uc3m | Universidad Carlos III de Madrid

University Degree in Telematics Engineering

2022-2023

Bachelor Thesis

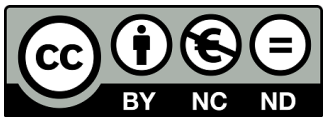
Design and Development of a Scalable Web Scraper for Large-Scale Data Collection

---

David Rico Menéndez

Angel Cuevas Rumin

Madrid, 1st of January 2023



SUMMARY

Data gathering is a fundamental technique widely used in various domains to collect and analyze information for research, decision-making, and insights generation. As the digital landscape continues to expand exponentially, the amount of available data has grown exponentially as well. However, this growth presents significant challenges. The sheer volume of data, coupled with the increasing size of samples and datasets, has made traditional manual data gathering methods cumbersome and time-consuming. Furthermore, the rise of automated blocking mechanisms employed by online sources further complicates the data gathering process, leading to extended waiting times and reduced accessibility to valuable information. These challenges highlight the need for scalable and efficient solutions, such as the automated system developed in this thesis, to address the complexities of data gathering in a rapidly evolving digital environment.

This thesis presents the design, development and test of a web scrapper tool for data extraction and analysis. The objective of the project was to create a scalable and efficient solution to gather large amounts of data from online sources while considering the limitations and challenges associated with the process. The system utilizes web scraping techniques and leverages multiple account tokens to optimize data retrieval rates and mitigate the risk of account blocking.

Overall, this thesis contributes to the field of automated data extraction and analysis by providing a scalable and efficient solution that addresses the challenges of large-scale data gathering. The findings and insights presented in this thesis can serve as a foundation for future research and development in similar domains.

Keywords: Web Scrapping, Scrapper, Crawler, Data gathering, Big data

DEDICATORIA

A todas las personas que me han acompañado en este trabajo, especialmente a mi tutor (por partida doble), Ángel Cuevas, cuyo saber crítico me ha motivado a mejorar constantemente.

A mi familia más cercana. Espero poder algún día devolver cada abrazo, cada sonrisa y cada empujón hacia delante. Os quiero mucho.

A mis abuelos, Ana y Fernando, por estar siempre dispuestos a recibirme con inmenso cariño y compartir su sabiduría, y a Nieves y Arsenio por tenerme siempre presente, cada día, a pesar de la distancia.

A Mercedes, este es solo uno más de los innumerables agradecimientos que te debo por todo lo que haces por mí. Iluminas mi vida.

A Rogelio, por ser mi brújula, siempre preocupado por guiarme por el mejor camino y ayudarme a crecer a todos los niveles.

A Carlos y Toñi, por su cariño y calidez, siempre dispuestos a ayudarme y brindarme un buen consejo cuando ha sido necesario.

Al PEAC y a la Dirección General de Universidades e Investigación, quienes sentaron las bases y aportaron los recursos necesarios para llevar a cabo los dos grados que finalizo hoy.

A la UC3M, por darme la oportunidad y confiar en mí, cuando no existía precedente, y permitirme simultanear los estudios con los que siempre soñé.

A todos y cada uno de ellos, les doy mi más sincero agradecimiento.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The Internet is a powerful tool that shapes our lives in many ways. It enables us to access information, communicate with others, and enjoy various forms of entertainment. However, every time we use the Internet, we also leave traces of our behavior and preferences. These traces can be collected and analyzed by various entities, such as advertisers, marketers, researchers, and governments. This data, at the same time, is used to create new products that can serve us, or improve our user experience while navigating. We have, unconsciously, become active participants in shaping the online landscape, utilizing the vast array of tools and services available to us.

Some of these entities use our behavior data to target us with personalized ads or offers, which can influence our decisions and actions. Others use our behavior data to understand how we think and feel, and what motivates us to behave in certain ways. This can provide valuable insights for various fields of study, such as sociology, psychology, economics, and education.

In this thesis, we explore the potential of using Internet behavior data for socio-economic investigations, and for that we create a tool that allow us to retrieve and study this bast amount of information.

We define Internet behavior data as any data that reflects how individuals or groups interact with the Internet or with devices connected to the Internet. This includes data from social networks, online platforms, Internet of Things devices, sensors, cameras, biometric devices, and more. We will use Facebook Marketing API as the source of this.

We argue that Internet behavior data can reveal important aspects of human behavior that are not easily observable or measurable by other means. For example, Internet behavior data can capture how people react to different events or stimuli, how they express their opinions or emotions, how they form social networks or communities, how they learn or acquire new skills, how they spend their time or money, and finally and the main focus in this use case, what (Facebook thinks) are we interested in.

1.1. Motivation and objective

The motivation behind this project stems from the research interests of my supervisor and the identified need for accessing scalable aggregated data from Facebook's ADs Manager. There was a requirement to download massive volumes of data to support various research activities focused on addressing socio-economic issues.

This new source of information presented an opportunity to explore and expand the measurement of culture, leveraging the ability to gather substantial data for future social studies.

One notable example of the studies conducted using this data source is the research titled "Expanding the measurement of culture with a sample of two billion humans"[1]. This study aimed to examine the feasibility of utilizing the vast amounts of data available from Facebook's Ads Manager to make meaningful contributions to future social studies. The research encompassed 60,000 topics across more than 225 countries, serving as an initial exploration of the potential impact of this data source.

Additionally, other studies have utilized the same data source to investigate topics such as the micro-targeting in the 2016 US election. This study harnessed the power of this data to gain insights into these sociopolitical phenomena, contributing to a deeper understanding of their implications.[2]

Furthermore, it is worth highlighting that in 2021, over 600,000 rows of raw data were required and obtained to conduct a social study on the differences in interests between countries in Europe.[3] A similar study in terms of dataset size was performed in the same year, this time regarding the gender gap in terms of interests and preferences online.[4] This exemplifies the substantial volume of data needed for comprehensive research in the socio-economic domain, and the significance of accessing data from Facebook's Ads Manager efficiently.

All this led to the desire to assist researchers and developers in enhancing their investigations by providing them with efficient access to vast amounts of data. Traditional methods of data collection for research purposes often face challenges such as limited availability, time-consuming processes, and restricted access imposed by certain entities. These limitations hinder the progress of socio-economic studies that could ultimately improve our daily lives.

The objective of this thesis is to address the difficulties faced by researchers and developers by developing an automated system that overcomes the barriers to accessing valuable data. By successfully creating a robust and scalable method for data extraction, this project aims to facilitate socio-economic investigations and contribute to the advancement of knowledge.

To illustrate this, let's consider a hypothetical study that aims to investigate three age groups (young, adult, and old) while differentiating between male and female individuals across a set of 4,000 interests within the ten most relevant countries in Europe. In order to obtain comprehensive results, we would need to gather data from approximately 250,000 rows, including baseline information. As the scope of the study expands, the dataset size could potentially grow into the millions.

Manual data gathering for such extensive datasets would be nearly impossible and extremely time-consuming, potentially causing significant and unacceptable delays in the experiment or research project.

Another approach to tackle this problem could involve conducting interviews with a smaller sample size and then extrapolating the results. However, relying solely on extrapolation would introduce a level of inaccuracy and reduce the reliability of the findings.

By automating the data collection process using a web scraper, we can efficiently gather these extensive datasets within a reasonable timeframe, with minimal effort required from the researchers. The development of this technology provides researchers with the ability to access and analyze large banks of data, enabling comprehensive socio-economic investigations and contributing to the advancement of knowledge in various fields.

Moreover, on a personal level, the motivation behind this thesis lies in overcoming the numerous obstacles posed by large companies that restrict access to certain data. By enabling access to such data for legitimate purposes, such as social studies, this project strives to empower individuals and promote positive impacts on our daily lives.

We have selected Facebook as the subject of our study because it is a leading platform in the field of social media and online behavior. Facebook has several features and advantages that make it an ideal case for our investigation. Some of these are:

- Facebook is one of the largest and most popular social networks in the world, with almost three billion monthly active users as of January 2023. This means that Facebook has a huge and diverse user base that covers different regions, cultures, demographics, and interests. Facebook also has a high level of user engagement, with an average of 1.93 billion daily active users in Q4 2022. [5]

- Facebook has expanded its reach and influence by acquiring other platforms such as Instagram and Snapchat in 2012 and 2014 respectively. These platforms have also grown significantly in terms of users and revenue, and have added new features and functionalities that complement Facebook's core services. For example, Instagram has become a dominant platform for visual content and influencer marketing, while Snapchat has introduced innovative formats such as Stories and Filters that appeal to younger audiences. By integrating these platforms into its Family of Apps, Facebook has increased its market share and competitive edge in the social media landscape. [6]

- Facebook has rebranded itself as Meta in 2021, signaling its ambition to create a metaverse, which is a virtual environment where people can interact with each other and with digital content in immersive and realistic ways. Meta's vision is to build a platform that connects the physical and digital worlds, and that enables new forms of social interaction, entertainment, education, commerce, and creativity. Meta's metaverse will leverage technologies such as virtual reality, augmented reality, artificial intelligence, blockchain, and cloud computing. The metaverse product will serve as a new feed of data for their marketing API we are focusing on.[7]

Facebook Marketing API allows users to access estimated values of potential customers based on their interests, and to filter them by various criteria that can be useful for research purposes. For example, users can query the Marketing API to get the number of people who are interested in a certain topic or product category, and who match certain attributes such as age, gender, location, education level, income level, etc. Users can also use the Marketing API to create and manage advertising campaigns on Facebook and its Family of Apps. The Marketing API provides a rich source of data and insights about online behavior and preferences that can be leveraged for socio-economic investigations.

1.2. Structure of the study

To provide a clear structure for this thesis, a summary of each chapter is presented below:

- Introduction: This chapter provides an overview of the project and aims to familiarize the reader with its objectives and significance.
- State of the Art: In this chapter, we examine the current state of web scraping tools and web crawling and discuss the various applications and significance of these technologies.
- Large-Scale Data Collection: This chapter serves as an introduction to web scraping tools, providing an overview of the basic concepts and features needed to understand the topic.
- Development of the Scrapper: Here, we provide a detailed description of the tool developed, including an in-depth analysis of the various functions.
  - Single account approach: In this section, we will discuss the first version developed that based all the queries on a single account and their limitations found.
  - Multi account approach: This section will focus on the second version developed, as well as the introductions of all the new features related to the inclusion of multiple accounts and their rotation and management.
- Results and Tests: In this chapter, we present a discussion of the results and utility of the developed tool, including a detailed analysis of the various tests conducted to evaluate the tool's effectiveness.
- Future Work: This chapter briefly covers potential improvements to the present and future technologies that could be used to enhance the developed tool's performance.
- Bibliography: This chapter contains a list of all sources used to conduct the study, including references to academic journals, books, and online sources.

2. STATE OF THE ART

2.1. Current state of web scrapping and web crawlers

Web scraping and web crawling are widely used techniques for extracting data from websites. These techniques have gained significant importance in various domains, including research, business intelligence, data analysis, and more. In this section, we will explore the current state of web scraping and web crawling, highlighting their applications and significance.

