

Reddit's API, like StackOverflow's, allows us to get almost any data related to subreddits and subreddit threads. However, as far as user data is concerned, this API does not offer as much data as StackOverflow's API does. In the StackOverflow API we have collected, if available, the user's location for further study. However, in the Reddit API we have not been able to take this data and this is because Reddit is a platform where privacy is valued, so many users choose not to share their location. Therefore, it is not possible to obtain a user's location on Reddit through the Reddit API. Before going into the details of the Reddit scraper, it should be noted that there are several methods of scraping the information from the Reddit API.

- The official Reddit API: It is the way that has been used in this work to obtain the information. The URL from which the requests are made is *https://api.reddit.com/r/*. From this address, we can get all the data of a given subreddit or user.

- Pushshift [14]: Pushshift is a scraper platform that has been collecting Reddit data and analytics for eight years. This application is updated in real-time and provides its dataset for those who wish. Pushshift's tools are able to perform filtered searches and with aggregations. Using Pushshift has several advantages. First of all, is that there is no limit to the number of requests and you can get as much data as you want. Probably, the second advantage is that it does not require authentication and allows you to make requests that the Reddit API does not allow. The last and most important one is that Pushshift stores historical Reddit data that is not available through the official API.

- PRAW (Python Reddit API Wrapper) / PSAW (Pushshift API Wrapper): PRAW and PSAW are Python libraries that interact with the Reddit and Pushshift API re-

spectively. PRAW simplifies the process of authentication, exception handling, paging and searching in Reddit. PSAW focuses on more advanced searches on Reddit and provides a better ability to search through large volumes of Reddit data.

- snoowrap: It is a JavaScript library that allows you to get data from the Reddit API. It's a simple JavaScript way to get data, for example, as snoowrap's Github [49] says, the method to get a user's profile is simply getUser(), and the method to upvote something is simply upvote().
- Third applications that rely on the previous ones such as bmai.dev/Reddit or camas.unddit.com. Both are web applications that provide a user interface that performs queries via Pushshift.

Once all the options that we have discussed have been evaluated, the first option has been chosen due to the ease of integrating said requests into the GSIcrawler application. Therefore, following the StackOverflow case, queries have also been made to all the threads of a given tag in order to get the maximum number of different threads on the same topic. In this response, the comments are sorted by the most recent. Therefore, once we get the comment we are able to get the question_id. Once we have located the threads, we have made the query to that particular thread in order to obtain all the comments related to that thread. Once we get all the comments from that first query, we are able to get the last comment and perform the query to get page two from the API. This was also done in the StackOverflow scraping process. However, the first difference between scrapers is that the general query (the tag query), in Reddit, is performed inside the GSIcrawler, allowing the scraper to obtain a large number of comments at once.

In the following listing, we can see represented what has just been explained. First, we query the first page. Once we get the API answer we store the user identifier of the last comment of that API page in "last_name". Therefore, when the while loop returns on the next iteration, we will get the next page of comments and so on up to page 9. The reason why the while loop only goes up to 9 is that Reddit only allows this iteration to be done 10 times (9 plus the first one)

Listing 4.4: Reddit scraper

```
while i < 9:
    url = f"https://api.reddit.com/r/{subreddit}/comments?&limit=500&after={
        last_name}"
    response = requests.get(url, headers={"User-Agent": "Mozilla/5.0"})
    data = response.json()
    try:
```

```
        last_reddit = data["data"]["children"][-1]
        last_name = last_reddit["data"]["name"]
    except:
        return hilo
```

This allows us to get a large number of comments from a single query. This number varies depending on the responses to the comments. For example, 1,100 comments have been obtained in Python and 600 in Terraform.

The second difference is what we have just discussed, the responses to the main comments. This implies that the API architecture is something more complex to manage. The following listing 4.5 shows the outline of the Reddit API.

Listing 4.5: Reddit API

```
"data":{
   "children":[{
         "data":{
            "subreddit":"Python",
            "replies":{
               "data":{
                  "children":[{
                        "data":{
                           "body":"Using calibre python api.\n\nhttps://manual.
                               calibre-ebook.com/db_api.html#calibre.db.cache.Cache.
                               add_books",
                           "permalink":"/r/Python/comments/14c5sg2/
                               sunday_daily_thread_whats_everyone_working_on/jokaf6i/
                               ,
                           "created_utc":1687075292,
                        }
                     }
                  ],
               }
            },
            "created_utc":1687074821,
            "body":"How do you save the books in the calibre library?",
            "link_id":"t3_14c5sg2",
            "permalink":"/r/Python/comments/14c5sg2/
                sunday_daily_thread_whats_everyone_working_on/jok9vnh/",
            "created":1687074821,
         }}],}
```

In this example we can notice how the main comment field "How do you save the books in the calibre library?" has received a response specifically to that comment and not to the thread, being the response "Using calibre python api...". If the user had replied to the general thread, this comment would be after the main comment and would not be a "child" of it. Therefore, in order to obtain all these responses, we have checked if the "children"

field exists within each comment.

In this listing, we also see that the thread identifier is preceded by a "t3_". This is because the Reddit API uses indicators to differentiate them from each other: "t1_" identifier represents a comment, "t2_" identifier represents a user account, "t3_" identifier represents a thread, "t4_" identifier represents a private message, "t5_" identifier represents a subreddit.

The third difference between scrapers is that in StackOverflow there were two queries: one for the question and one for the answer. However, as we just discussed in the previous paragraph, the answers and the question of a Reddit thread are in the same API URL.

Another difference from StackOverflow is that the Reddit API is much more complicated to manage. The way Reddit works, it is possible to reply to a specific comment. This use case, in the API is represented as an entry to the comment ["entries"] making the extraction of comments quite complicated. For this, a series of "ifs" conditions have been set to query if a particular comment has replies.

Integration in GSIcrawler

This next step is applicable to both the Reddit script and the StackOverflow script. We are just going to comment the StackOverflow script but the process for the Reddit script is exactly the same. So as to integrate the script into the GSIcrawler system there are a number of steps to be performed. The first step would already be done, it would be the development of the scraper. Then we have to define the scraper in the different places so that the GSIcrawler frontend integrates it correctly. This means that the scraper and requests the parameters that are necessary are requested by GSIcrawler. To start with, we have to define the scraper in the task list of the GSIcrawler package named tasks.py. The task has to be named as for example stackoverflow_scraper(question_id), indicating the necessary parameters, in our case question_id. For this definition, two decorators must be defined: @celery.task and @crawler. The decorator @celery.task shows that it is going to be a task that can be executed in parallel using Celery [17]. The decorator @crawler indicates that it inherits the basic data from the scrapers (number of comments to return, output format...).

Finally we have to include the script definition in the YAML of the application. In this environment, we can see the parameters that are mandatory, as for example question_id and other parameters that are inherited as the output, index, nubmer, etc. At the end of this YAML we define the different answers that the application should give. In this case we have two types of response. The responses are in HTTP and HTTP 200 means that the scraper execution has been successful while the HTTP 202 response means "Accepted" which indicates that the request has been accepted for execution, although it has not been

executed yet.

Listing 4.6: Scraper definition in YAML

```yaml
/scrapers/stackoverflow/:
  get:
    operationId: gsicrawler.controllers.scrapers.stackoverflow_scraper
    description: Run a scraper to search in Stackoverflow
    tags:
    - scrapers
    parameters:
    - name: question_id
      schema:
        type: string
      in: query
      description: Topic to search in Stackoverflow
      required: true
    - $ref: '#/components/parameters/output'
          .
          .
    - $ref: '#/components/parameters/number'
    responses:
      200:
        $ref: "#/components/responses/done"
      202:
        $ref: "#/components/responses/pending"
```

Scraper's integration in Orchestrator Module

This step is also applicable to both the Reddit script and the StackOverflow script. To be able to make the call in the orchestrator module, it must be previously defined in the set of scrapers. This definition is the one that will allow us to call the GSIcrawler from the orchestrator. The definition is a simple function that in our case has been called scrape_stackoverflow_with_gsicrawler. This function has several attributes that are related to those that have been previously configured. The parameters are the following:

- scrapper: Through the 'scraper' field we can identify which data collection method is been called from GSIcrawler. In this field, we are able to select both scrappers: Reddit and StackOverflow.
- query: indicates the type of search to be performed, in our case this parameter will be of type "search".

- crawler_endpoint: crawler_endpoint indicates the port where the GSIcrawler is hosted which in our case is port 5000 of the localhost.
- subreddit or question_id: This field depends on the platform used. If we use Reddit's API the *'subreddit'* is requested in GSIcrawler meanwhile in StackOverflow's API the question_id need to be provided to GSIcrawler. On the one hand, the subreddit has been used for reddit. A subreddit is an online community within the Reddit platform. Subreddits are specific sections or categories; for example Azure or Jenkins. On the other hand, the question_id field is the unique identifier used by the platforms to determine the questions or threads. This identifier is a code formed by numbers in the case of StackOverflow, being the assignment of these codes completely random. Therefore, in order to collect Azure-related threads on StackOverflow, a request is made to the StackOverflow API for the topics that have had the most recent activity with Azure as a tag.
- output: GSIcrawler gives the option to configure the output as JSON (in file format) or upload it directly to Elasticsearch. This work has been done by obtaining the data in JSON files. However, it could have been done by uploading it to Elasticsearch without any problem.
- number: is the number of responses to return. This parameter is a number with the smallest being 1 and the largest being the maximum number of queries allowed by the API.
- library: The 'library' field indicates the library to be used for the scraper which in this case is the social scraper "snscrape".
- 'before' and 'after' fields: They are used in case we need to obtain data in a specific time frame.
- timeout: It indicates the maximum waiting time for the GSIcrawler response. If none is specified, the timeout must be -1.
- timeout: It indicates the maximum waiting time for the GSIcrawler response.
- Elasticsearch fields: Finally, some parameters related to Elasticsearch are defined, if needed. The Elasticsearch port, the index to upload the documents and the version that is being used are defined through es_endpoint, es_index and es_version respectively.

As mentioned above, there are certain limitations on API requests set by the owners to avoid undesired attacks on their platforms. In the case of StackOverflow, it has a query limit

that has made development a bit more complex. As the number of queries is limited to such a low number that it is not possible to perform an analysis with just one query, the new data obtained had to be checked in this module. By making requests on the same topic twice, there was a possibility that the same thread with several comments could be reanalyzed and distort the reality, having for example ten comments on the same topic indicating the same results of the analysis. In order to carry out this filtering, we have downloaded the data from the index stored in Elasticsearch and through the *'question_id'* field (which stores all the threads obtained) we have been able to determine if it is a new thread. If so, it is saved and if not, it is discarded.

4.1.3 Implementation of Data Processing Module

In this section, we are going to discuss how we have analyzed the information that we have collected from the StackOverflow and Reddit APIs. As we see this module also applies to both data sources, although as we have commented previously, StackOverflow has a slightly superior data processing. In this section, we will discuss the modules of sentiment and LIWC analysis as well as localization. Therefore, this section will be divided into two. Next, we will start by explaining the localization module.

Implementation of Data Processing Module: Localization

To explain this section, we will rely on the code shown below. If we look at the code, we see that it has two sections. The first one checks if the user has the location field and if so, the process continues, otherwise it is terminated. In case it has no location, the fields that are related to the user's location are removed from the JSON document.

Once we determine if the user has the location field in the StackOverflow API, we query the OpenStreetMap API with the value in "locationUser" that the user has included in their profile. This query will return a JSON, which will specify the latitude and longitude value of the user's city. Once here, we can ask ourselves if the users always insert a real location or if they can insert for example a location as "Unknown" or something like "Home". The reality is that users insert what they want, producing the JSONDecodeError on request return in case someone set "Unknown" or "Home" as their own location. That is why all this code is executed under a try-except. The second part of the code is where we get the country and city of the user. Once we verify that the user really has a location, a new OpenStreetMap query is performed to determine the city, town, or village. Finally, these data are saved in the document.