Web scraping refers to the automated extraction of data from websites, allowing users to gather structured information for various purposes. With web scraping, researchers can collect large volumes of data from multiple sources, which can be used for data analysis, trend identification, and informed decision-making. Web scraping has proven to be a valuable tool in fields such as market research, sentiment analysis, price comparison, and content aggregation.

Web crawling, on the other hand, involves systematically navigating through websites and gathering data by following links and exploring different pages. Web crawlers, also known as web robots or spiders, traverse the internet to index web pages for search engines or collect data for various applications. They enable the retrieval of vast amounts of data from multiple websites in an automated and efficient manner.

In recent years, advancements in web scraping and web crawling technologies have led to the development of more sophisticated tools and frameworks. These tools offer enhanced capabilities, such as handling dynamic web content, dealing with authentication and session management, and handling large-scale data extraction.

However, it is essential to acknowledge that web scraping and web crawling practices can potentially infringe upon users' privacy, particularly when personal data is collected without consent. To address this concern, one of the main pillars of this project is to prioritize and protect user privacy. This is achieved by exclusively gathering aggregated data through the tool.

Aggregated data refers to information that has been combined, summarized, or grouped together to form broader insights and patterns. It entails the anonymization and consolidation of individual data points to ensure the privacy of users. Rather than focusing on specific individuals or their personal details, aggregated data provides a high-level view of trends, behaviors, and statistical patterns within a dataset.

By utilizing aggregated data in social investigations, researchers can derive meaningful insights while maintaining user privacy. The data obtained through the tool is stripped of any personally identifiable information, preventing the identification or tracking of individuals. Instead, it allows for the analysis of collective behaviors and societal trends.

The use of aggregated data not only safeguards user privacy, but also enables researchers to uncover valuable insights that can contribute to socio-economic investigations. By examining patterns and trends at a broader level, researchers can gain a comprehensive understanding of various phenomena without compromising the privacy rights of individuals. This approach ensures that the project adheres to ethical guidelines and legal requirements while still providing valuable data for social research.

Overall, the current state of web scraping and web crawling presents both opportunities and challenges. Researchers and businesses can leverage these techniques to access and analyze valuable data, but they must also protect user's privacy and adapt to evolving measures employed by websites to protect their content.

By understanding the current state of web scraping and web crawling, we can better appreciate the context in which our project operates and the significance of the advancements and contributions it brings to this field.

2.2. Overview of existing and commercial web scrappers tools and their limitations

Web scraping tools are software applications or libraries that can automate the process of extracting data from web pages. Web scraping tools can vary in terms of functionality, complexity, usability, and cost. Some web scraping tools are designed for specific purposes or domains, while others are more general and flexible. Some web scraping tools require coding skills or technical knowledge, while others are more user-friendly and intuitive. Some web scraping tools are free and open-source, while others are commercial and proprietary.

Some examples of existing and commercial web scraping tools are:

- Scrapy: Scrapy is a popular and powerful open-source framework for web crawling and scraping, written in Python. Scrapy allows users to create custom spiders that can crawl and extract data from any website, using various features such as selectors, pipelines, middlewares, extensions, etc. Scrapy also provides a command-line interface and a web service for controlling and monitoring the spiders.[8]

- Octoparse: Octoparse is a cloud-based web scraping tool that enables users to scrape data from any website without coding. Octoparse provides a visual interface that allows users to configure the scraping tasks by pointing and clicking on the web elements. Octoparse also supports advanced features such as dynamic loading, pagination, login, captcha, etc. Octoparse offers both free and paid plans with different limits and features.[9]

- ParseHub: ParseHub is another cloud-based web scraping tool that allows users to scrape data from any website using a simple point-and-click interface. ParseHub can handle complex websites with JavaScript, AJAX, cookies, etc. ParseHub also

provides a desktop app and an API for accessing the scraped data. ParseHub has a free plan with some limitations and a paid plan with more features and support.[10]

- Apify: Apify is a web scraping platform that provides various tools and services for web crawling and scraping. Apify allows users to create custom web scrapers using JavaScript or TypeScript, or use ready-made scrapers for popular websites such as Amazon, Google, Facebook, etc. Apify also provides a cloud infrastructure for running and scaling the scrapers, as well as an API and a marketplace for accessing and sharing the scraped data.[11]

However, web scraping tools also have some limitations that need to be considered before using them. Some of these limitations are:

- Anti-scraping mechanisms: Some websites may deploy various anti-scraping mechanisms or countermeasures to prevent or deter web scrapers from accessing their data. These include captcha, IP blocking, user-agent detection, dynamic content loading, encryption, obfuscation, etc. Web scrapers may need to use advanced techniques or tools to overcome these challenges, such as optical character recognition (OCR), IP rotation, headless browsers, etc.
- Web structure and format changes: Web pages may change their structure or format over time due to updates, redesigns, maintenance, etc. This may affect the accuracy and validity of the scraped data or cause errors or failures in the scraping process. Web scrapers may need to update or modify their code or configuration to adapt to these changes, or use robust selectors or parsers that can handle variations in the web elements.
- Data quality and validity: Web scraping may result in incomplete or inaccurate data due to various factors such as missing values, duplicates, outliers, errors, inconsistencies, etc. Web scrapers may need to perform data cleaning and validation steps to ensure the quality and validity of the scraped data or use reliable sources or methods for verifying the data.

2.3. Related work in the field

In recent years, there have been several notable research studies and projects focused on web scraping, data extraction, and automated data collection. These works have contributed to the development of techniques, methodologies, and tools that enhance data accessibility and enable researchers to gather valuable insights. In this section, we will discuss some of the relevant studies conducted in this field.

Web scraping and web crawling have gained significant attention from the research community in recent years. A comprehensive study titled "Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application"[12] provides a valuable

overview of the advancements and benefits associated with these techniques. It not only showcases the current state of the art but also offers insights into the future paradigm in this field.

Python has emerged as a popular choice for web scraping, and one notable reference book in this domain is "Web Scraping with Python: Collecting More Data from the Modern Web" by Ryan Mitchell (2019)[13]. This book provides a comprehensive guide, ranging from fundamental concepts to tackling complex scenarios, making it an invaluable resource for researchers.

When focusing on Facebook as a data source for research purposes, several recent studies have explored the extraction of data from this platform using tools like Selenium. These studies have employed classification techniques, such as Support Vector Machines (SVM), to identify cyberbullying scenarios[14] or classify aggregated users sets based on the Big Five Personality traits[15]. However, these studies primarily emphasized flexibility rather than performance, and the data extraction processes were semi-automated, lacking high-speed and large-scale data gathering capabilities.

In a related work by Manscosu et al., the advantages of utilizing Facebook data as a research source are explored. The study highlights how the use of web scraping techniques, particularly with aggregated data, enables researchers to conduct investigations while ensuring the preservation of user privacy. By focusing on aggregated data, which does not contain personally identifiable information, researchers can extract meaningful insights without compromising user privacy.[16]

These works demonstrate the advancements and innovations in web scraping and automated data collection techniques. They provide valuable insights and methodologies that have paved the way for the development of our own automated data extraction system.

3. LARGE-SCALE DATA COLLECTION

3.1. A Brief Introduction to Large-Scale Data Collection

Large-scale data collection refers to the process of gathering and analyzing vast amounts of data from various sources. In today's digital age, the availability of large datasets has opened up new opportunities for research and analysis in various domains, including economics, sociology, marketing, and public policy. The abundance of data provides researchers with the potential to uncover hidden patterns, gain insights into complex socio-economic phenomena, and make data-driven decisions.

However, large-scale data collection comes with its own set of challenges. Managing and processing massive volumes of data require advanced tools, techniques, and infrastructure. Traditional methods of data collection may not be sufficient to handle the scale and complexity of modern datasets. This is where automation and web scraping play a crucial role.

Another challenge of large-scale web scraping is the significant amount of time required to gather data, primarily due to the presence of multiple anti-scraping systems and blocks implemented online. It is crucial to be mindful of the actions performed by your web scraper and the potential issues it may encounter while collecting data.

Anti-scraping measures employed by websites and online platforms can include captchas, IP blocking, user agent detection, and rate limiting. These measures aim to prevent automated scraping and protect the integrity of the website's data and infrastructure. However, they can pose obstacles for web scrapers and result in delays or interruptions in data collection.

Automation, through the use of web scraping tools and techniques, enables researchers to efficiently gather data from online sources on a large scale and without needing many human resources. Web scraping involves extracting data from websites and other online platforms, allowing researchers to access valuable information that would otherwise be time-consuming or difficult to obtain. By automating the data collection process, researchers can save time and resources while ensuring a higher level of accuracy and consistency in the collected data.

Web scraping enables researchers to collect data from diverse sources, including social media platforms, e-commerce websites, news portals, and more. The extracted data can be used to analyze consumer behavior, track market trends, study social interactions, and support evidence-based decision-making.

3.2. The Facebook Ads Manager

The Facebook Ads Manager is a powerful tool provided by Facebook for managing and running advertising campaigns on its platform. It offers advertisers a wide range of options to target specific audience segments, set campaign objectives, and track performance metrics. However, beyond its primary purpose in advertising, the Facebook Ads Manager also holds significant potential as a source of data for socio-economic investigations.

Facebook charges advertisers for using their Ads Manager platform to create and run advertising campaigns. However, for the purpose of data collection and analysis, it is not necessary to actually create or publish real ads. Valuable data can be obtained by leveraging the information provided on the ad creation page itself.

When using the ad creation page on Facebook, advertisers are presented with estimated reach figures. These estimates provide insights into the potential audience size that an ad campaign could reach based on various parameters such as demographics, target interests, and geographical regions. By manipulating these parameters, researchers can study the impact and changes in reach estimates, allowing for the analysis of different targeting strategies and their potential effects on audience reach.

By leveraging the data available in the Facebook Ads Manager, researchers can gain valuable insights into user behavior, preferences, and interactions. The platform captures information about user demographics, interests, engagement with ads, and other relevant metrics. This data can be analyzed to understand consumer trends, measure the effectiveness of marketing campaigns, and explore socio-economic patterns.

The Facebook Ads Manager provides researchers with a rich dataset that reflects real-world user interactions and behaviors. By applying analytical techniques and data mining approaches, researchers can uncover valuable information about the impact of advertising, consumer preferences, and societal dynamics. This data can be instrumental in shaping public policies, improving marketing strategies, and advancing our understanding of socio-economic phenomena.

In the next chapters, we will explore how we utilized web scraping techniques to collect data from the Facebook Ads Manager, the challenges we encountered, and the insights we gained through our socio-economic investigations.

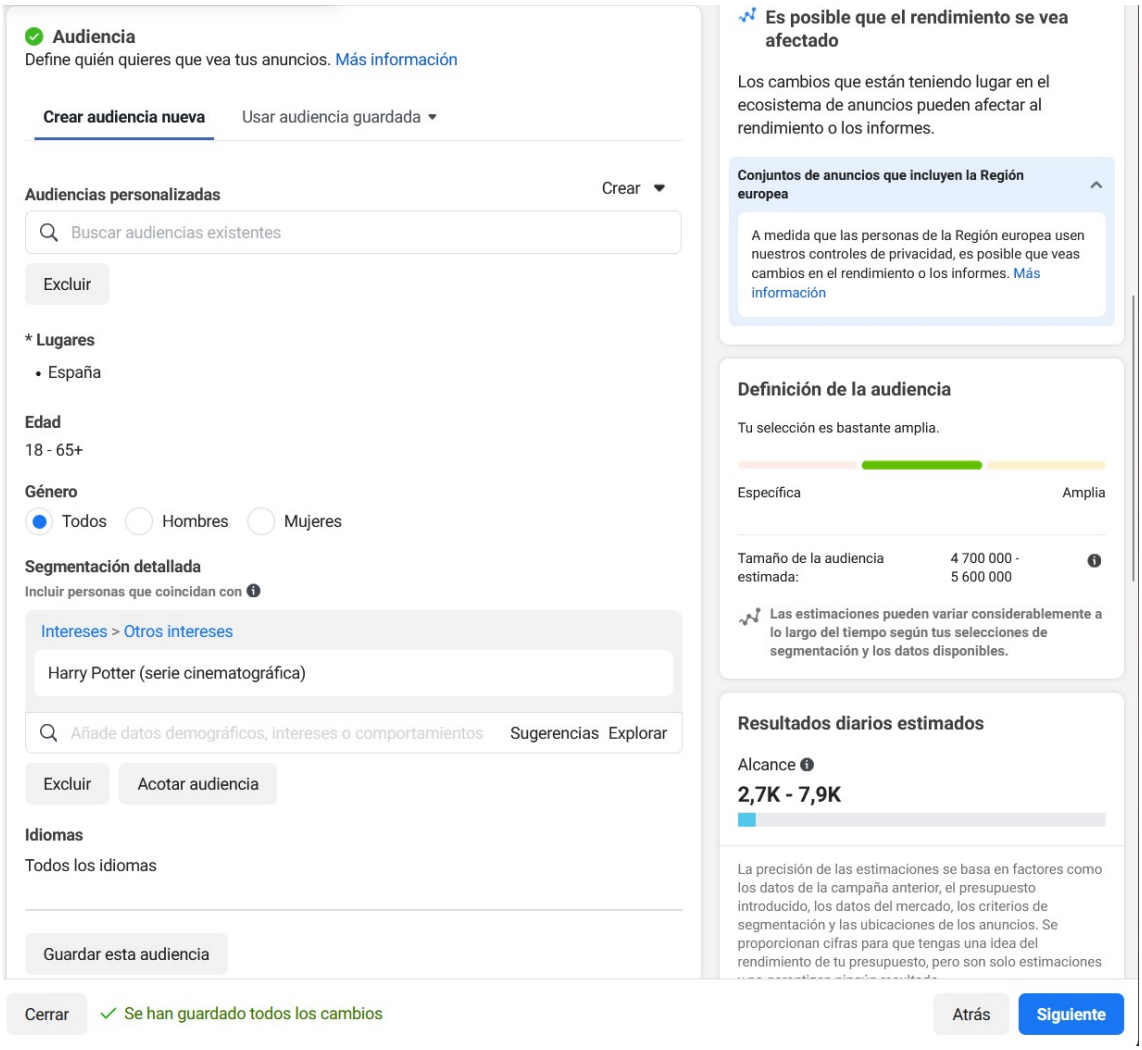


Fig. 3.1. Ads Manager UI and reach estimation

4. DEVELOPMENT OF THE SCRAPPER

In this part, we will follow step by step the development of the scrapping tool as well as the problem encountered, the solutions found to overcome them, and different improvements made to the core project.

4.1. Introduction to the project and task

As part of a research project with researchers from SMU (Dallas, US), we wanted to be able to obtain, as fast as possible, bounded sets of data from the Facebook Ads Manager system in order to study the interests of different demographic groups of people.

To do so, the first thing is required is the account ID (from now on, will be referred to as “act_ID”), as well as the access token assigned by Facebook to that account. In order to obtain those, we captured the HTML Requests and Responses on the Facebook Ads Campaign manager, firstly manually then, using a modified version of the script developed and provided by the research team of my supervisor as we will review in section 4.4

Then we will replicate the query generated by Facebook called “delivery_estimate” (See Figure 4.1) with the desired data, send it with the account data previously mentioned. This will return a JSON file with plenty of info, that we will process and store in different CSV files.

Since we wanted to maximize the rates and throughput, it was required a multithreaded approach, so we can manage multiple queries at a time.

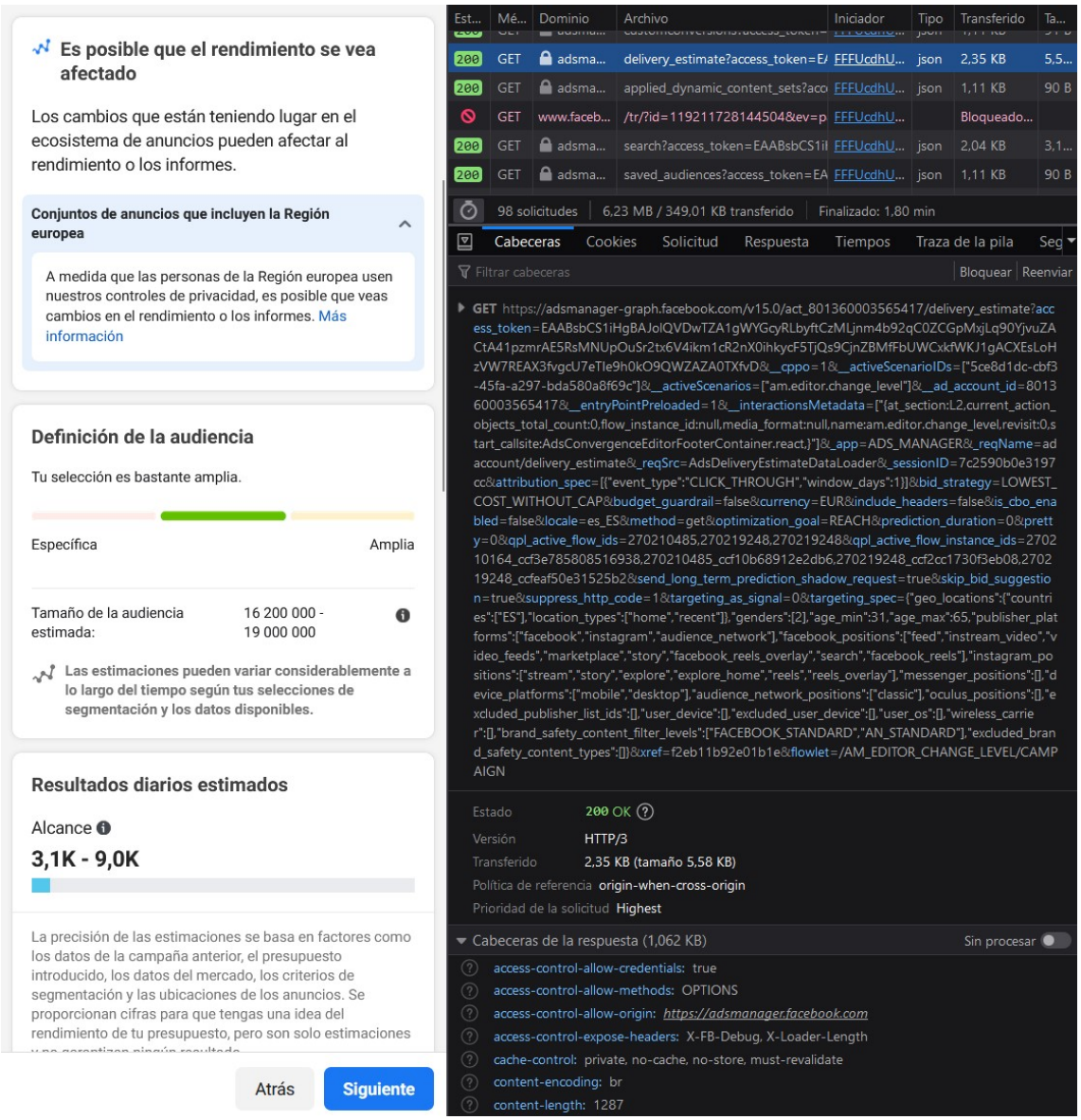


Fig. 4.1. Manual capture of GET request "delivery estimate"

4.2. File and query storage management

Before diving into the main script from the scrapper, it is important to understand how the information is accessed, stored and manipulated previously, during and once finished the execution of it.

Of course, the structure of the file system was modified during the lifetime of the project, but those changes will be covered in their respective following points in chronological order, for simplicity. Therefore, in this point, we will only cover the initial design.

4.2.1. Input related files

On the early stages of the script, input was given by 5 different core files:

- Logging file: This file contained only two rows of data, being the first one the access token, and the second one the act_ID.
- Interests file: This file contained a list of the interests we wanted to obtain the data, each interest in a new line.
- Region/Country files: Facebook divides the terrain in Countries or Regions, and the structure of the queries varies depending on which one we are using, so two separated files stored a list of the country IDs and/or Region IDs, again, each in a new line.

4.2.2. Output related files

These files were, if we are executing the script for the first time, created, if not, accessed, and stored different data returned by the scrapper following the next scheme:

- Output file: CSV file with all the validated data obtained, structured like follows:
  - Interest: This is a unique ID consisting of 11 to 13 numbers that corresponds to different interest tags associated with users based on their usage of Meta products such as Facebook, Instagram, or Snapchat. During the project, more than 40,000 different interest values were used. The wide range of interests collected is significant as it enables us to perform studies on the general interests of the population and cross-reference them with geographical or demographic data to obtain valuable insights.
  - Country: This value restricts the target audience to users known to live in a specific country. The country codes used can be obtained by utilizing the Facebook Graph API Explorer and searching for "search type= adgeolocation&location_types =['country']". This query will return a JSON file with 246 country codes at the time of retrieval.
  - Region: This is a different location ID that provides an additional layer of precision, allowing us to target specific cities, counties, or postal areas within a country.
  - Demographics: In this section, we store the demographic values used to refine our results, such as specific age ranges. This allows us to conduct further studies, such as analyzing certain trends while targeting different age groups, and study how they differ or not.
  - MAU: Monthly Active Users represents the number of users who are active on a monthly basis and match the specified search parameters (interest, country, demographics).
  - DAU: Daily Active Users represents the number of users who are active on a daily basis and match the specified search parameters (interest, country, demographics).

- Log file: This file records the details of the scraping process, including timestamps, requests made, and data retrieved. It helps in tracking the progress of the scraping operation and can be useful for troubleshooting or analysis purposes.
- Error Handling files:
  - Non-existing Interest IDs: This file contains the IDs of interests that do not exist or are invalid. These could be interest tags that have been removed or deprecated by the Facebook platform.
  - Non-existing region/countries: This file lists the region or country codes that are not recognized or do not exist in the Facebook Graph API. It helps in identifying any inconsistencies or inaccuracies in the provided location data.