Listing 4.7: location DAG

```
try:
    if 'locationUser' in document:
        locationUser = document["locationUser"]
        responseGeoUserQuestion = requests.get(f"https://nominatim.
            openstreetmap.org/search?q={locationUser}&format=json&limit=1")
            .json()
        if responseGeoUserQuestion and "lat" in responseGeoUserQuestion[0]
            and "lon" in responseGeoUserQuestion[0]:
            lat = float(responseGeoUserQuestion[0]["lat"])
            lon = float(responseGeoUserQuestion[0]["lon"])
            document["locationUser"] = {"lat": lat, "lon": lon}
            es.index(index="geo_point", document={  "location": {    "lat":
                 lat,    "lon": lon  }})
            r = requests.get(f"https://nominatim.openstreetmap.org/reverse?
                lat={lat}&lon={lon}&format=json").json()
            if "country" in r["address"]:
                document["countryUser"] = r["address"]["country"]
            else:
                document.pop("countryUser", None)
            if "city" in r["address"]:
                document["cityUser"] = r["address"]["city"]
            elif "town" in r["address"]:
                document["cityUser"] = r["address"]["town"]
            elif "village" in r["address"]:
                document["cityUser"] = r["address"]["village"]
        else:
            document.pop("locationUser", None)
            document.pop("countryUser", None)
            document.pop("cityUser", None)
except json.decoder.JSONDecodeError:
```

Implementation of Data Processing Module: Sentiment and LIWC Analysis

In this section, we will discuss sentiment analysis and LIWC. For this, we are going to explain previously some concepts of NLP. The basic steps for an NLP analysis are as follows:

- Tokenizacion: The first step is to divide our text into tokens. A token is the minimum unit of characters representing a word. Therefore, a sentence that we want to analyze will be composed of a set of tokens.

- Token reduction: reducing tokens to lowercase is important because we avoid two different analyses for the same word. In other words, you can treat the same word as two different tokens in the same sentence.

- Stopwords In a sentence there are many tokens that we can consider "irrelevant" for our analysis. For example, the articles of a sentence or some determiners or monosyllabic words do not tell us anything, so in order to simplify the number of tokens these words are eliminated.

- Lematización / Stemming. The process of lemmatization or stemming is a process to reduce the number of tokens. Both aim to reduce the tokens that exist in the same sentence and try to group similar tokens in two different ways. The process of lemmatization is to find the lexeme of the word and substitute the word, for example, substituting any verb form for the infinitive verb would be a form of lemmatization. The stemming process substitutes the word for the root of the word. In these two ways, we are able to reduce the number of keywords within a sentence.

On the one hand, the LIWC dictionary has thousands of predefined words and linguistic categories. These categories cover different aspects of language, such as emotions or social relationships. When analyzing a text with LIWC, the analysis counts the number of words belonging to each category and based on these values shows the emotions, the main topic and other aspects found in the text. This analysis, as we can see, takes into account the tokenization process discussed above. The categories that will be shown in this analysis are the following. On the other hand, sentiment analysis involves knowing whether a text is positive, negative or neutral.

Affect words	**Social words**	**Personal concerns**	**Emotion words**
Positive emotion	Female	Work	Anger
Negative emotion	Male	Death	Sad
		Neutral	Positive
		Leisure	Negative
		Home	Anxiety
		Money	

Table 4.1: LIWC categories considered in the analysis

Now that we have explained the objective to be achieved, we return to our pipeline. To be able to apply LIWC and sentiment analysis in our pipeline, the following steps have been carried out. As explained above, sentiment analysis140 and LIWC analysis will be

performed, for which the following calls have been made to Senpy methods that have been adapted for this use case.

Listing 4.8: Senpy calls

```
# Anlyze through sentiment140
resp_sentiment140 = analyze.request_senpy_for_reddit(data["results"][n]["
    cleaned_text"], "sentiment140", SENPY_ENDPOINT, "json-ld", "en")
data["results"][n]["Sentiment"] = resp_sentiment140["entries"][0]["marl:
    hasOpinion"][0]["marl:hasPolarity"]
data["results"][n]["Sentiment"] = data["results"][n]["Sentiment"].replace("
    marl:", "") # Delete tag 'marl:'
# Analyze through LIWC
resp_LIWC = analyze.request_senpy_for_reddit(data["results"][n]["
    cleaned_text"], "LIWC", SENPY_ENDPOINT, "json-ld", "en")
data["results"][n]["entries"] = resp_LIWC["entries"]
```

In this code 4.1, we can notice the calls made to Senpy to perform the analysis. As we can see the Senpy method request_senpy_for_reddit requires five variables.

- The first one is the comment to be analyzed. As we have previously stated, "cleaned_text" is the text without the alphanumeric comments, without the HTML tags or elements that distort the analysis. This filtering has been done thanks to the Python Beautiful Soup library. This parameter is data["results"][n]["cleaned_text"]
- The second parameter required by the method is the algorithm to be used in our case "sentiment140" and "LIWC".
- The third variable that the method required is the Senpy endpoint. This variable was defined in the orchestrator module with the GSIcrawler and Elasticsearch variables. The value of the Senpy endpoint is http://localhost:8007/api/ 4.1
- The fourth variable is the format of the analysis output, in our case it will be in "json-ld" format
- Finally, we indicate the language in which the comments are in, this value can be "en" or "es".

We can also appreciate that in the sentiment140 analysis, the marl: tag is removed from the Sentiment field. This is only for later representation. In fact, the call to perform the LIWC analysis is totally analogous. Once the call is made to Seny, we would get the response which, for sentiment140's analysis, would look something like the following.

Listing 4.9: Senpy Sentiment140 example response


```
{
 "@context": "http://localhost:8007/api/contexts/YXBpP2FsZ289c2VudGltZW50MTQwJmk9V2
     hvJTIwa25ldyUyME5MUCUyMGFuZCUyMHRleHQlMjBwcmVwcm9jZXNzaW5nJTIwY291
     bGQlMjBiZSUyMHNvJTIwZWFzeSUyMHdpdGglMjBweXRob24lM0YlMjAlMjNEYXRhU2NpZW5
     jZSUyMCUyM05MUCZvdXRmb3JtYXQ9anNvbi1sZCZhbGlhc2VzPWZhbHNlJmNvbnZlcnNpb249
     ZnVsbCZleHBhbmRlZC1qc29ubGQ9ZmFsc2UmaW4taGVhZGVycz1mYWxzZSZpbmZvcm1hdD10ZXh0
     JmludHlwZT1kaXJlY3QmdXJpc2NoZW1lPVJGQzUxNDdTdHJpbmcmdmVyYm9zZT1mYWxzZSZ3
     aXRoLXBhcmFtZXRlcnM9ZmFsc2UmbGFuZ3VhZ2U9YXV0byM%3D",
 "@type": "Results",
 "entries": [
  {
   "@id": "prefix:",
   "@type": "Entry",
   "marl:hasOpinion": [
    {
     "@type": "Sentiment",
     "marl:hasPolarity": "marl:Positive",
     "prov:wasGeneratedBy": "prefix:Analysis_1683138670.4678774"
    }
   ],
   "nif:isString": "Who knew NLP and text preprocessing could be so easy with python?
       #DataScience #NLP",
   "onyx:hasEmotionSet": []
  }
 ]
}
```


In this listing 4.9, we can see the sentence "Who knew NLP and text preprocessing could be so easy with python? #DataScience #NLP" has been analyzed. The result of sentiment140's analysis can be seen through the "marl:hasPolarity" field, it has the result "marl:Positive". This is where we see the "marl:" tag that we discussed earlier. The result of the LIWC analysis will not be provided as it would be a similar result. In addition, the listing 4.10 shows a result of a LIWC analysis. Although it is not Senpy's answer, we can get an idea. Once all the data has been analyzed, we move on to the storage module, where the data will be stored in Elasticsearch.

4.1.4 Implementation of Storage Module

In this section, we are going to discuss the data obtained from StackOverflow and Reddit APIs. The structure of the Elasticsearch index is the same for both platforms, only that there are fields that do not apply to both social networks, therefore, these fields have been set as optional to continue with the development. These optional fields has been mandatory for this solution (one index for both platforms), since Elasticsearch requires a value for

each field defined in the index definition. The format chosen to organize the data is JSON (JavaScript Object Notation). This model has been chosen because it is highly compatible with Elasticsearch and provides a quick and easy way to both collect and visualize the data. Each JSON element contains the following elements:

- question_id: As explained above, this field is the unique identifier of the threads obtained from the two platforms. In the case of StackOverflow it will be a set of numbers and in the case of Reddit it will be composed of numbers and letters. Thanks to this identifier we can quickly locate all questions and answers of the thread.

- RedditOrStackOverflow: This field determines the origin of the thread. This field is populated with the field 'Reddit' or 'StackOverflow'.

- locationUser: This field is unique to StackOverflow. As discussed above, Reddit is a forum that appreciates privacy and doesn't even give the user the option to fill out this field. As far as StackOverflow is concerned the location is not added through a Maps API. StackOverflow allows you to add any location you want and does not verify that the destination exists. Therefore, what you get in this field is a string of characters.

- title: The thread title is obtained in this field. It is a field common to Reddit and StackOverflow and is limited to certain characters.

- body: This field contains the comment as it has been written. This field does not contain a limited number of characters. It can also have non-alphanumeric characters, so the 'cleaned_text' field had to be created. This field contains the same content as the 'body' field. Because non-alphanumeric characters (e.g. emojis) have been removed in this field, it is easier and more accurate.

- AnswerOrQuestion: This field determines whether the element is a question or an answer. So by making a request with the question_id, we can determine all the comments in the thread and quickly locate the question.

- tagQuestion: On both Reddit and StackOverflow there is the possibility of assigning a tag to a question so that users can relate it to other similar topics. For the same topic several tags can be applied, for example, if we are dealing with a topic related to Azure DevOps we could apply the 'Terraform' or 'Azure' tags for example. In addition, on Reddit you can also add specific tags for some replies, which StackOverflow does not. There is a difference in the handling of tags by the two platforms. On the one hand, on StackOverflow the tags are predefined and we have to find the right tag for each thread. On the other hand, on Reddit, however, you can create the tags you want.

This has not been a problem for the analysis, since in the end the most common tags are usually used on Reddit. The analysis also showed that StackOverflow is much more likely to use tags than Reddit.

- view_count. This field provides the number of times a given thread has been viewed, making it possible to quickly see the number of times a thread has been viewed.
- score. This field indicates the number of times a user has supported a thread, so this field, like the previous one, also indicates the relevance of the obtained thread. This field and the previous one apply to both platforms.
- link: In this field we store the URL of the thread, both for StackOverflow and Reddit. The StackOverflow URL format is api.stackexchange.com/2.2/questions/question_id?site=stackoverflow and the Reddit URL format is https://www.reddit.com/r/subreddit/comments/question_id/title/.
- is_answered This field is a unique StackOverflow metric. In this social network, the creator of the question or thread is able to give a green check to the answer that solved his problem or simply if he considers the answer to be good. Therefore, this field has also been considered important for the analysis.
- timestamp: Indicates the date on which the comment was created. It is a numeric field since the data in the two APIs is in Unix timestamp format.

e have been the fields that have originally been obtained from the APIs, and then other fields have been obtained due to data analysis such as the following:

- cityUser and countryUser: The OpenStreetMap response has been added to these fields. The city or town of the city and the country have been saved so that later analysis can be made and conclusions can be drawn based on the location.
- Sentiment: The Sentiment field is obtained through the Senpy tool and indicates the result of the analysis. This field is a word that can be: Positive, Negative, or Neutral. This analysis is determined by Senpy's sentiment140 processing.
- word_count: The LIWC analysis needed this field. In order to determine what type of text it is depending on the LIWC categories, the parameter word_count has been added, which allows to know how many words form the text being processed.
- LIWC entries: The LIWC dictionary word processing adds this field. It consists of a JSONs that contains many different fields, such as anger, drives, or informal. Each of these fields indicates how many words are associated with that category. There

are as many fields as there are different categories in a sentence. An example of an analysis would be the following. The input is *"Go to the Azure portal > Azure Active Directory. On the main screen, you should see your tenant ID."* and the result is

Listing 4.10: LIWC analysis example

```
"LIWC:result": {
"adj": 1, # Number of adjectives used in the text.
"affect": 1, # Number of words related to emotions or affections.
"article": 2, # Number of articles (such as "the") used in the text.
"auxverb": 2, # Number of auxiliary verbs used in the text.
"cogproc": 1, # Number of words related to cognitive processing or thinking.
"discrep": 1, # Number of words related to discrepancies or disagreements.
"focuspresent": 2, # Number of words indicating a focus on the present.
"function": 8, # Number of functional words (such as pronouns, prepositions, etc.)
    used in the text.
"home": 1, # Number of words related to the home or dwelling.
"i": 1, # Number of times the pronoun "I" is used in the text.
"motion": 1, # Number of words related to physical movement or action.
"percept": 2, # Number of words related to perception or senses.
"posemo": 1, # Number of words related to positive emotions.
"ppron": 3, # Number of personal pronouns used in the text.
"prep": 2, # Number of prepositions used in the text.
"pronoun": 3, # Total number of pronouns used in the text.
"relativ": 2, # Number of words related to relatives or comparisons.
"see": 2, # Number of times the word "see" is used in the text.
"social": 2, # Number of words related to social interaction.
"space": 1, # Number of words related to physical space or environment.
"verb": 4, # Total number of verbs used in the text.
"you": 2 #  Number of times the pronoun "you" is used in the text.
}
```

- onyx fields: Onyx is a standard used to represent semantic and emotional information in natural language processing. Several aspects are analyzed in this field: "radar_drives", "radar_social", "radar_affective" and "radar_concerns": These fields are representations or summaries of the emotions that have been detected. A numerical value is associated with each emotion, indicating the intensity or importance of that emotion. onyx:hasEmotionSet: This field displays a list of the emotions in the text. Each emotion has a category and intensity onyx:hasEmotionIntensity: Indicates the intensity of the emotion detected through a numerical value, the higher the more importantonyx:hasEmotionCategory: Represents the category of emotion. For example, it can be the previously mentioned LIWC categories "Home" and "Positive". onyx:predominantEmotionSet Shows the predominant emotion in the text. It has four attributes which are "drives", "social", "affective" and "concerns", and they

are assigned LIWC emotions.

All these fields are collected during the execution of the pipeline and are stored in JSON format in the Elasticsarch index *tfm-data*. The following table summarizes the type of variables discussed above. The type is very important since it is transcendental for the upload to Elasticsearch and its subsequent visualization in Kibana.

variable	question_id	RedditOrStackOverflow	locationUser	title	body/cleaned_text
type	keyword	keyword	geo_point	keyword	text/text

Table 4.2: Variables with their types in Elasticsearch. I

variable	AnswerOrQuestion	tagQuestion	view_count	score	link	is_answered
type	keyword	keyword	integer	integer	keyword	boolean

Table 4.3: Variables with their types in Elasticsearch. II

variable	cityUser	Sentiment	word_count	LIWC entries	onyx fields	timestamp
type	keyword	keyword	long	keyword	keyword	date

Table 4.4: Variables with their types in Elasticsearch. III

4.1.5 Implementation of Visualization Module

As mentioned in the previous chapter, Kibana is the visualization tool for monitoring results. For this work, we have developed a dashboard with interactive graphics.

The first thing that has been done in Kibana is to define the index pattern. An index pattern allows Kibana to recognize and access the Elasticsearch indexes that will later be visualized and analyzed. The index pattern allows Kibana to determine what data is viewable and how it should be represented. In addition, Kibana allows to represent multiple index patterns in the same dashboard, which is why the creation of several indexes (one per platform) had been contemplated, but it was later discarded as we have previously stated. This functionality also allows us to classify the indexes according to the date, that is to say, if we have several indexes classified by name and date, for example, Azure-Sentinel-2023-04-25, we can define the index patern as Azure-Sentinel-* and thus we will have all the logs together. In Kibana Dashboards there is the option of filtering the documents through a

specific label for a time, and to be able to use this filter. In the index pattern definition, we have to select which variable controls this filter. The variable "timestamp" has been selected, with date format, as the time axis.

Once the index pattern has been configured and the data is well recognized, we can create the dashboard and configure the visualization components. Multiple diagrams such as pie charts, bar charts or maps can be created. In Kibana several kinds of graphs can be made, but in this project, only two types have been used, so we will focus on these two: aggregation-based diagram and lens diagram. The differences between them can be seen in the following table:

Aggregation-based charts	**Lens charts**
more traditional	provides an intuitive and user friendly visual interface
it is based on aggregation concepts	codeless
requieres a deeper understanding of aggregation concepts	useful for non-technical users
offers more flexibility and control to perform queries	it focus on specific fields

Table 4.5: Aggregation-based charts vs Lens charts

Having explained the differences between them, we will begin to explain the configuration of an aggregation diagram. The concepts to be explained are also suitable for lens diagrams. Next, we will explain the configuration of one of the tag cloud diagrams.

As we can see, there are two parts of the configuration: metrics and buckets.

Metrics determine how large or small the size of each value is to be represented in the diagram. In a cloud tag, the size of the tags is usually associated with a number, such as the number of times a value or certain value is repeated. Some of the most common metrics options for a diagram are:

- Count: This option calculates the number of times a certain value is repeated and represents the value. It is capable of representing the largest or smallest least common value depending on its configuration.

- Average: This option calculates the average of a field if this value were numeric. In

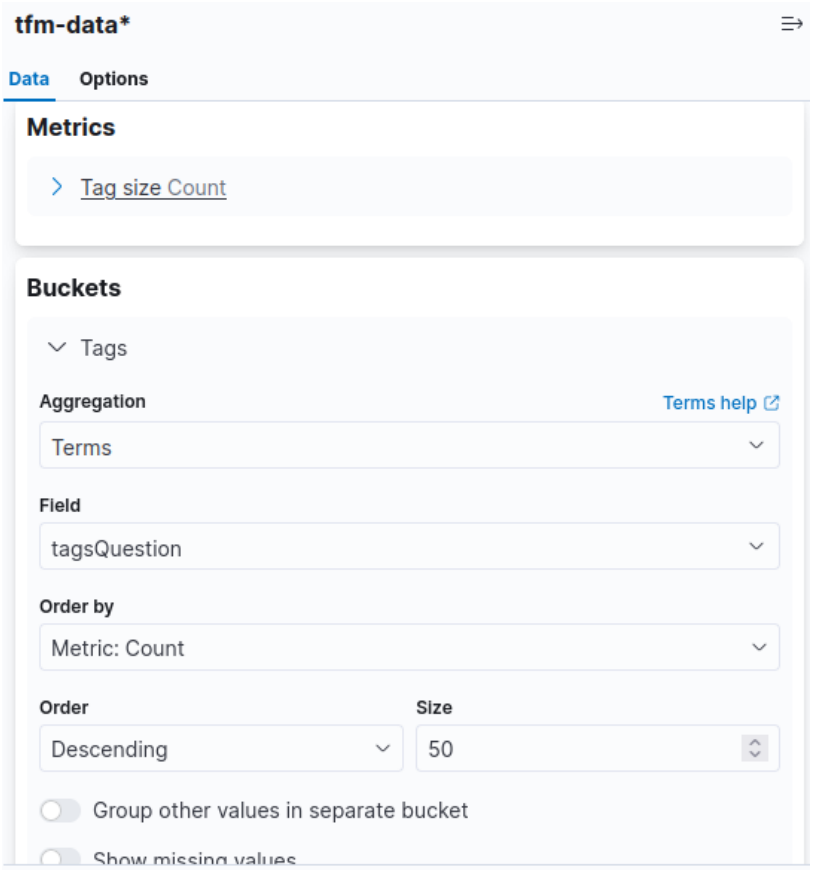


Figure 4.5: Tag Cloud configuration.

this case as we show the country name does not apply.

- Sum: This option calculates the sum of the values that are selected.
- Unique Count: This option calculates the number of unique values and adjusts the size depending on how it is configured.
- Siblign aggregations options: These options allow you to perform calculations and operations on the results of the main aggregation.

Furthermore, we see the Bucket option configured. Buckets are used to group data and select which field is going to be represented in the diagrams. The most common bucket options are Terms and Filters. Terms use the value of a field to represent the graph. In the Tag Cloud, each value becomes a tag represented in the graph. Filters allow you to filter the documents based on a specific selected parameter. Besides, we see on the top of the figure the option 'Options'. In this option, we can configure the front size range in pixels, the orientations, color palette and also the text scale. Thanks to this explanation, we see how configurable are Kibana's diagrams and how well it fits into our system.

4.2 Datasets and results

This section will show the datasets obtained in the execution of this project, as well as their representation in the final dashboard. First of all, we will comment on the conditions of the comments that have been chosen. In addition, it will also show the final results of the analysis for each of the sections of this chapter.

We will now state the conditions we have had to obtain the comments.

- As previously mentioned, the comments are chosen from the Reddit and StackOverflow social networks.
- The comments have to be in English, in order to have a fair comparison in the analysis by localization.
- Duplicate comments will not be taken into account. Before each comment is analyzed, a check is made to see if the comment is in the database. If such a comment is present, it is discarded.
- The date of the comments is not a problem. All types of comments will be analyzed, regardless of when they were created.
- Data has been collected from threads or subreddits of Docker, Azure, Amazon Web Services, Vagrant, Google Cloud Platform, Visual Studio Code, Kubernetes, Python and Terraform.

Therefore, 11,747 comments have been obtained that meet all these requirements. Finally, once the conditions to enter into the analysis have been exposed, the different analyses and results will be explained.

4.2.1 StackOverflow vs Reddit

First of all, we are going to present the proportion of the comments analyzed. As we can see 64.03% of the data that have been analyzed come from Reddit, while the remaining 35.97% are from StackOverflow. In addition, in this same graph, we can see the proportion of comments based on sentiment. We can see that we have a total of 64.73% of neutral comments, 18.64% of positive comments and the remaining 16.63% are negative. These calculations that we have made, we will see that they match with the graph that we will present in the sentiment analysis section.

On the right of the graph we just discussed, a bar chart has also been created in which the variable that distinguishes the origin of the comments, the tag, has also been used. We can see that the proportion is similar in both forums and there is no trend in either.

Finally, we can see the ratio of StackOverflow and Reddit comments as a function of tags, in blue are those belonging to StackOverflow, and in green are those belonging to Reddit.

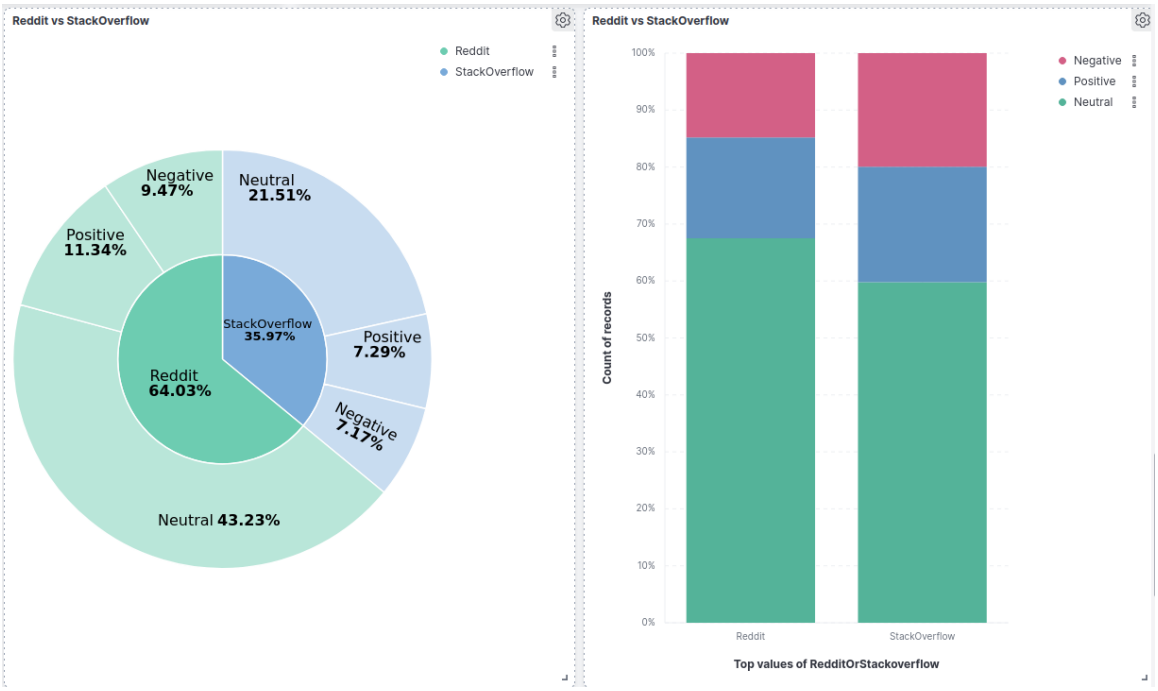


Figure 4.6: Reddit and StackOverflow analysis I.