4.3. Obtaining the Token

As we mentioned earlier, obtaining the token and act_ID (but mainly token since this was renewed/updated more often) was done manually, since we were working with only one account at a time. But as the very title of this document says, we wanted our tool to scale, and as the workload increased, we were required to upgrade our system to use multiple accounts, and in order to obtain those, we decided to base our work on a side-script made by a previous researcher on the same project.

He designed and created a scrapper that, in simple terms, given a file containing a mail and password accessed the ads manager and gets both, access token and act_ID, writing them on a new file.

This approach had several problems for the usage we desired, so we worked in improving those features:

- WebDriver's selection: To automate the process of obtaining the access token and act_ID, we explored various alternatives for the web driver, including Mozilla and Firefox. After thorough testing, we determined that using Firefox was the most suitable choice for our needs. Its stability and robustness, especially when interacting with the Facebook Ads Manager, provided a reliable foundation for our web scraping operations.
- Headless Usage: To ensure the seamless execution of the script and enable its integration with the main web scrapper, it was crucial to run it in headless mode. By running the script in headless mode, it could be executed in the background without a visible browser window, allowing it to operate concurrently with the main web scrapper. Moreover, the ability to run the script in headless mode proved vital when deploying our system on the provided server, as it allowed us to maximize resource efficiency while handling the high throughput and frequency of data requests and retrievals.

- File-based data input to Argument-based data input: Initially, the script relied on command-line parameters to provide the required data for accessing the ads manager, so no complex querying was supported.However, to enhance flexibility and usability, we transitioned from argument-based data input to file-based data input. This involved using configuration files that contained the necessary information, enabling us to perform queries for multiple experiments in a single execution (we could now load many accounts, and it will return all the tokens accordingly). This transition not only streamlined the process, but also standardized the crawler system, making it more user-friendly and adaptable to different scenarios.

4.4. The initial scrapper (v1)

In this section, we will cover the first version of the scrapper, as well as the different modifications and upgrades performed on the main skeleton of the "first version" of our scrapping tool (We consider as the first version the base structure, as well as all the adjustments made on it prior to the multi-account oriented implementation.)

In order not to start from scratch, I built upon the existing work from the research team of my supervisor, who had been previously tested on similar projects [17] and whose code and guidelines were provided to me. The team had already made some initial attempts to query the Facebook API, which served as the foundation for my own work. By using their previous work as a starting point, I was able to build upon his knowledge and progress to further develop the project. This allowed me to approach the project with a greater understanding of the problem at hand, and gave me a head start in terms of the development process. Ultimately, their previous work proved to be instrumental in helping me achieve my goals and make meaningful contributions to the project.

For this first part of the project, we want to be able to target specific groups of people, and be able, on top of the queries we are already able to make, to specify gender and age group biases, without altering the performance, if not improving it as much as possible. We also want to be able to make mixed requests for regions and countries, without having to split these into two separate runs. To do so, the following changes were made:

4.4.1. Parameters

Arguments received were expanded in order to handle new features, like gender and age selection. Now, when calling our program we should include:

- Files: Logging file, Interests file and both Regions and Countries files paths or names if they are in the same folder should be indicated.
- Country/region flag: In order to select if we want to use countries, regions or both, we should input the character "c", "r" or "all"

- Gender flag: In order to select the gender bias (baselines will always be generated independently since for statistics purposes are crucial), we must input the character "m" for male, "f" for female or "all" again for both

- Age flag: Here we are able to specify an age range for our queries (Again, baselines will always be generated independently since for statistics purposes are crucial). This range will be indicated as "Age1,Age2".

  Note that, when handling this argument, a fix was introduced. When handling ages below 13 and above 65, those values were restored into 13 and 65 (since those are the maximum values Facebook API works with).

4.4.2. Link generation function

In order to develop and improve the query generation function in this script, it was essential to study the mechanism behind the Facebook API's requests/responses on the advertiser's platform. To do so, our first step will be accessing the platform and selecting the option to create a new ad. This led us to a page where we had to input the advertiser's information. Once this was done, we accessed the estimation platform and captured the HTML packets generated by Facebook.

Between the various packets generated by Facebook API, we are particularly interested in the one that provided the estimate for the ad's reach. This estimate is used by Facebook Ads Manager to represent the ad's reach in a graph in the lower right corner. By studying the structure and content of this HTML packet, we were able to obtain a basic template that we will use for our queries later on. This template also served as the base and guide for developing the function that would generate all the queries used to get and evaluate the necessary data from the Facebook API.

With this template in hand, we proceeded to create a function that will fill in the login data, including the token and act_id, as well as the desired demographic parameters such as gender and selected age ranges if required. For each country or region, we will generate the corresponding URL, and for each URL, we also create a baseline that includes the corresponding regions and countries. This results in the creation of multiple URLs, each tailored to a specific demographic, which could then be used to query the Facebook API for the desired data.

Parameter	Value
access_token	**********
__ad_account_id	**********
__interactionsMetadata	ADS_MANAGER
_reqName	adaccount/delivery_estimate
_reqSrc	AdsDeliveryEstimateDataLoader
_sessionID	**********
bid_strategy	LOWEST_COST_WITHOUT_CAP
currency	EUR
method	get
optimization_goal	REACH
targeting_spec	“countries”:"ES",“genders”:2,"age_min":31,"age_max":65

TABLE 4.1. DELIVERY ESTIMATE PAYLOAD

4.4.3. Request and response analysis

Once we have generated the queries and stored them in a to-do list, the next step is to request them from the Facebook API and verify the responses to ensure that the data is accurate and error-free. To achieve this, we follow a series of steps that ensure the quality of the data obtained from the API.

1. First, we check whether the query we are requesting exists and is available for a particular country. We do this by reading two data files, “nonexisting_ids_**date**.csv” and “nonexisting_regions_countries_**date**.csv”, which contain all the non-existent or flagged countries and regions values. We check that none of the values in the queries we are examining appear in these files before proceeding with the request. This step ensures that we are not wasting time and resources on non-existent or unavailable data.

2. Next, we prepare to request the URLs by emulating headers for our request library. These headers are non-compulsory, but adding "personalized/human" data to the requests increases the likelihood of successful queries. We use a user-agent header, as shown in the example below:

```
headers = {'user-agent': 'Mozilla/5.0 (Macintosh; Intel
    Mac OS X 10_14_3) AppleWebKit/537.36 (KHTML, like
    Gecko) Chrome/74.0.3729.157 Safari/537.36'}
```

3. Finally, we use a RateLimiter to control the querying speed and avoid overloading the API. The RateLimiter limits the number of requests made within a specified period. We have determined that the best values for maximum calls are 700/5,

which limits the number of requests to 700 every 5 seconds, and a period of 10, which ensures that the requests are spread out evenly over time. By using a rate limiter, we can avoid getting kicked out of the API and ensure a steady flow of data.

Now that we have prepared the necessary headers and set up the RateLimiter, we can proceed to the request and validation part. For this task, we are using the requests' library for Python, which, given a URL and some headers, will return the response from the Facebook API. However, before reading into the file, this library also returns the HTTP status code associated with that response, so we will firstly check these codes and act accordingly.

According to the HTTP specifications [18], status codes in the 2XX range indicate success, with the standard response for a successfully completed query being 200 OK. Therefore, if we get this code, we can trust that the query is successful, and we will read the JSON file that is returned. To extract the estimated MAU and DAU from this file, we have studied the structure of these replies and know that they can be found in the keys ['data'][0]['estimate_dau'] and ['data'][0]['estimate_mau']. We will provide more details on the structure in the version 2 of the script.

However, we also need to handle error status codes, such as those in the 4XX and 5XX range. A status code in the 4XX range means a client error, while a code in the 5XX range is a server-side error. In both cases, we will read the text response and try to search for the reason of the error.

If we get "Invalid country code" in response to a query in the 4XX range, then we add the country code to the "nonexisting_regions_countries_**date**.csv" file. Similarly, if we get "Invalid Regions" in response to a query in the 5XX range, we will add the code to the same file. If none of these errors are detected, or if the status code is different from those mentioned above, we will register those queries in the "nonexisting_ids_**date**.csv" file.

This function previously defined and explained will be called by a Pool of 20 worker processes, applying the function to each URL in the total URL list in parallel, and then waiting for all the worker processes to complete before terminating them and exiting the program. It's important to remind that using 20 processes here is possible, since we're running this script on a big server, but this value should be adjusted if we wanted to make it run in a different environment.

4.4.4. Scrapper Workflow

In this subsection, we will provide an overview of the script workflow, aiming to summarize the key aspects discussed in this chapter and offer a visual representation that enhances the understanding of the overall functioning.

Figure 4.6 illustrates the main flow of the script, showcasing the functional design for retrieving a complete set of URLs. The workflow begins with the script being called and proceeds to read the parameters, which include the address of the input files. Subsequently, the output files are generated, and the core URL components are established. These core components are independent of the parameters or request characteristics.

Next, the script iterates over each characteristic, ensuring the generation of all desired combinations. With each iteration, the subsequent parameters are updated accordingly. Once the set of URLs is ready, the script proceeds to create a pool of threads and initiates the process of requesting each URL. Upon receiving the responses, the script analyzes the results and saves them accordingly.

This workflow design ensures a dynamic and varied pattern of user interactions, contributing to the appearance of natural and human-like behavior within the system. By systematically generating combinations of characteristics and utilizing multi-threading for efficient URL retrieval, the script optimizes the process of data extraction while mimicking real user behavior.

The visual representation provided in Figure 4.6 offers a comprehensive overview of the script workflow, emphasizing the sequential steps involved in retrieving URLs and processing the obtained data. This visual perspective enhances the understanding of the script's functionality and reinforces the significance of its design in achieving efficient and realistic web scraping operations.