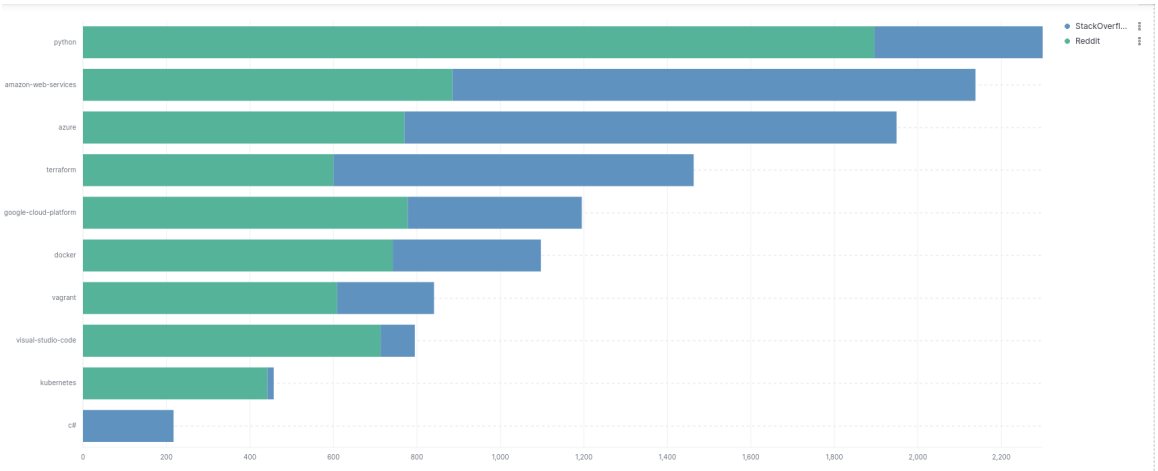


Figure 4.7: Reddit and StackOverflow analysis II.

4.2.2 Tags analysis

The first analysis to be described will be the tags analysis. In the image, we can see that 4 types of graphs have been selected. The first one that we can notice is the first lens graph, which is a diagram in which we see the distribution of comments over time. We can see that

the green line, that belongs to Amazon Web Services, has the most number of comments in the last few months. We can also see that most of the comments have been chosen in the year 2023.

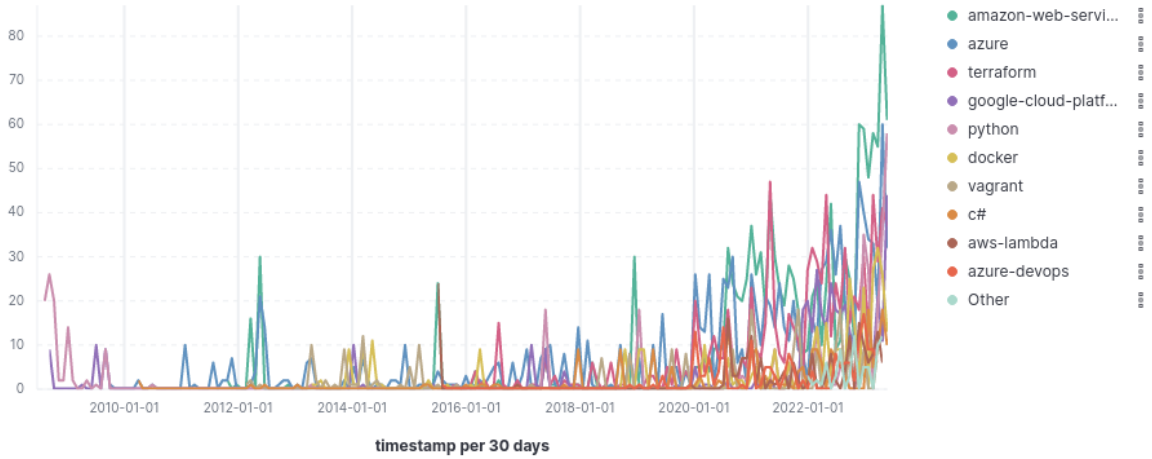


Figure 4.8: Tag graphs.

Next, we see the second diagram, the second lens type, which is a bar chart, in which we see the number of tags. We can see that the threads with the most comments have been Python, Amazon Web Services, and Azure. We also have in red color the negative comments, in blue color the positive ones, and in green color the neutral comments.

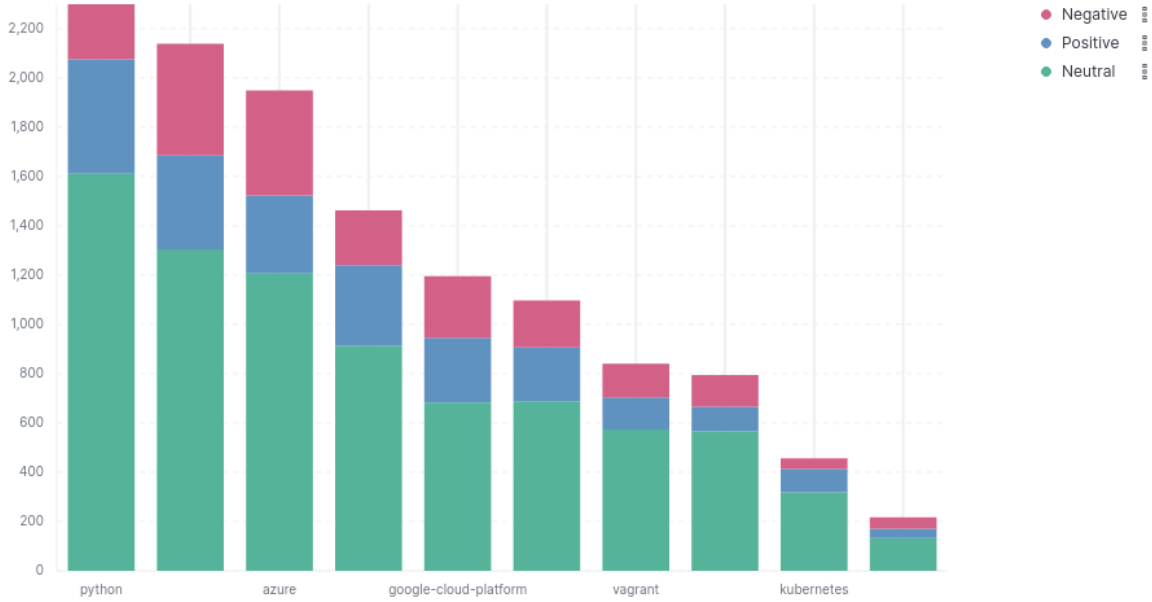


Figure 4.9: Tag graphs I.

The third one is the one we have already mentioned, the tag cloud diagram which is an

aggregation-based diagram. Since the most chosen threads have been the ones previously mentioned, they are the ones that are highlighted the most in this tag cloud. Therefore, as we have explained previously it is shown in a larger font size the more comments there are of that tag.

Figure 4.10: Tag graphs II.

Finally, we see a stacked horizontal bar chart, the fourth lens diagram. This diagram shows the social concern by tags, in which we see that Python has a greater presence in Leisure and Religion and also we can appreciate that Amazon Web Services (in blue color), Azure (in red color), and Google Cloud Platform (in purple color), the three clouder providers,

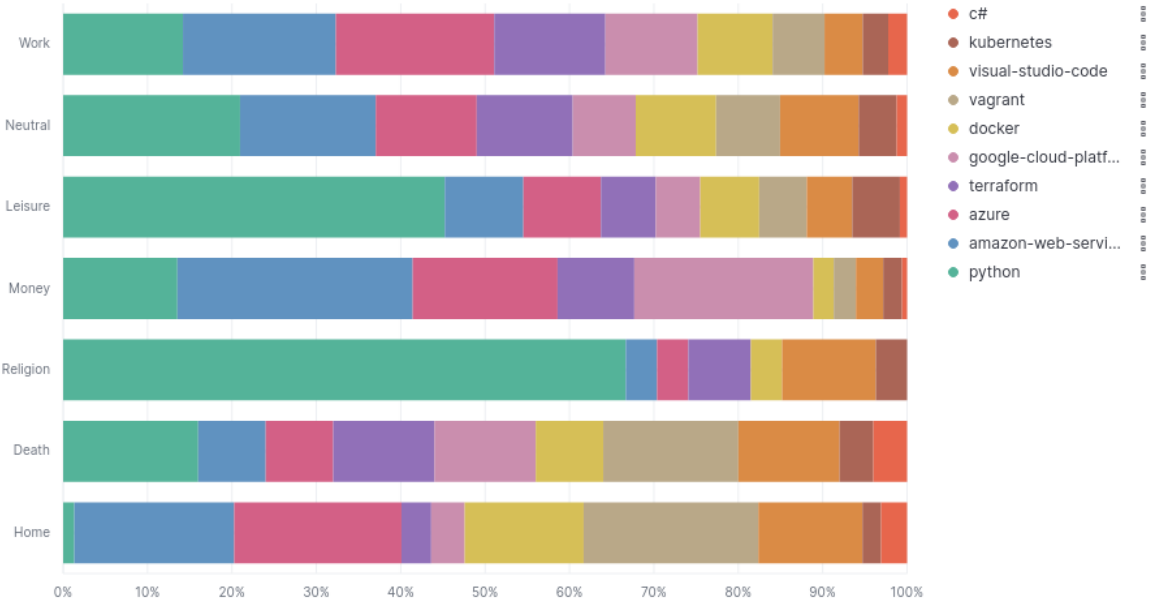


Figure 4.11: Tag graphs III.

4.2.3 Sentiment analysis

We have already seen some diagrams that included sentiment analysis, however here we are going to represent other new figures. Regarding sentiment analysis, it has been represented within a pie chart. They have been represented based on three criteria: Positive (in blue color), Negative (in red color), or Neutral (in green color).

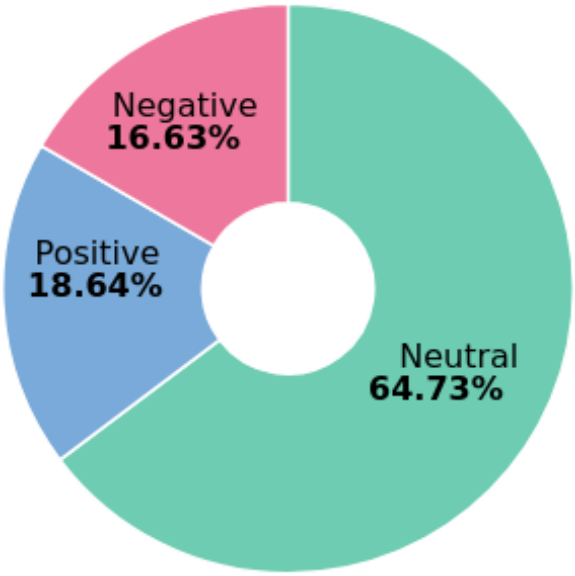


Figure 4.12: Sentiment Analysis Graph I.

This sentiment analysis was carried out by the Senpy tool and the Sentiment140 plugin. In order to obtain the response from Sentiment140, several requests were made to an end-point, which provided us with the result of the comment we inserted. We can see the final result, in which 68.89% of the votes are neutral, 18.64% is positive and 16.48% is negative. The following diagram shows the results per dataset.

The following diagram is a bar graph in which we can see the tags of the comments obtained and the proportion of the sentiment of the comments. The bars in order from left to right are Python, Amazon Web Services, Azure, Terraform, Google Cloud Platform, Docker, Vagrant, Visual Studio Code, Kubernetes, and C#. Although no C# specific thread has been analyzed as it is such a popular programming language, a large amount of C# feedback has been garnered through other threads. This is what explains why it is represented in the diagram. Analyzing the diagram we can see that Amazon Web Services and Azure contain the highest proportion of negative comments.

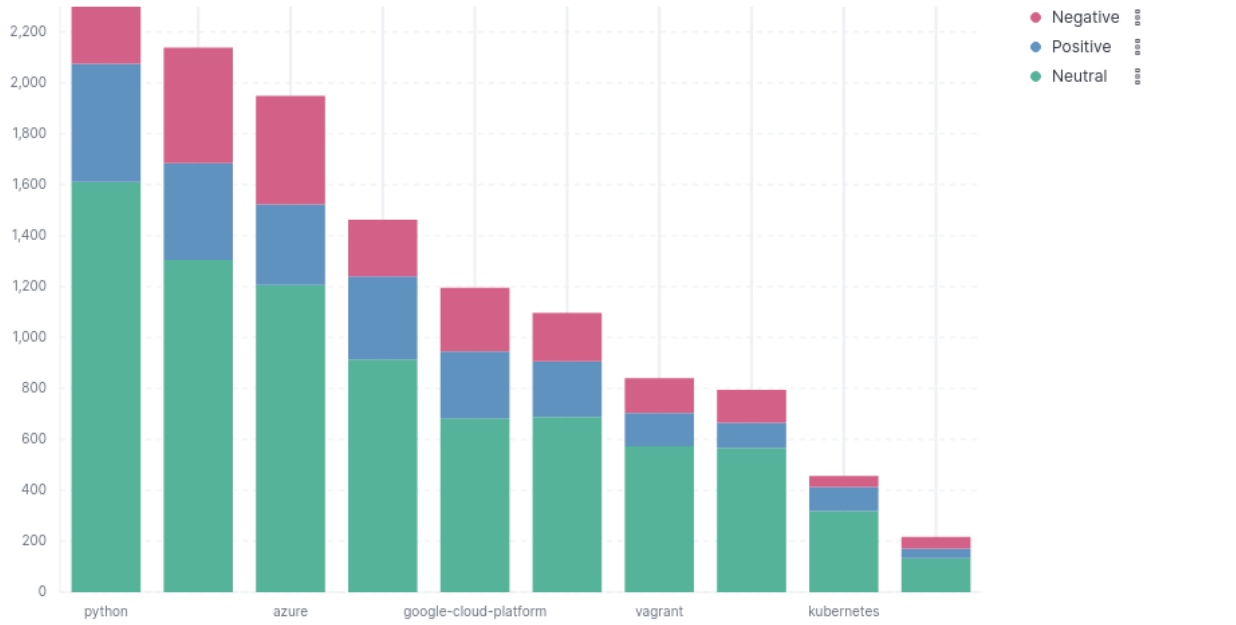


Figure 4.13: Sentiment Analysis Graph II.

4.2.4 Location

The analysis of the locations is composed of seven graphs: two tag clouds, a map, and four bar charts. On the one hand, in the tag clouds, as explained above, we can see the countries with the highest presence among all the countries in the dataset. On the other hand, in the map, we can see the representation of the whole dataset in the same graph.

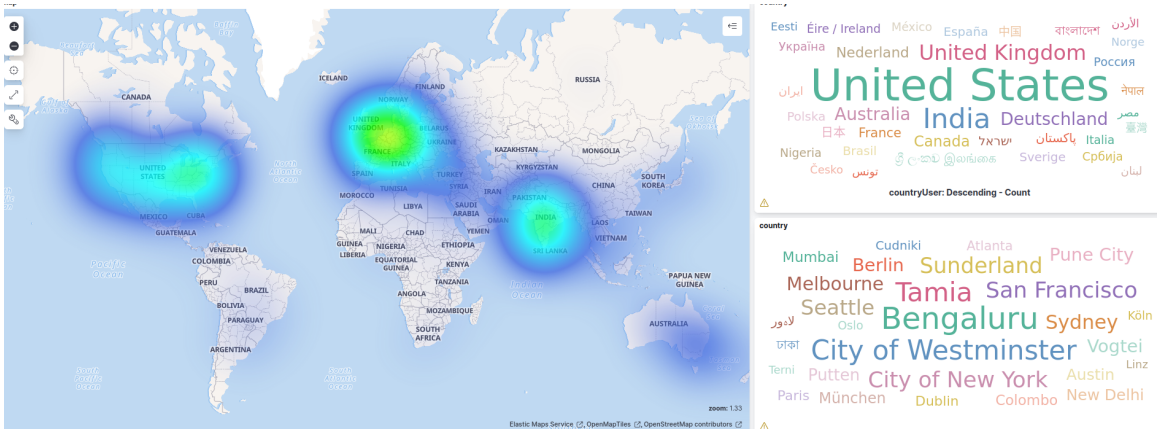


Figure 4.14: Location diagram graph I.

We can notice that the areas with the greatest presence are North America, India, Europe, and a little bit of Australia. This is mainly due to having chosen a dataset in English. If we had chosen the Spanish datasets we could be sure that the influence of South

America would have been greater. The top contributors to these datasets were the United States, the United Kingdom, and India. This result fits with the map shown above in which we discussed the origin of Reddit users 1.2. It also fits with the analysis done by similar-web with StackOverflow's origin [48].

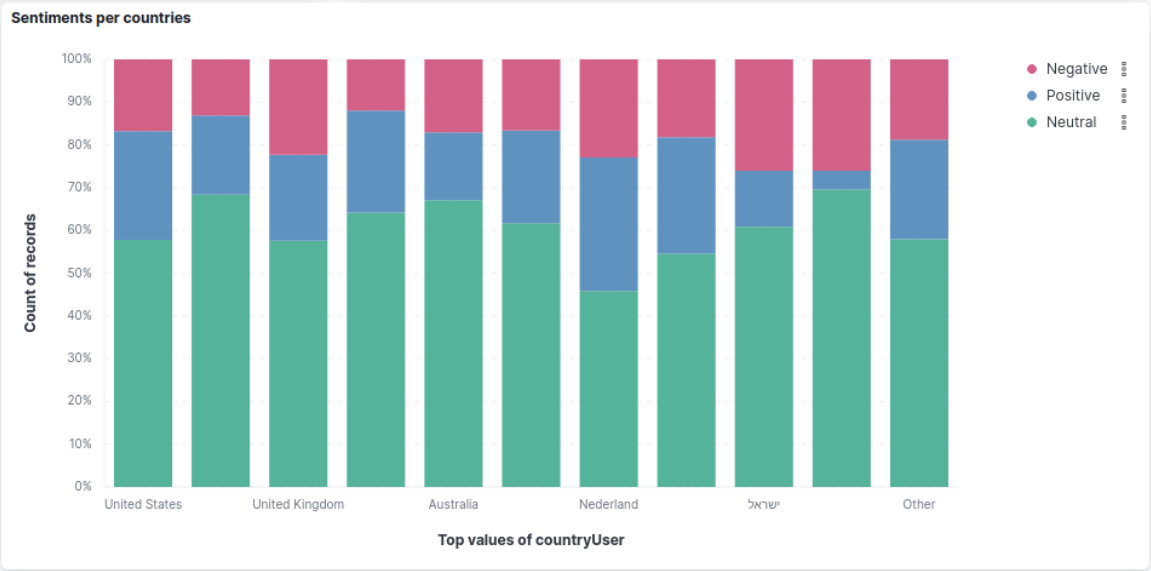


Figure 4.15: Location diagram graph II.

As mentioned above, the latitude and longitude coordinates of OpenStreetMap were used to create the map diagram. In order to realize the tag cloud diagram, OpenStreetMap has also been used to obtain the city and country of the location that the user has entered in StackOverflow. Moreover, a bar stacked chart has been created so as to visualize the sentiment analysis per country in which we can see that most of the comments are neutral (in green color) and also that in Sri Lanka we see a lot of positive impacts. In addition, a bar chart has also been added so as to count the origin of the country comments, through it we can notice that the United States is the country with the most comment analyzed.

The next diagram is the relation between the tags and the country. Thanks to this diagram we can see the impact of a specific technology per country. We can notice that all the Kubernetes comments that have been scraped and analyzed come from Germany. These last four diagrams are configured to show the 10 countries with the most comments, although they can be configured for as many as we wish.

Finally, the last graph is the relationship between social concerns and the countries. We can see in this diagram that there is a homogeneous distribution of interests by country. The predominant social concerns will be explained in the upcoming section.

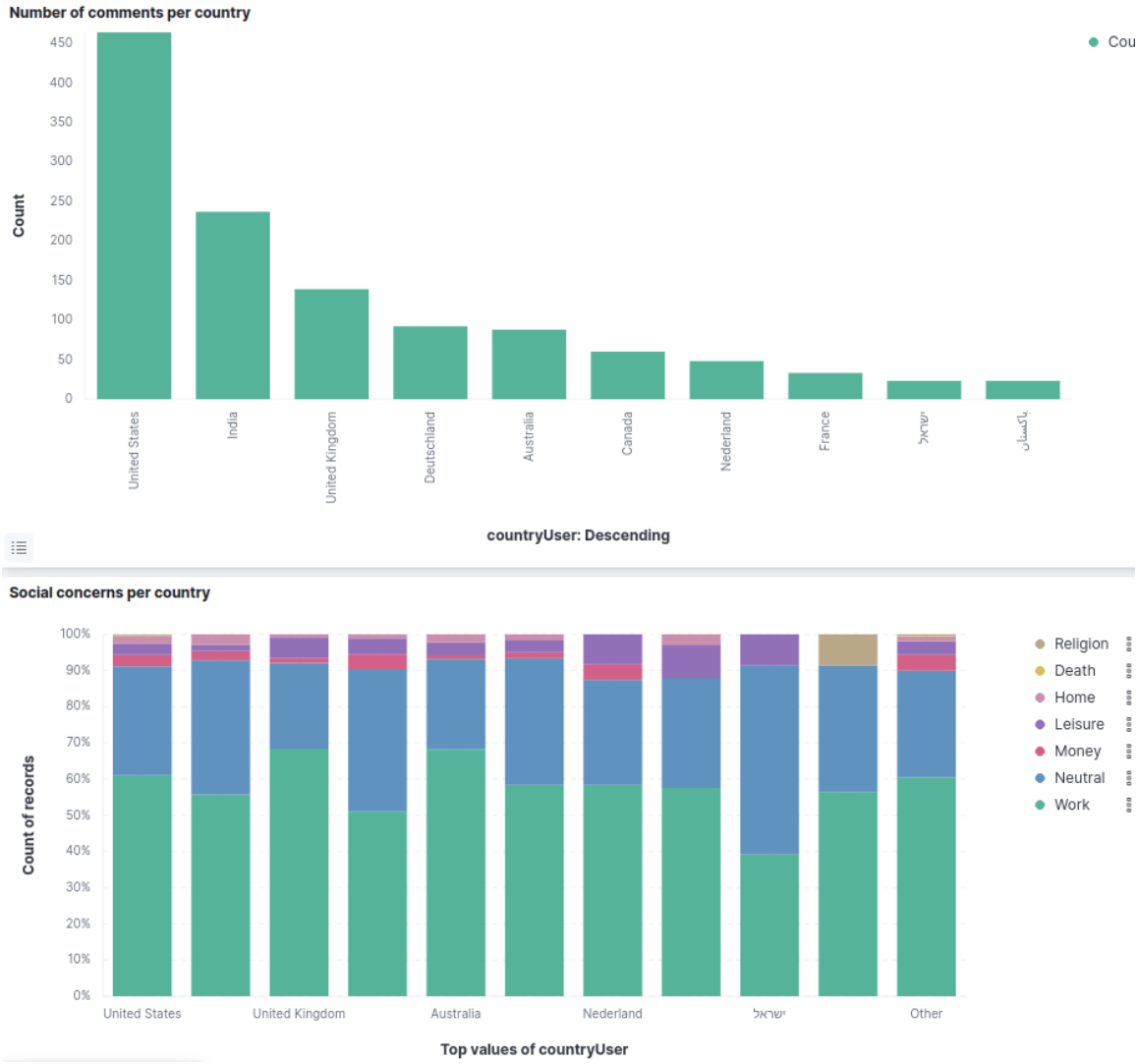


Figure 4.16: Location diagram graph III.

4.2.5 LIWC Analysis: predominant social concerns, affect words, social words, and emotion analysis

In this last section we will deal with the last four analyses, although as in previous cases, we have already shown some graphs in which we could notice some results of these analyses. We will start by explaining the predominant social concerns analysis. In order to explain the results of these LIWC analyses, we have based ourselves on the article [54]. As we have explained previously, this analysis is based on the words. This algorithm is capable of classifying words based on categorization, and as a result, it returns the amounts of each category.