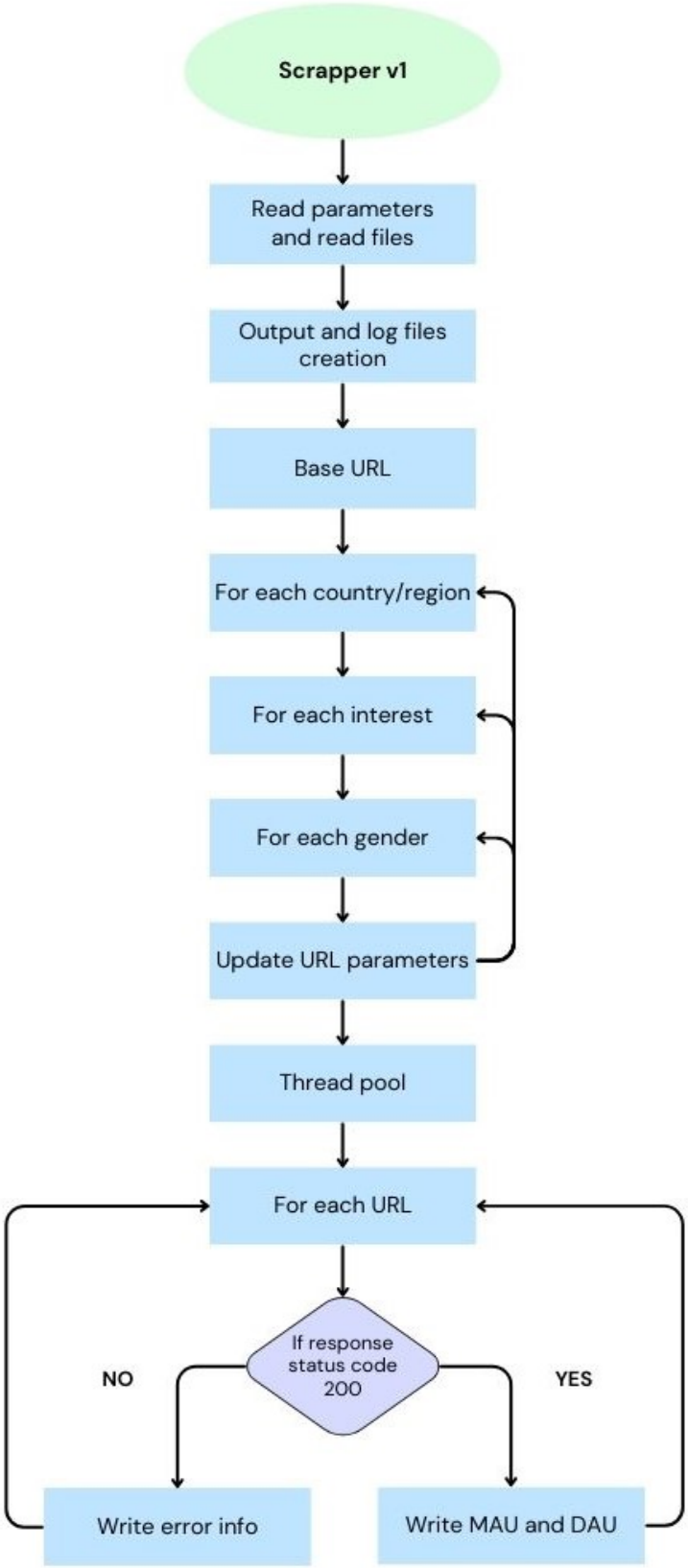


Fig. 4.2. Flowchart for first version of the script

4.5. Scrapper (v2) with multi-account rotation

In the following section, we present an overview of the enhancements and improvements made to the initial version of the script. Our main focus was to optimize the performance of the script while ensuring accurate results. We have implemented several new techniques and design decisions to achieve this objective.

During the development and lifetime of the script, we encountered several changes made to the Facebook API. These changes necessitated several modifications to the script to prevent high volume querying, and to ensure that the script remained functional despite the changes made to the API. We identified the critical areas that needed modification and made the necessary updates to the script.

Furthermore, I took a proactive approach to improve the script's overall quality by restructuring and refactoring it. We aimed to create a cleaner and more straightforward approach to the script, making it easier for users to navigate and understand the code. By refactoring the script, we were able to remove redundant code and optimize its performance.

In summary, this section provides an in-depth look at the enhancements and improvements made to the script, including the techniques and design decisions made to achieve better performance and results. The modifications made to the script were not only reactive, but also proactive, anticipating future changes to the Facebook API. Finally, the script was restructured and refactored to provide a cleaner, more efficient, and user-friendly approach.

4.5.1. Script restructuring

As mentioned before, in order to improve understanding and increase usability, a restructuring of the main script was undertaken. This restructuring also allowed for the introduction of new functionalities, which will be described below. The following changes were made and were necessary to enhance the overall system.

Query Generator and Config Files Implementation

Similar to the token extraction script, configuration files were implemented instead of using parameters. This shift was crucial for conducting complex queries and automating the query generation process.

Due to the increased size of the requested data, the computational load of generating all the queries became significant. To mitigate long wait times and increase parallelization, a separate script was created called "to-do-generator." This script reads values from different configuration files, such as demographics, interests, countries, and regions, and generates a grid with all these parameters. It then iterates over the grid, generating a

query for each unique combination of values. This systematic approach allowed for the generation of all desired queries efficiently.

CÓDIGO 4.1. To-do generator script

```
grid = {'gender': gender ,'country': country_list2,'interests':
    interests2,'age':list(ages_chart.keys())}
for grid in ParameterGrid(grid):
    country=grid['country']
    interest=grid['interests']
    gender=grid['gender']
    age=ages_chart[grid['age']]
    with open(todo_list,'a') as todo:
        todo.write(country+','+interest+','+gender+','+age_finder(
            ages_chart,[country,interest,gender,age])+'\n')


rate_limiter= RateLimiter(max_calls=700/5, period=10)
import csv

todo=[]
with open(todo_list,'r') as todo_csv:
    csv = csv.reader(todo_csv, delimiter=',')
    for row in csv:
        row[3]=ages_chart[row[3]]
        todo.append(row)
```

Additionally, a null/0 value was introduced for each parameter to create a baseline value for all possible combinations. For example, if we are requested to obtain values for "Harry Potter," "18-40" years old, and "ES" (Spain), we would obtain the query for that specific combination. However, with the inclusion of the baseline parameters, we would also obtain queries for "Harry Potter" in "ES" with any age range, "Harry Potter" in the "18-40" age range in any country, and lastly, just "Harry Potter" without any demographic or geographic restrictions. This flexibility allowed the adaptability of baselines based on the requirements of specific investigations. The baseline additions can be configured via parameters when calling the new generator script.

By separating the query generation process into a dedicated script, the main scrapper only needs to read the set of to-do queries and generate the subsequent links. This approach streamlined the main scrapper's functionality and improved its efficiency.

In the following subsection, we will explore how we implemented a tracking mechanism to monitor the progress of requested and remaining queries.

Tracking mechanism and safe restart procedure

In this section, we address the development of the recovery methodology, created to patch an incident that resulted in the loss of significant amounts of data during our data extraction process (See section 5.2.2). To mitigate future risks and ensure the preservation of valuable extraction time, especially considering the high cost of server usage and potential unexpected script failures, we recognized the importance of developing a robust tracking mechanism and recovery procedure.

To achieve this, we implemented a four-file system control configuration. Firstly, we have the "to-do" file, which is generated by the script discussed in the previous section and contains all the queries that need to be extracted. Secondly, the "done" file keeps track of the queries that have already been successfully executed. Thirdly, the "remaining" file compiles the queries that are yet to be completed. Lastly, we introduced a "log" file that acts as a backup, also storing the IDs of the queries marked as "done" and adding some redundancy to the system as a protection measure.

CÓDIGO 4.2. Recovery functionality

```
  def rem_writer(row):
        if row not in done:
            with open(remaining_list,'a') as remaining:
                remaining.write(row[0]+','+row[1]+'\n')

    if(os.path.exists(remaining_list)):
        print ("remaining found")
        with open(remaining_list,'w') as remaining:
                remaining.write('')

        print(' Processing the remianing queries ')
        with Pool(50) as p:
            records = p.map(rem_writer, tqdm(todo))
            p.terminate()
            p.join()
            p.close()

        print(' End remaining mapping ')
        shutil.copyfile(remaining_list, todo_list)

    else:
        print("remaining not found")
        with open(remaining_list,'w') as remaining:
                remaining.write('')

        shutil.copyfile(todo_list, remaining_list)
```

During script execution, the system checks for the presence of these files in the designated directory. If the "remaining" file is absent, it signifies a new execution, and the system creates the "remaining" file by copying all the data from the "to-do" file (since everything is now considered "remaining"). The script then proceeds with its normal execution, updating the "done" file as queries are completed.

However, if the "remaining" file exists, the system recognizes that a previous execution was interrupted and enters recovery mode. In this mode, a thread pool of 50 threads is launched to compare the values in the "to-do" file with those in the "done" file (See the function at code 4.2. This is required due to the huge volumes of queries we are working with, that ranges between 50k to 2.5 million queries. Any queries not found in the "done" file are considered remaining and are added to the "remaining" file for subsequent processing. To optimize efficiency, we implemented a smart strategy to shorten the search process. At the end of the recovery mode execution, the "to-do" file is overwritten with the computed remaining values. This exponentially reduces the search time in subsequent executions without losing any valuable data.

By implementing this tracking mechanism and the safe restart procedure, we ensure the integrity of the data extraction process, minimize potential data loss, and optimize the execution time of the script.

4.5.2. Account analyzer

In this section, we will discuss one part of the main significant improvement made in the second version of the script, which was the ability to handle and utilize multiple account tokens. The primary goal was to achieve a higher request rate without risking account suspension or blocking. A key aspect of this enhancement was implementing a mechanism to check the status of each account, determining if it was blocked or available for use.

In our initial approach, we simply fed the accounts into the system, and if certain errors occurred during querying, we marked those accounts as blocked to be sent to idle. However, this approach proved problematic due to the high-speed querying and multithreaded nature of the system. Requesting data from potentially blocked accounts triggered Facebook's block detector, resulting in longer wait times between availability of the accounts, as explained in Facebook Business Guidelines [19]. This led to wasted time and resources, which was unacceptable considering the value of every second in the data extraction process of such a volume of queries.

To address this issue, we re-designed the system to prioritize the use of available accounts and implemented a less intrusive method to check the status of blocked accounts. This involved creating the "account analyzer" component, which is executed when an account is blocked. Before selecting the next account for use, the analyzer checks all the accounts marked as idle to determine if they have become unblocked (See code 4.3).

CÓDIGO 4.3. Account Analyzer functionality

```
def acount_analyzer(path,new_path):
        trial_row=['10101', '6003107902433']
        acounts=login_set_creator(new_path)
        for acount in acounts:
            r=request_query(link_gen(trial_row,acount[2]))
            [verified,valid]=response_verifier(r)
            if verified:
                print('Account '+acount[1]+' is working')
                shutil.move(os.path.join(new_path, acount[1]),os.
                    path.join(path, acount[1]))
                resp_analyser_time(r,trial_row)
```

The account analyzer performs a single, generic request to each idle account and analyzes the success or failure of the query. Based on the outcome, the status of the accounts is updated, either moving them to the available pool or keeping them in the idle state.

By incorporating the account analyzer into our system, we ensure that only available accounts are used for data extraction, minimizing the risk of account blocks and optimizing the overall efficiency of the process.

4.5.3. Rotation and RR implementation

In the "Rotation and RR implementation" section, our objective was to create a multi-account web scraper that could efficiently rotate between accounts. To achieve this, we initially implemented a round-robin (RR) algorithm for account rotation. The RR algorithm is a simple scheduling algorithm that cyclically distributes resources among a set of participants. It ensures that each participant takes turns being used in a sequential manner, thereby evenly distributing the workload. [20]

The implementation of the RR algorithm allowed us to rotate through the available accounts, preventing over utilization of certain accounts while others remained underutilized. This approach was beneficial for maintaining a balanced usage of accounts and reducing the likelihood of account blocks due to excessive activity on a single account.

However, during the implementation and testing phase, we observed that not all accounts were blocked in a uniform manner. Some accounts demonstrated greater resistance to blocking than others, potentially due to varying usage patterns or account characteristics. [21] This finding led us to reconsider the fixed rotation approach based on a predetermined number of queries.

We recognized that rotating accounts after a fixed number of queries, such as every 5000 queries, could still result in occasional blocks. This approach led to unnecessary rotations and underutilization of the more resilient accounts, which impacted the overall efficiency of the web scraping process. To address this issue, we decided to adopt a slightly more performance-oriented approach.