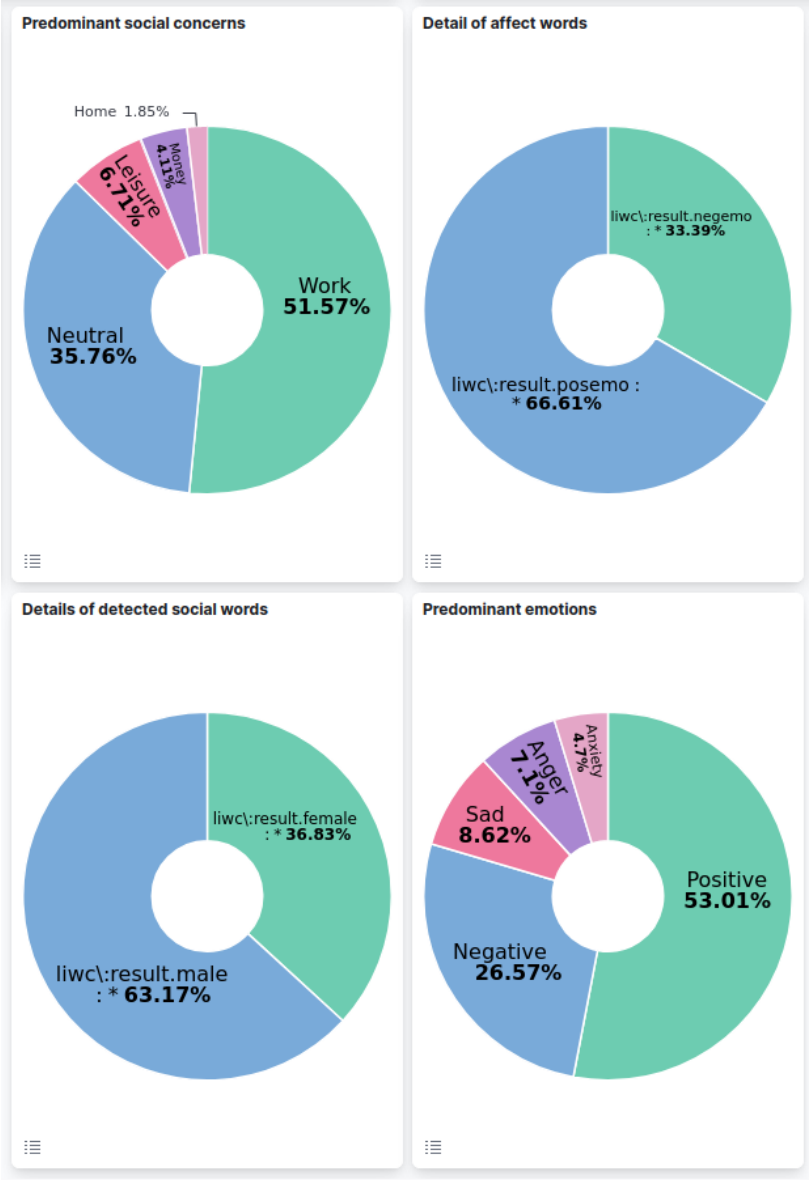


Figure 4.17: Predominant social concerns graph.

Predominant social concerns

The LIWC analysis has produced a very interesting result, which is that of predominant social concerns. This analysis allows us to categorize the comment according to the context in which it is written, with 5 different categories: Work, Neutral, Leisure, Money, Home, Religion, and Death.

We have already presented some figures in which this analysis has appeared. We see this analysis in the relationship between countries and social concerns 4.9 and also in the previous section in the 4.14 figure. Thus, now we are going to present the result of the whole

analysis in a pie chart (first graph). The results are 51.57% related to work content, 35.76% neutral, 6.71% related to leisure content, 4.11% related to ecnonmic content (money), and 1.85% related to home content. We can notice that most of the comments refer to people's work. This lets us know that it is a place that has much more than a simple gamers' forum. We also have a significant percentage of the neutral category. In rather smaller proportions we have a few comments belonging to Leisure, Home, and Money.

Detail of affect words

Affect words are words that have a positive or negative impact. This does not indicate the sentiment of the comment. It simply counts the positive or negative words in each comment. We can see in the result that there are 66.61% of positive words while there are 33.39% of negative words.

Detail of detected social words

This analysis has also allowed us to obtain the LIWC analysis. This graph allows us to know whether the message is going to a man or a woman. Social words are words that refer to other people for example they, she, he, we, talk, or friends. Therefore, in this analysis, we will only present the proportion of comments referring to men and women. In the below-left graph, we can see that the proportion is 63.17% for men (in blue color) and 36.83% for women (in green color).

Detail of emotion words

The last of the LIWC analyses conducted indicate the influence of people's emotions on their comments. In the diagram below right we can see the result in various colors. We can see that five emotions have been mainly represented. In green color we have the positive with 53.01%, the negative represented with blue color with 26.57%, the sad in red with 8.62%, the angry in purple color with 7.1%, and the anxiety with 4.7 in light pink color.

4.2.6 Final dashboard

In this last section of this chapter, we are going to show the set of graphs configured in the final dashboard. In this figure, we can see all the comments that have been explained throughout this section.

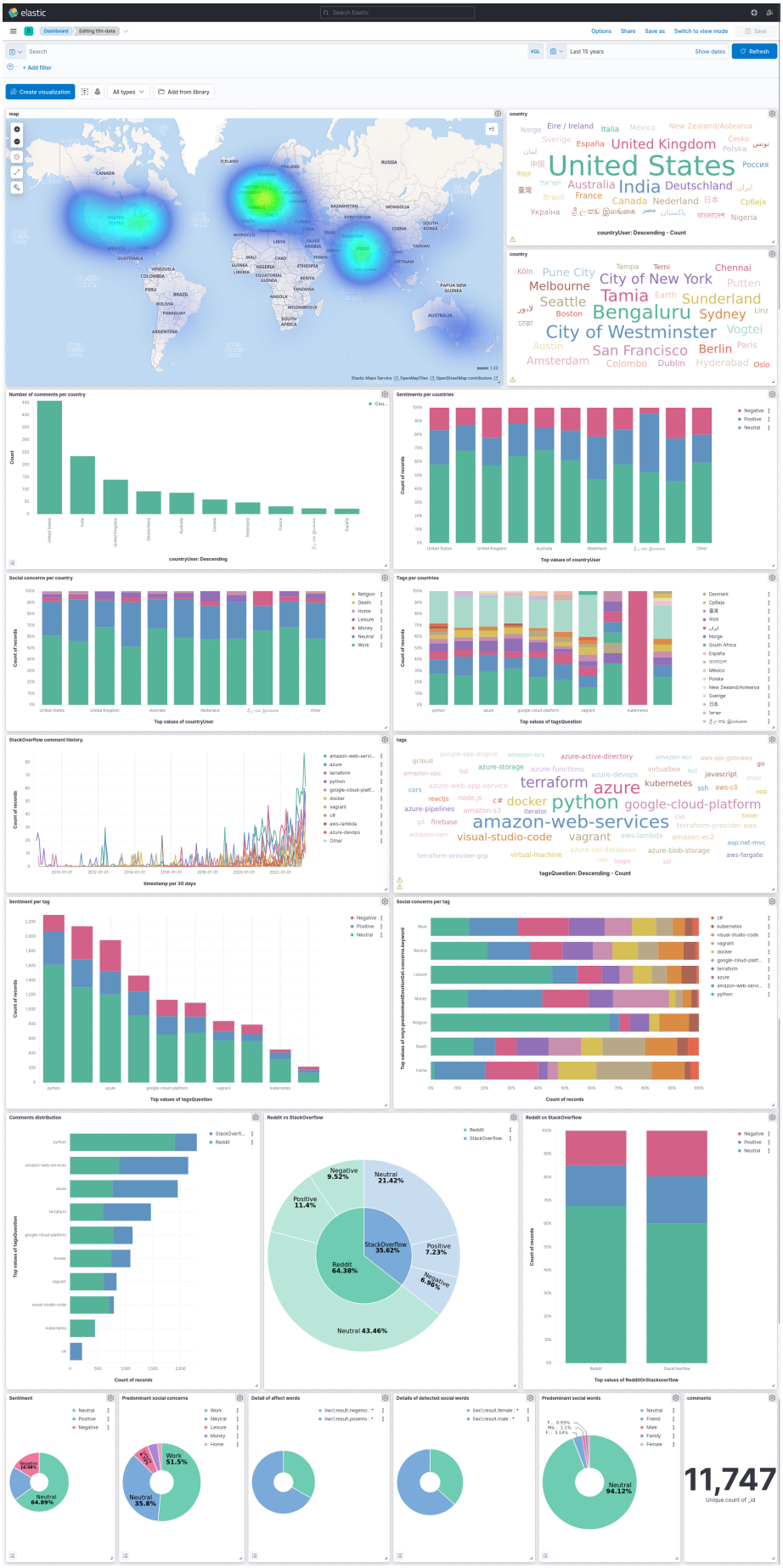


Figure 4.18: Final dashboard.

CHAPTER 5

Conclusions

In this chapter, we are going to focus on the aspects that have been achieved in relation to what was discussed in the first chapter. There will be a review of the work done. In addition, future lines of work will be discussed. These improvements should certainly complete and improve the system. Finally, we will present the conclusions that have been carried out in the execution of the project.

5.1 Achieved Goals

In this project, we have developed a system capable of obtaining, analyzing, and visualizing user comments on DevOps (Amazon Web Services, Visual Studio Code, Vagrant, Google Cloud Platform...) technologies topics on Reddit and StackOverflow. The objectives established at the beginning of this project have been met through the execution of this thesis were the following.

1. Develop and implement in a scraper system a way to get the information This objective has been achieved because two systems have been integrated into GSIcrawler to obtain information through the APIs of two platforms that are world leaders in their area.

2. Add this information into a database. This objective has also been met because the Elasticsarch search manager has been used as the database. A single index has been created in Elasticsearch to store the data needed.

3. Natural Language Processing techniques are used to analyze language. In addition, NLP techniques have been used for the analysis of these data through the Senpy tool, thus obtaining different results. Several analyses have been performed: sentiment analysis and LIWC analysis. Therefore, with what has been argued on this point, this objective could also be considered met.

4. Create a dashboard so as to visualize the data analyzed. This objective has been achieved because the Dashboard has been presented with all the details previously mentioned. In addition, combined graphs have been included to make it easier to read the results and draw conclusions.

5. Analyze the results obtained through the analysis. In the last chapter, the results have been presented and analyzed, so this objective has also been met.

In addition, an orchestrator, Airflow, has been configured for the execution of these tasks. This has enabled the management and control of the pipeline through the web interface and Airflow DAGs.

Therefore, it can be said that all the objectives that had been proposed at the beginning of this project have been met. However, during the execution of this same development, new lines of development have arisen in which progress can be made, as we will see in the following section.

5.2 Future lines

In this section, we will discuss the future directions of this work. To begin with, the future lines can be developed in new functionalities, new integrations, new platforms, and new technologies. Therefore we are going to comment on these points in the following bullet points.

- Integration with more platforms. Although this paper has monitored two of the largest Internet forums in terms of technological development, there are many more Internet forums that can be monitored. Examples of these forums are Quora, Code Project, Github Discussions, Question2Answer. It should be noted that these platforms must have an open API. Of the cases that I have exposed, Quora does not have an API to which requests can be made, but Github does. Therefore, Quora would not be a valid example for the integration of this platform into our system despite being a forum that has great similarities to those studied in this work.

- Visualizations and process improvements. As mentioned above, Kibana offers a large number of services that have not been configured, for example, notification systems to detect inconsistencies or a certain range of values. This allows you to receive real-time alerts when certain events or metrics exceed set limits. From an optimization point of view, certain aspects of the pipeline could be improved to make it more efficient and more intuitive. A parallel Dashboard could be created for the integrations without having to do it through the console.

- Cloud development. For a fully delegated execution in the cloud, this system could be integrated into a cloud such as Azure. Integrating it into Azure as a PaaS system has multiple advantages. To begin with, if we want to use this system on a large scale and have high availability and disaster recovery plans, without a doubt the cloud is the most appropriate option to set up the system due to the architecture of public and private clouds. In addition, the cloud provides great scalability plans and costs based on usage. To finish this point, the migration to, for example, Azure would benefit from the Power BI environment for the dashboard.

- Get more feedback for deeper analysis. The fundamental aspect of this project is the data, and obviously, one line of work for the future would be to include more comments in order to carry out an even better analysis. In addition, we could not only look at comments in English, but we could also create sentiment analysis algorithms for different languages and include them in our analysis.

5.3 Conclusion

This project has carried out the monitoring of comments on social networks. At the beginning of the project, I was not aware of the large number of comments that can be monitored through open-source tools. From the development point of view, this work is the first step on the road to the development of a monitoring system. As we discussed in the previous section, there are several areas where this project can be substantially improved from a business point of view.

The importance of this project is significantly high, as we have commented in the first chapter of this thesis. On the one hand, there are several cases in which companies use this type of system in order to have almost instantaneous feedback on their products. This allows cost savings in the dissemination of user surveys. In addition, real and objective feedback can be obtained from social networks. On the other hand, you can also study market trends, and analyze and look for explanations as to why a product is improving or worsening in popularity. Moreover, as we have shown, they can be obtained and analyzed by geographical areas. With this location analysis, many results and explanations of trends and usage of a platform in one region or another can be obtained.