In order to maximize the request capabilities of each account and test the limits of Facebook API blocking procedures, we relied on the presence of a stable source of tokens maintained by an automated bot system. This system trained our accounts to exhibit more human-like behavior, strengthening the credibility of our tokens and enabling us to scale our reach by utilizing a larger number of accounts.

Considering the potential trade-off between occasional blocks and increased request capabilities, we decided to push the limits and perform a continuous limit test on the Facebook API blocking procedure in terms of speed and volume. This approach allowed us to fully exploit the potential of each account, even if it meant encountering some blocks along the way. By thoroughly testing the boundaries, we aimed to achieve optimal performance and maximize the efficiency of our web scraping process.

CÓDIGO 4.4. Login Set Creator

```
    def login_set_creator(path):
        acounts=[]
        for i,file_name in enumerate(os.listdir(path)):
            log=os.path.join(path, file_name)
            acount=[i,file_name,logging_reader(log)]
            acounts.append(acount)
        random.shuffle(acounts)
        return acounts
```

To facilitate the account rotation and management, we designed a request loop with the following structure. First, we invoked the login set creator (See code 4.4), whose primary function was to create a set of accounts available for login. This process involved reading the accounts from the "available" directory, creating a list of accounts, and shuffling them to introduce randomness into the account selection process.

Subsequently, we moved into the main function, which played a central role in our system. It handled all the key calls, including making requests for queries, analyzing the obtained results, and performing the necessary account swaps when required. For now, we will focus specifically on this last part and detail the final build.

When the function row_analyzer is called, it takes three arguments: the row being processed, the list of accounts, and the identifier "i" indicating the current account being used. This identifier allows us to track and manage the active account during the web scraping process.

Once the reply is collected, the program proceeds with the necessary steps if the gathered data is correct or contains an error unrelated to account blocking. These steps may involve saving the data in the "done" folder or recording the error for further analysis. However, if an error related to account blocking is detected, immediate actions are taken to address the issue.

First, the program moves the account token file into the "blocked" folder to isolate the affected account. Then, the account analyzer is invoked to check the status of all accounts, including both the blocked and active ones. This analysis ensures the overall health of the accounts by identifying any previously blocked accounts that have become unblocked. Next, the login set function is called to create a new list of accounts, incorporating any unblocked accounts and shuffling the list to introduce randomness.

This approach guarantees a fast and efficient mechanism for maintaining the account health and rotation. By promptly addressing account blocks and periodically analyzing the account status, we ensure that all accounts are utilized optimally and contribute to a randomized and unpredictable performance during the web scraping process. Finally, the "i" identifier is updated to reflect the current account being used, and the program proceeds to process the next row.

The combined implementation of the account analyzer, login set function, and account rotation strategy allows for robust account management, effectively handling account blocks and maintaining a dynamic and diverse account selection.

4.5.4. Burnt account token and recovery token time measurements

In this section, we will discuss the measurement of idle time for accounts measured, and the implementation of a score system to improve future performance, although it was not utilized in the final version due to project completion.

Firstly, we will analyze the mean idle times measured throughout the duration of the project, as well as the frequency of account blocks, which have been normalized into blocks per day for ease of comparison across different stages of the project. To observe the evolution of these metrics, we will present three separate graphs corresponding to the different project stages mentioned: (1) One account script, (2) Early multi-account with basic round-robin (RR) implementation and no account check, and (3) Final script version.

The first combined graph (Figure 4.3) will illustrate the mean idle times experienced during each stage of the project. By examining this graph, we can observe any variations in idle times between the different stages and assess the effectiveness of the implemented strategies in reducing idle periods.

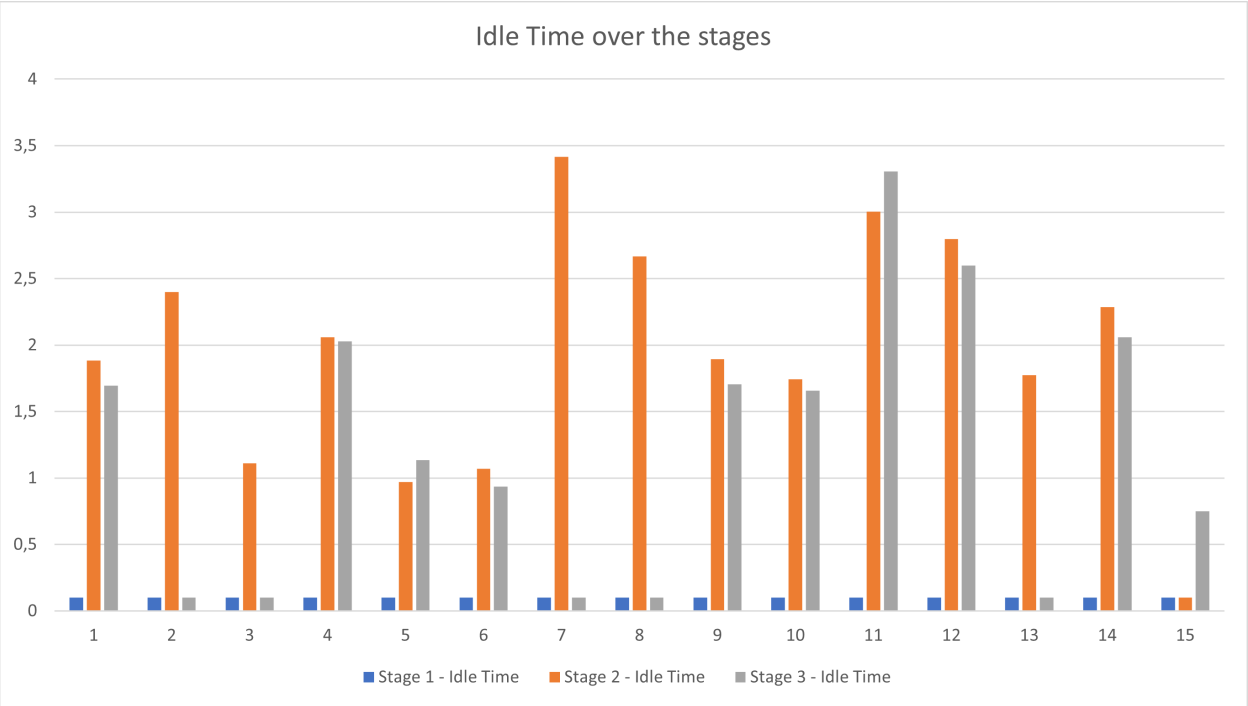


Fig. 4.3. Mean idle time across project lifetime

The second combined graph (Figure 4.4) will focus on the frequency of account blocks, measured as blocks per day. This metric allows us to compare the rate of account blocks across the various project stages. By analyzing this graph, we can identify any patterns or trends in the occurrence of blocks and evaluate the impact of different approaches on account stability.

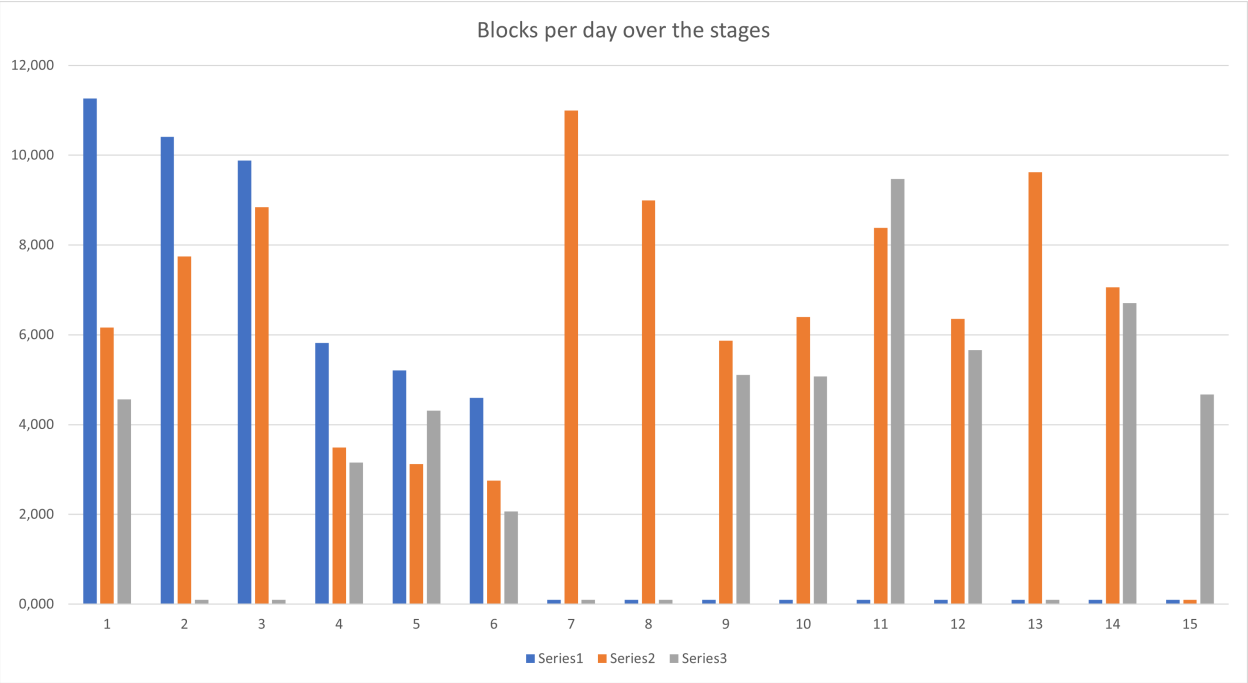


Fig. 4.4. Mean account blocks per day across project lifetime

Regarding the score development, it was computed based on the following metrics:

1. Mean of time in idle state prior to reactivation: This metric measures the average duration an account remains idle before being reactivated. A smaller value indicates a more efficient utilization of the account.
2. Times sent to idle: This metric tracks the number of times an account is sent to the idle state. A lower value indicates better account stability and a reduced frequency of being blocked.
3. Total number of queries performed: This metric captures the overall count of queries executed by an account. A higher value indicates a higher request throughput and better utilization of the account's capabilities.

Total number of queries performed: This metric captures the overall count of queries executed by an account. A higher value indicates a higher request throughput and better utilization of the account's capabilities.

For each account, these metrics are recorded when the account is blocked, and the score calculation is performed as an additional step in the "login set" function. The weightings assigned to these metrics for score computation are as follows: 40% for the total number of queries, and 30% each for the mean idle time and the times sent to idle.

By incorporating these metrics into the score system, we aimed to evaluate the performance of each account, identify potential bottlenecks, and make informed decisions regarding account rotation and reactivation. However, due to the completion of the development phase at the end of the project, the score system was not fully implemented in the final version.

The measurement and analysis of idle time and the development of the score system provided valuable insights into the usage patterns and performance of the account tokens. Although not implemented in the final version, these metrics can serve as a foundation for future enhancements and optimization of the account rotation strategy.

4.5.5. Scrapper Workflow after updates

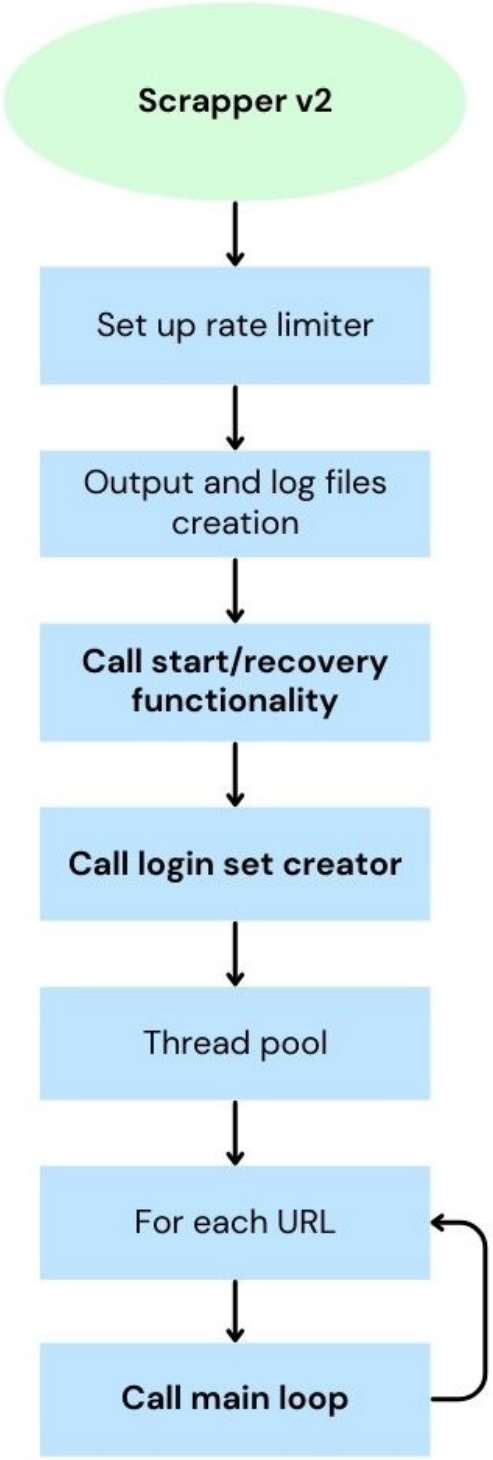


Fig. 4.5. Flowchart for second version of the script

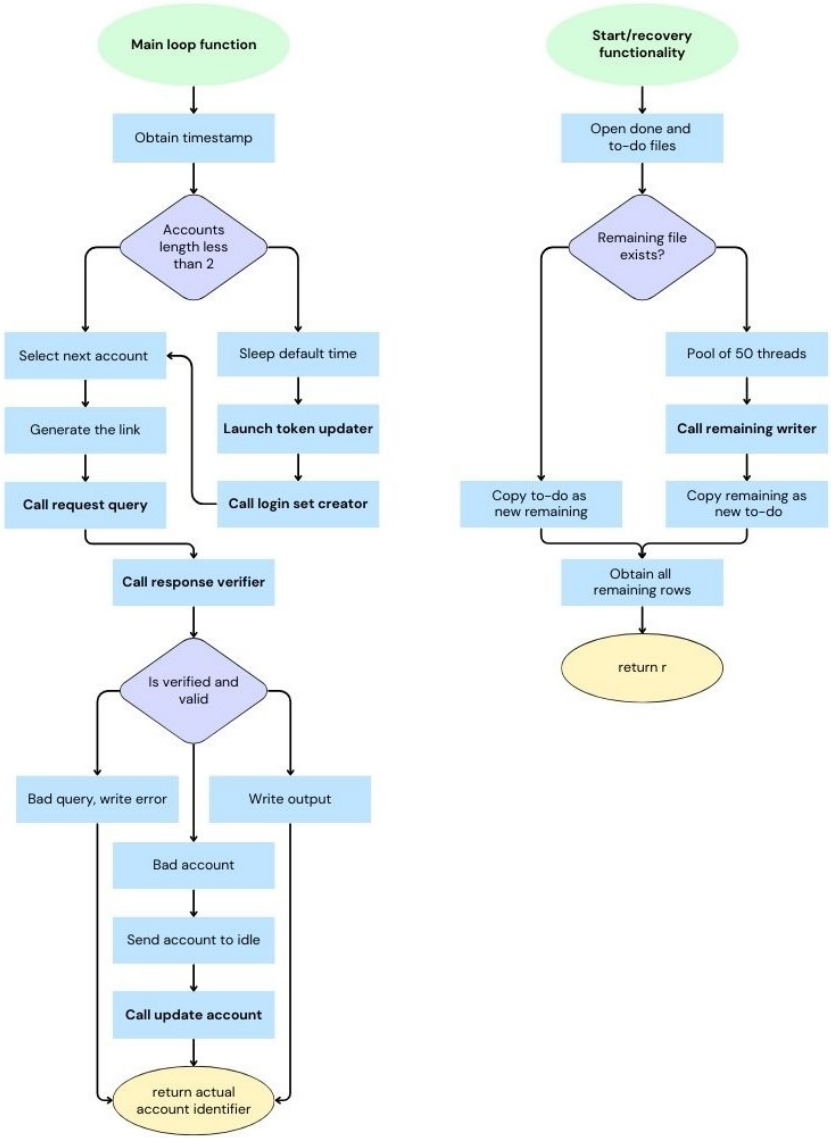


Fig. 4.6. In-detail main and start/recovery functionalities

4.6. Autonomous functionality

In the following section, we will delve into the concept of autonomous functionality, which plays a crucial role in ensuring the continuous and uninterrupted operation of the script. Given the need for the script to run 24/7 without human intervention, it becomes essential to address scenarios where the script may encounter errors or unexpected situations, leading to shut down or crashes. To tackle this challenge, we have developed an autonomous script that handles such occurrences by automatically relaunching the main script.

To achieve this autonomous functionality, we leverage the capabilities of the subprocess library, specifically utilizing the Popen functionality. This feature enables us to

launch and control external processes from within a script. By utilizing Popen, we can monitor the execution of the main script and promptly respond to any errors or abnormal terminations.

When an error or an unhandled situation occurs, the autonomous script detects the issue and initiates the relaunch process by executing the main script again. This approach ensures that the script remains operational, even in the face of unforeseen errors or system interruptions. (See code 4.5)

CÓDIGO 4.5. Auto launch script

```
    while True:
    p = Popen("python " + filename, shell=True)
    p.wait()
    if p.returncode != 0:
        print("Script ended with an error. Relaunching...")
    else:
        break
```

The autonomous functionality implemented in our script provides a robust and reliable mechanism for maintaining continuous operation without requiring manual intervention. It serves as an essential component in ensuring the script's resilience and ability to handle various scenarios, allowing for uninterrupted data retrieval and processing.

By employing the subprocess library and leveraging the capabilities of Popen, we have successfully created an autonomous script that proactively manages errors and relaunches the main script, guaranteeing a seamless and continuous web scraping operation.

5. PERFORMANCE EVALUATION

This chapter focuses on the performance evaluation of the developed system, specifically analyzing the output obtained from the queries made using the Facebook Ads API. The chapter also discusses issues encountered during the development process due to incorrect structuring and explains how these issues were addressed and resolved.

5.1. Output format and post-processing

In this section, we examine the format of the output data received from the Facebook Ads API and discuss the tools developed to clean and process the data for usability. To illustrate the analysis, we present an example query and its corresponding output (see in Table 5.1).

country	interest	gender	age	point	predict	predi	reach_lower_bound	reach_upper_bound	actions_lower_bound	actions_upper_bound	reach	bid	spend	impressions	actions	dau	mau	resp_tp
AE	6003139953447	1	AgeGroup1839	0	0	0	0	0	0	0	0	0	0	0	0	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	1	1.7	1.7	7.4539305731191	21.541859356314	0	10	12.671681974303	4	0	22.302160274772	0.069969133901688	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	2	1.7	1.7	221.05040782672	638.83567861921	0.53083953443414	10.530839534434	375.78569330542	21	72	1217.3916892398	5.5308395344341	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	3	1.7	1.7	494.93640194574	1430.3662016232	6.1363079335412	17.733929927934	841.39188330776	43	221	2647.6187688294	10.43172348702	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	4	1.7	1.7	1195.6205405161	3455.3433620915	12.586951166831	36.376288872142	2032.5549188774	213	1199	6026.217027012	21.397816983613	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	5	1.7	1.7	1480.0636323004	4277.383897348	14.052443213699	40.611560887591	2516.1081749106	426	1859	7283.5552992949	23.889153463289	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	6	1.7	1.7	1558.3396558829	4503.6016055015	14.508929621992	41.930806607558	2649.1774150009	639	2217	7663.7361932692	24.665180357387	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	7	1.7	1.7	1587.1565904245	4586.8825463268	14.692710184755	42.461932433943	2698.1662037217	853	2453	7843.6944444151	24.977607314084	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	8	1.7	1.7	1612.1975500707	4659.2509197044	15.024511799008	43.420839099132	2740.7358351202	1279	2925	8115.4982787632	25.541670058313	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	9	1.7	1.7	1631.4092021678	4714.772594265	15.082693046063	43.588982903123	2773.3956436853	1705	3065	8204.2006417238	25.640578178308	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	10	1.7	1.7	1634.8040245849	4724.5836310503	15.130879262015	43.728241067222	2779.1668417943	2131	3175	8264.6056835663	25.722494745425	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	11	1.7	1.7	1634.8051526048	4724.5868910279	15.135707243618	43.742193934055	2779.1687594282	2558	3194	8277.3973439659	25.73070231415	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	12	1.7	1.7	1635.3537539381	4726.172348881	15.146645826824	43.77380643952	2780.1013816947	2984	3241	8288.3884508951	25.7492979056	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	13	1.7	1.7	1637.974849197	4733.7473141792	15.153624672546	43.793975303657	2784.5572436348	3410	3279	8296.2307679098	25.761161943328	5881	8000	2021-03-30 08:40:54.023947
AE	6003139953447	1	AgeGroup1839	14	1.7	1.7	1640.5959444559	4741.3222794775	15.160603518268	43.814144167794	2789.013105575	3836	3320	8304.0730849244	25.773025981055	5881	8000	2021-03-30 08:40:54.023947

Fig. 5.1. In-detail example output query

The example query used is as follows: "Daily active users (DAU) and Monthly active users interested in 'Hearts' (code: 6003139953447), living in the United Arab Emirates (Country code: AE), who are male (Gender code: 1) and aged between 18 and 39 years old." As shown, the obtained output contains various columns apart from those required, and we will provide a breakdown of the meaning of each of those columns:

- **Country**: Represents the country targeted in the query (in this case, United Arab Emirates - AE).
- **Interest**: Indicates the specific interest targeted in the query (in this case, "Hearts" - code: 6003139953447).
- **Gender**: Refers to the gender targeted in the query (in this case, male - Gender code: 1).
- **Age**: Specifies the age range targeted in the query (in this case, between 18 and 39 years old).