Another issue to be considered is the privacy of forum users. This is an important aspect that has been observed in the execution of the project, as we have seen that in the APIs we can obtain a large amount of user data simply by making a request. Through requests to the APIs of these networks, a tracking system can be set up to track the desired users. Therefore, when building a system like this, if user data is needed, anonymization of these data should be considered.

To sum up, we can see that this type of tool has a really powerful use and that you can get a lot out of it thanks to the large amount of data that stands behind a simple comment.

APPENDIX A

Impact of this project

A.1 Introduction

This annex will take into account privacy, economic, environmental, and ethical impacts.

A.2 Social impact

Sentiment analysis of comments has an important social impact. This thesis describes the sentiment of people commenting on aspects of social networks (Reddit and StackOverflow and analyzes their status). Therefore, in the case of anyone thinking about which technology to use then, in this case, he could benefit from this project.

On the other hand, the exercise of using a dashboard and representing the whole analysis should inspire anyone who wants to analyze the impact of any technology.

A.3 Environmental impact

Regarding the environmental impacts, the project is developed on a local computer, it is not carried out on a large scale and it is not replicated on different servers. Therefore, the use of computers and computer screens should be taken into account. We should especially focus on the computer battery as the most polluting aspect of the tools used.

In summary, it can be stated that this thesis does not have an excessive environmental impact.

A.4 Economic impact

Regarding the economic issue, the following appendix provides more detailed information on the cost of the project. As far as the advantages or disadvantages of this project are concerned, there is no doubt that the use of monitoring techniques saves companies a lot of time and money. In fact, many large companies already have systems similar to these in their infrastructure.

A.5 Ethical Implications

The ethical implications go hand in hand with user privacy in the execution of the project. This aspect has been taken into account in the project description, in fact, it has been commented in several examples of API requests that all user-related data has been deleted. Moreover, in the project execution, these fields have not been taken into account for any analysis. Therefore, this project as executed would respect the privacy of the users. However, the development of these tools may involve monitoring user accounts and tracking their activity in case user identifiers were saved, so this aspect would have to be taken into account.

APPENDIX B

Economic budget

In this appendix, we will discuss the economic budget. We will consider that the work has 700 hours since it corresponds to the 30 ECTS. In addition, certain direct costs have been considered, such as the computer and the monitor. Indirect costs have not been considered, since they have not been necessary. Therefore, the total cost is 7.020,95 €. More information can be found in the next table.

Table B.1: Economic Budget

LABOR COST (direct cost)		**Hours**	**€/hour**	**Total**
		750	15 €	**4.500 €**
COST OF MATERIAL RESOURCES (direct cost)	**Purchase price**	**Usage in months**	**Amortization (in years)**	**Total**
Personal laptop (software included)	1.000,00 €	12	5	200,00 €
Monitor	300,00 €	12	5	60,00 €
TOTAL COST OF MATERIAL RESOURCES				**260,00 €**
GENERAL EXPENSES (indirect costs)	15%	DC		**714,00 €**
INDUSTRIAL PROFIT	6%	DC +IC		**328,44 €**
CONSUMABLES				0, 00 €
SUBTOTAL BUDGET				**5.802,44 €**
VAT			21%	**1.218,51 €**
TOTAL BUDGET				**7.020,95 €**

APPENDIX C

System deployment and implementation process

This annex will detail the technical procedure for visualizing and executing the monitoring system.

1. First of all, what you have to do is to raise the containers through *docker compose up*. This will cause the applications that make up the system to start working. Once we have waited the necessary time, we can check through the following command the status of the containers, obtaining the following result C.1.

2. The next check to perform is to identify the tag from which you want to get the comments. In order to do this, two lines have been displayed at the end of the pipeline file, which has to be configured with the desired parameter. It has been configured in two variables because the same subject can be configured in two different forms. For example, the Google Cloud Platform on StackOverflow is GCP and on Reddit is googlecloud.

Listing C.1: Tag Variables

```
tag_stack = "azure"
tag_reddit = "azure"
```

```
ualBox:~/Descargas/airflow-data-pipelines-main$ docker ps
  IMAGE                                                COMMAND                  CREATED      STATUS
 NAMES
  apache/airflow:2.3.1-python3.8                       "/usr/bin/dumb-init …"   6 days ago   Up 6 days (healthy)
 airflow-data-pipelines-main-airflow-webserver-1
  apache/airflow:2.3.1-python3.8                       "/usr/bin/dumb-init …"   6 days ago   Up 6 days (healthy)
 airflow-data-pipelines-main-airflow-scheduler-1
  apache/airflow:2.3.1-python3.8                       "/usr/bin/dumb-init …"   6 days ago   Up 6 days (healthy)
 airflow-data-pipelines-main-airflow-triggerer-1
  apache/airflow:2.3.1-python3.8                       "/usr/bin/dumb-init …"   6 days ago   Up 6 days (healthy)
 airflow-data-pipelines-main-airflow-worker-1
  gsiupm/gsicrawler:dev                                "/usr/src/app/init.s…"   6 days ago   Up 6 days
 airflow-data-pipelines-main-gsicrawler-1
  docker.elastic.co/elasticsearch/elasticsearch:7.17.0 "/bin/tini -- /usr/l…"   6 days ago   Up 6 days
 airflow-data-pipelines-main-elasticsearch-1
  postgres:13                                          "docker-entrypoint.s…"   6 days ago   Up 6 days (healthy)
 airflow-data-pipelines-main-postgres-1
  gsiupm/senpy                                         "python -m senpy -f …"   6 days ago   Up 6 days
 airflow-data-pipelines-main-senpy-1
  docker.elastic.co/kibana/kibana:7.17.0               "/bin/tini -- /usr/l…"   6 days ago   Up 6 days
 airflow-data-pipelines-main-kibana-1
  redis                                                "docker-entrypoint.s…"   6 days ago   Up 6 days
 airflow-data-pipelines-main-rediscrawler-1
  redis:latest                                         "docker-entrypoint.s…"   6 days ago   Up 6 days (healthy)
```

Figure C.1: Docker state

For example, showing it on StackOverflow we would see the following example.

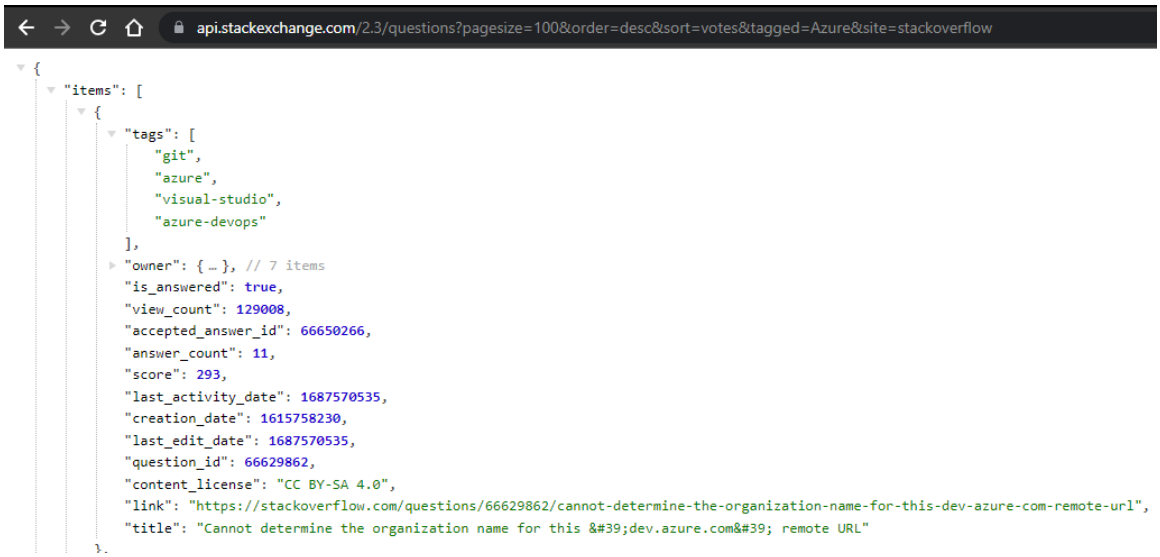


Figure C.2: Query StackOverflow

3. Once the tag to be scraped is configured, the pipeline must be executed in the Airflow control panel. Through the following page http://localhost:8080/home. Within the DAG, we have multiple options. For example, a very interesting option to display is the duration of the modules. The following image shows the duration of each of the tasks during the execution of the project. It should be noted that this DAG was not the only one executed during the project; in fact, it was the last of the three to be created. This was due to the fact that initially one DAG was created for each technology and then they were unified into a single DAG.

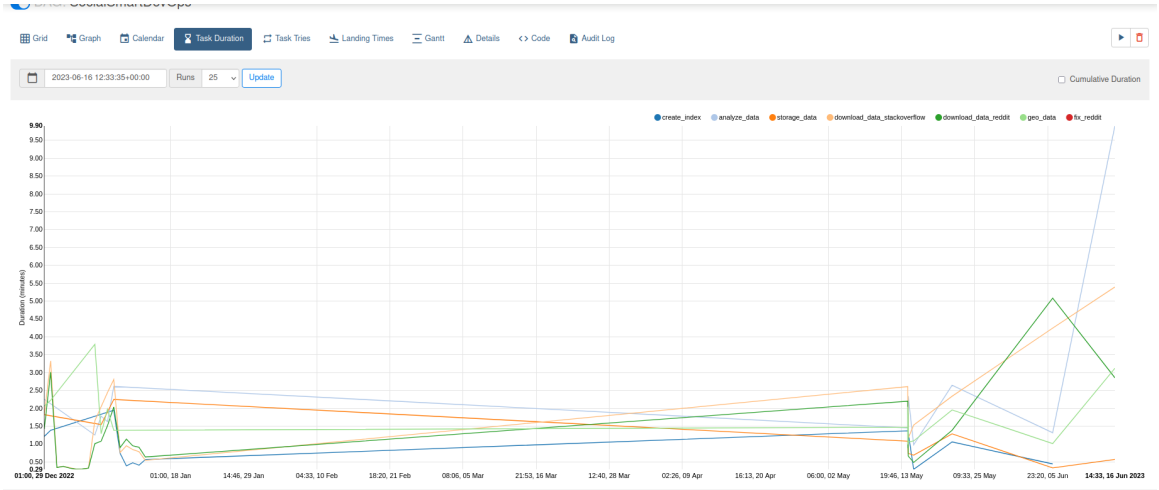


Figure C.3: Task Duration

4. Once all the states have been executed correctly as shown in figure 4.3, the pipeline will be finished and we will have obtained all the comments.

5. One check we can make to see if our application has been processed correctly is to consult the database. To do this we have to perform the following query http://localhost:9200/tfm-data/_search?q=tagQuestion:azure. In this query, you can check that we are selecting the comments that contain the Azure tag, which has been the example shown in step 1. For example, for this use case, you would expect a response like the following figure C.4:

6. The last step, in order to analyze the results in the Dashboard we can check them at the following URL http://localhost:5601/app/dashboards#/view/c3c64650-049a-11ee-8b34-55dd26004ccb?_g=(filters:!(),refreshInterval:(pause:!t,value:0),time:(from: now-15y,to:now)). At this URL you will get the Dashboard shown above 4.18.

   In addition, as mentioned above, the diagram is interactive and can be filtered by the result you want to obtain. For example, in the following example, I have filtered the comments obtained from Spain between 01/01/2021 and 12/31/2023 C.5.

Finally, all this development is available in the following public GitLab link: https://lab.gsi.upm.es/agonzalez/tfm

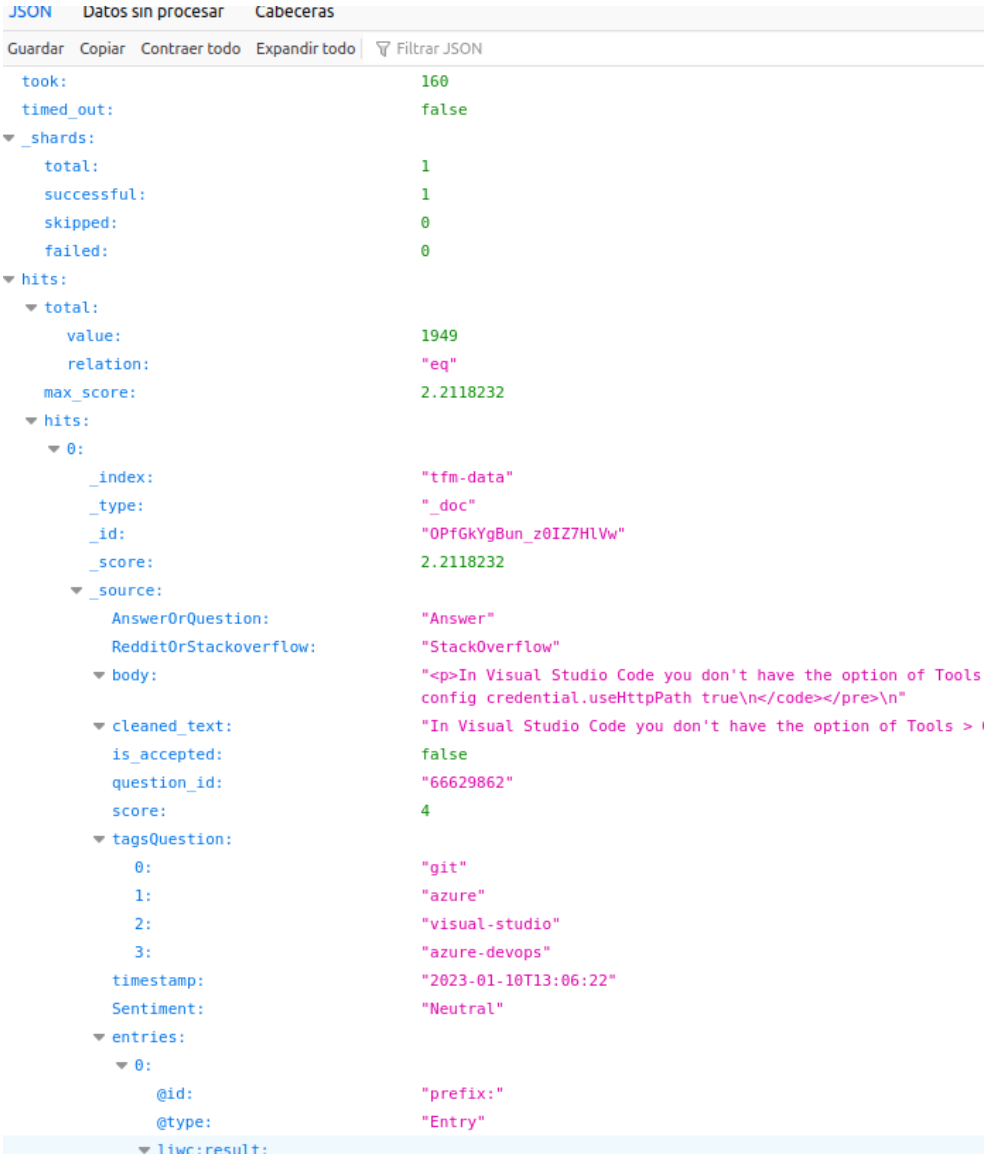


Figure C.4: Elasticsearch database

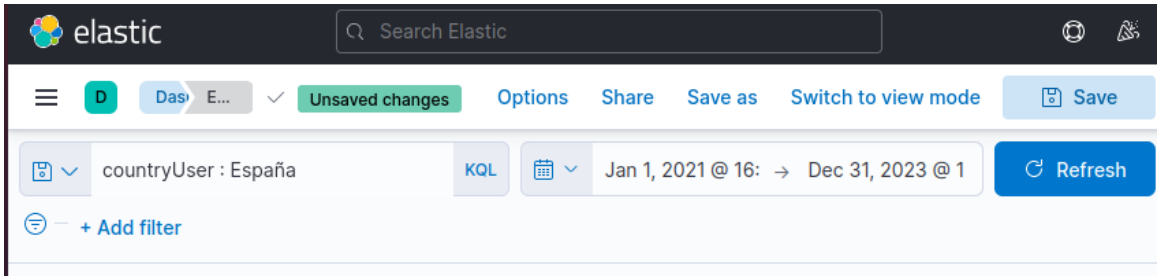


Figure C.5: Kibana filter

Bibliography

[1] Apache airflow — Wikipedia, the free encyclopedia. Accessed: February 13, 2023.

[2] Nlp2rdf: Rdf/owl-based linguistic linked data. https://persistence.uni-leipzig.org/nlp2rdf/. Accessed on: 09-06-2023.

[3] Reddit: publicidad online, 10-06-2023.

[4] K21 Academy. Docker vs virtual machine: What's the difference? https://k21academy.com/docker-kubernetes/docker-vs-virtual-machine/, 30-05-2023.

[5] Irfan Ahmad. Twitter, facebook, google plus, instagram, pinterest - the real time internet [infographic]. https://www.digitalinformationworld.com/2014/05/Twitter-Facebook-GooglePlus-Instagram-Pinterest-Real-Time-Internet-infographic.html, mayo 2014.

[6] Airflow. Architecture overview. https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/overview.html. Accessed on June 20, 2023.

[7] Apache Airflow. Airflow cli and environment variables reference. https://airflow.apache.org/docs/apache-airflow/stable/cli-and-env-variables-ref.html, 2021. Accessed: 2022-05-04.

[8] Javier Alonso. Así era eliza, el primer bot conversacional de la historia. Xataka, June 2019.

[9] Amazon Web Services. What is natural language processing? https://aws.amazon.com/es/what-is/nlp/, 2021. [Accedido el 23 de marzo de 2023].

[10] Jennings Anderson. Openstreetmap diary entry, 10-06-2023.

[11] Apiumhub. Usar elasticsearch: Ventajas y libros recomendados, 2020.

[12] AprendeMachineLearning. Procesamiento del lenguaje natural (nlp), 2021.

[13] Tucker Arrants. Using google translate for nlp augmentation. https://www.kaggle.com/code/tuckerarrants/using-google-translate-for-nlp-augmentation, 2021.

[14] Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. The pushshift reddit dataset. In Proceedings of the Fourteenth International AAAI Conference on Web and Social Media (ICWSM 2020), 2020.

[15] Margaret M Bradley and Peter J Lang. Affect and the functional organization of the brain. neurological research, 21(3):203–206, 1999.

[16] Abram Brown. Así es reddit: una compañía con 16 años más conocida ahora que nunca antes en su historia, 10-06-2023.

[17] Miriam Martínez Canelo. ¿qué es celery? https://profile.es/blog/que-es-celery/, 30-05-2023.

[18] Planeta Chatbot. Proyectos de análisis de sentimientos en github [nlp], 2022.

[19] Kate Conger. Uber's massive scraping program collected data about competitors worldwide, 12 2017.

[20] Ignacio Corcuera-Platas. Development of a Deep Learning Based Sentiment Analysis and Evaluation Service. Master thesis, ETSI Telecomunicación, Madrid, January 2018.

[21] Crawlbase. How does google scrape websites? https://crawlbase.com/blog/how-google-scrape-websites/, 2019. [Accessed: 22 de mayo de 2023].

[22] Creately. Tutorial de diagrama de despliegue. https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-despliegue/#:~:text=Un%20diagrama%20de%20despliegue%20es,el%20middleware%20que%20los%20conecta, 30-05-2023.

[23] Datademia. ¿qué es apache airflow? - datademia. https://datademia.es/blog/que-es-apache-airflow#%C2%BFQue_es_un_DAG, 2023. [Online; accessed 13-Febrero-2023].

[24] Domo. Data never sleeps. Website, Accessed: 2023.

[25] Elastic. What is Elasticsearch? https://www.elastic.co/es/what-is/elasticsearch, 2021. [Online; accessed 23-March-2023].

[26] Elastic. What is elk stack? https://www.elastic.co/es/what-is/elk-stack, 2021. [Online; accessed 23-March-2023].

[27] Elastic. What is elastic stack security?, 2022.

[28] Elastic. ¿qué es kibana?, 2023.

[29] Elastic. Kibana dashboards, Current.

[30] FinancesOnline. Benefits and advantages of social media monitoring. https://financesonline.com/benefits-and-advantages-of-social-media-monitoring/, Fecha de acceso: 2023-04-21.

[31] Hostinger. ¿qué es docker? guía completa para principiantes. https://www.hostinger.es/tutoriales/que-es-docker, 30-05-2023.

[32] Carlos A. Iglesias. Ciencia de datos y aprendizaje automático en la web de datos. Universidad Politécnica de Madrid, 2022.

[33] Ksenia Kazanova. Sentiment140 dataset with 1.6 million tweets. https://www.kaggle.com/kazanova/sentiment140, 2019.

[34] KeepCoding. Scraping vs crawling: ¿cuál es la diferencia? https://keepcoding.io/blog/scraping-vs-crawling-cual-es-diferencia/. Accedido el 20 de junio de 2023.

[35] MappingGIS. Openstreetmap, la plataforma de mapas libre más grande del mundo. MappingGIS, April 2021.

[36] Microsoft. Concepto de airflow administrado en azure data factory. https://learn.microsoft.com/es-es/azure/data-factory/concept-managed-airflow, 9 2021. Accessed: May 4, 2023.

[37] Paolo Missier, Khalid Belhajjame, and James Cheney. The w3c prov family of speci[U+FB01]cations for modelling provenance metadata. 03 2013.

[38] Pascal Neis and Dennis Zielstra. Recent developments and future trends in volunteered geographic information research: The case of openstreetmap. Future Internet, 6:76–106, 03 2014.

[39] OpenStreetMap. Openstreetmap. Página web, 2023.

[40] Stack Overflow. Prosus acquires stack overflow, junio 2021. Accessed: 11-06-2023.

[41] Data Science PE. Clasificación de textos utilizando aprendizaje profundo. Data Science PE, 2021.

[42] James W. Pennebaker, Ryan L. Boyd, Kathleen Jordan, and Kate Blackburn. The liwc2015 (linguistic inquiry and word count) dictionary. https://lit.eecs.umich.edu/wordcount/, 2015. [Online; accessed May-03-2023].

[43] Red Hat. What is virtualization. https://www.redhat.com/es/topics/virtualization/what-is-virtualization. [Fecha de acceso: 20-06-2023].

[44] RPP. Reddit ahora está valorado en más de us $10 mil millones. https://rpp.pe/tecnologia/mas-tecnologia/reddit-ahora-esta-valorado-en-mas-de-us-10-mil-millones-noticia-1352295, 2021. [Accessed: el 22 de mayo de 2023].

[45] Muhibul Sabur. How to identify spam using natural language processing (nlp). Towards Data Science, 2021.

[46] Severalnines. What is elasticsearch and why use it? https://severalnines.com/blog/what-is-elasticsearch-and-why-use-it/, September 2021. Accessed on February 13, 2023.

[47] Similarweb. reddit.com traffic and engagement, 10-06-2023.

[48] Similarweb. stackoverflow.com traffic and engagement, 10-06-2023.

[49] snoowrap contributors. snoowrap: A JavaScript wrapper for the Reddit API. https://github.com/not-an-aardvark/snoowrap, 2021. Accessed: 2023-05-06.

[50] Sonar Platform. Why is social media monitoring important for your brand business? https://sonarplatform.com/why-is-social-media-monitoring-important-for-your-brand-business/, 2023. Accessed: Feb. 13, 2023.

[51] David Taylor. Data lake architecture - what it is and why it matters, 10-06-2023.

[52] Velotio Technologies. Web scraping: Introduction, best practices, caveats. Velotio Perspectives, 2019.

[53] UNIR. Nlp: procesamiento de lenguaje natural. Revista UNIR, 2020.

[54] University of Texas at Austin, Department of Psychology. Liwc tat results. http://www.utpsyc.org/TAT/LIWCTATresults.php, 19 June 2023. Accessed: el 22 de mayo de 2023.

[55] Wikipedia. Docker (software). https://es.wikipedia.org/wiki/Docker_(software). Accessed: Fecha de acceso.

[56] Wikipedia. RDF Schema. https://es.wikipedia.org/wiki/RDF_Schema. Accessed: 22-05-2023.

[57] Wikipedia. Virtualización a nivel de sistema operativo. https://es.wikipedia.org/wiki/Virtualizaci%C3%B3n_a_nivel_de_sistema_operativo. [Fecha de acceso: 20-06-2023].

[58] Digital Information World. Real-time internet: Twitter, facebook, google+, instagram, pinterest [infographic]. https://www.digitalinformationworld.com/2014/05/Twitter-Facebook-GooglePlus-Instagram-Pinterest-Real-Time-Internet-infographic.html, 2014. [Accessed: 22 de mayo de 2023].

[59] World Wide Web Consortium. W3C PROV Diagrams. https://www.w3.org/2011/prov/wiki/Diagrams, 2011.

[60] Xataka. Tencent compra stack overflow por 1.800 millones de dólares: la web de referencia para programadores, 60 2021. Accessed: 11-06-2023.