- **Point**: Given that the query we are executing returns a graph curve consisting of 14 points, it is important to note that the values of Daily Active Users (DAU) and Monthly Active Users (MAU) remain constant throughout the curve. Therefore, for the purpose of our analysis and to streamline the process, we only need to consider a single point from the curve. While any point could be selected, we have chosen to use the 0th point for clarity and efficiency.
- **Predicted Errors Conversions**: Indicates the predicted errors related to conversions.
- **Predicted Errors Reach**: Represents the predicted errors related to reach.
- **Reach Lower Bound**: Specifies the lower bound of the estimated reach.
- **Reach Upper Bound**: Specifies the upper bound of the estimated reach.
- **Actions Lower Bound**: Indicates the lower bound of the number of actions taken.
- **Actions Upper Bound**: Indicates the upper bound of the number of actions taken.
- **Reach**: Represents the actual number of unique individuals or users who have been exposed to a particular advertisement or campaign. It indicates the extent of the audience reached by the advertising efforts. Reach is an important metric in advertising, as it helps measure the potential impact and visibility of the message among the target audience.
- **Bid**: Refers to the cost associated with the bidding process, which represents the price paid by the advertiser for each action. Specifically, it represents the cost of the Cost Per Mille (CPM) or Cost Per Click (CPC) that the advertiser is willing to pay. The bid reflects the advertiser's investment in reaching their target audience and capturing user engagement.
- **Spend**: Refers to the amount of money that has been spent on a specific advertising campaign. It represents the total cost incurred by the advertiser for running the campaign, including factors such as ad placement, targeting, and duration.
- **Impressions**: Number of times an ad or content is displayed or shown to users. It represents the total count of instances in which the ad or content has been viewed or appeared.
- **Actions**: Number of desired user interactions or engagements expected from the campaign. These actions can vary depending on the specific goals of the advertising campaign, such as visiting a website, installing software, making a purchase, filling out a form, watching a video, or any other predefined user activity. The number of actions provides insights into the level of user engagement and the effectiveness of the campaign in achieving its desired objectives.

- **DAU**: Refers to the number of Daily Active Users.
- **MAU**: Represents the number of Monthly Active Users.
- **Resp_tp**: Indicates the response time and date.

To improve the usability of the data and focus on the relevant information for the investigation and, even if the extra data was extracted on purpose and stored in case it could be needed or required for future researches, a refinement process was performed on the raw data. This process involved filtering out these unnecessary columns and retaining only the data that was specifically requested or required. The refined output is presented in Table 5.1, showcasing a more concise and targeted representation of the queried data, using the previous example.

country	**interest**	**gender**	**age**	**dau**	**mau**	**resp_tp**
AE	6003139953447	1	AgeGroup1839	5881	8000	2021-03-30 08:40:54.023947

TABLE 5.1. FINAL OUTPUT DEMO ROW

In order to achieve the desired data processing tasks, we developed a new script that consists of a set of tools. These tools handle the output files obtained from various executions and merge them into a single file.

Additionally, they perform cleaning operations to remove unnecessary columns and repeated rows. Throughout the development process, we experimented with different tools, and we implemented them as function-based scripts, making it easy to invoke them as needed.

For the data processing tasks, we extensively utilized the powerful and efficient pandas library. Among the various tools created, we would like to highlight the following:

- Dropping Unnecessary Rows: This tool focuses on eliminating redundant rows from the dataset. Specifically, we drop all rows that do not have a point value of 0. Since we are primarily interested in the Daily Active Users (DAU) and Monthly Active Users (MAU), which do not vary across different points, removing these duplicate rows streamlines our dataset. Please refer to Code 4 for the implementation details.
- Dropping Unnecessary Columns: In this tool, we extract the header of the file and then eliminate the columns that are not relevant to our investigation. By removing these unnecessary columns, we further refine the dataset to contain only the essential information.

Both of these functions leverage the flexibility and speed of pandas, allowing us to efficiently process large amounts of data, and can be found in Code 5.1.

CÓDIGO 5.1. Processing funtions

```
def dupl(file1):
    df = pd.read_csv(file1)
    df = df[df.point == 0]
    df.to_csv(file1,index=False)

def delcol(file1):
    print("start")
    data = pd.read_csv(file1)
    del data['predicted_errors_conversions']
    del data['predicted_errors_reach']
    del data['reach_lower_bound']
    del data['reach_upper_bound']
    del data['actions_lower_bound']
    del data['actions_upper_bound']
    del data['reach']
    del data['bid']
    del data['spend']
    del data['impressions']
    del data['actions']
    data.to_csv(file1,index=False)
```

To create a unified output file containing data from all iterations of our script, we utilize the "concat" functionality provided by pandas. This allows us to concatenate multiple data frames into a single data frame. By combining the outputs obtained from different iterations, we can consolidate the data into a comprehensive file.

The following code snippet (Code 5.2) demonstrates the usage of the "concat" function to create a unified output file:

CÓDIGO 5.2. Concatenation of output files

```
import os
import shutil
import pandas as pd
import glob

lst=[]
src = './outputs'
all_files = glob.glob(os.path.join(src, "output_*.csv"))
for i in all_files:
    lst.append(i)
df_from_each_file = (pd.read_csv(f) for f in lst)
outputs_df   = pd.concat(df_from_each_file, ignore_index=True)
outputs_df.to_csv('output.csv',index=False)
```

By refining and consolidating the output data, we obtained a more streamlined and relevant dataset for further analysis. This improved the efficiency and effectiveness of our research process, allowing us to focus on the insights derived from the collected data.

5.2. Lifetime Performance

In this section, we will evaluate the amount of data extracted, and the extraction rates achieved to assess the overall performance and functionality of the script.

5.2.1. Total Data Gathered Over Time

During the lifetime of the project, an impressive **6.89 million** rows of successful data (We are not counting corrupted data or wrongly extracted) were gathered across three different research projects. We must keep in mind that, during this 18 month, we had to develop, test and build the tools needed, so the extraction was performed over 10 months, resulting on a rate of 700,000 queries each month approximately This achievement was made possible through the development of not only this tool but also an automated system that simulated human behavior on Facebook, allowing for continuous data extraction at unprecedented daily and monthly rates.

The success of this endeavor can be attributed to several factors. Firstly, the automation tool ensured that the accounts performed as humans, avoiding detection by Facebook's algorithms and enabling the extraction process to run smoothly. Secondly, the efficient data extraction mechanism and optimized script design contributed to the high-speed rates achieved.

The significance of this accomplishment lies not only in the sheer volume of data collected, but also in the potential impact it can have. The availability of such a vast dataset opens up new avenues for research and analysis, providing valuable insights into user behavior, advertising trends, and market dynamics. This data can be utilized in various domains, including marketing research, consumer behavior analysis, and strategic decision-making.

To provide a visual representation of the data extraction progress throughout the project, Figure 5.2 displays the amount of data extracted each month. This graph illustrates the project's growth and highlights the periods of intensive data collection.

We can observe, that the total amount of data will add more than the 6.9 million mentioned before, and this is due to a data loss incident we suffered and that will be later related in section 5.2.2.

To assess the speed rate of data extraction, we will examine a graph illustrating the mean speed rate for each day within the mentioned months. This graph provides a clear understanding of the project's efficiency in terms of data extraction speed.

Figure 5.3 presents the daily mean speed rate for each month, showcasing the average amount of data extracted per day. By analyzing this graph, we can identify any variations or patterns in the extraction rate over time.

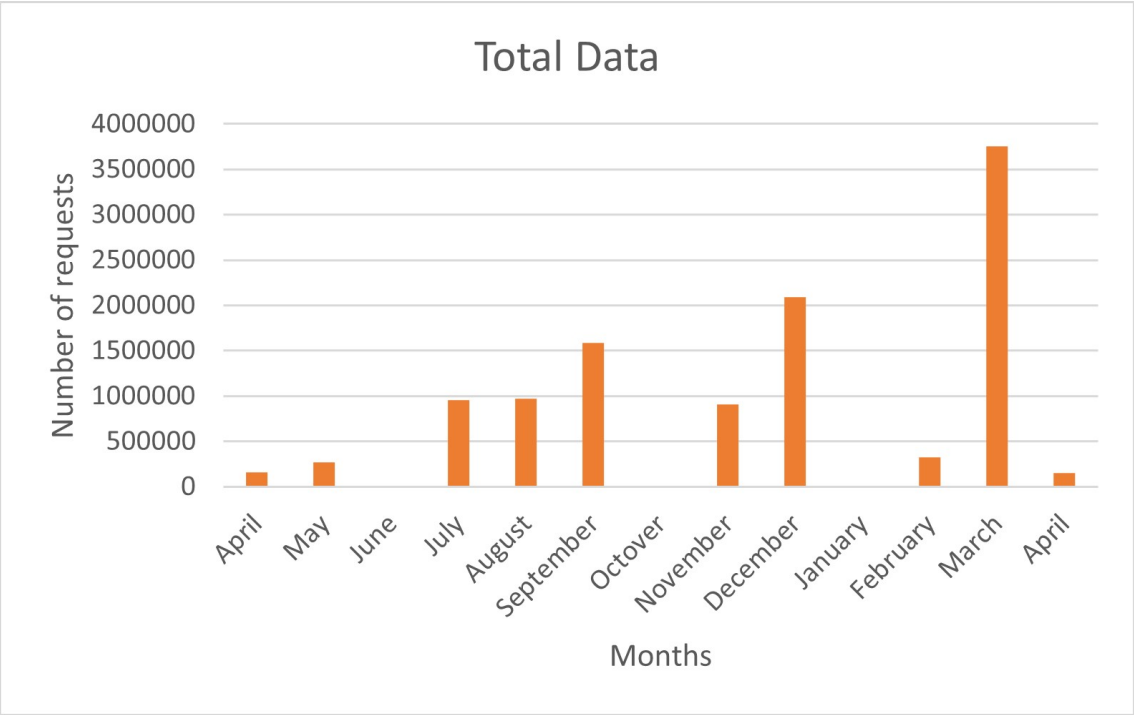


Fig. 5.2. Monthly Data Extraction

The graph allows us to observe the fluctuations in the extraction rate, indicating periods of increased or decreased efficiency. Analyzing these patterns can provide insights into the factors that influenced the extraction speed, such as changes in Facebook's algorithms, network conditions, or adjustments made to the script.

By evaluating the mean speed rate over time, we can gain a comprehensive understanding of the project's performance in terms of data extraction speed and identify potential areas for optimization or further investigation.

5.2.2. Data loss incident

During the months of November and December, a significant data loss incident occurred, resulting in the loss of nearly 3 million extracted data points. This incident was identified after an update to the script, which was intended to implement a new output management fix.

In mid-December, it was discovered that the data extraction process was not properly appending the extracted data to the existing output files. Instead, it was overwriting the files, leading to the immediate shutdown of the system. Recognizing the severity of the situation, the subsequent month was dedicated to addressing the error and attempting to recover as much of the lost data as possible.

Efforts were made to recover the overwritten files, utilizing a backup created by the server and employing various retrieval techniques. As a result, approximately 600,000 data points were successfully recovered. However, regrettably, approximately 2.4 million

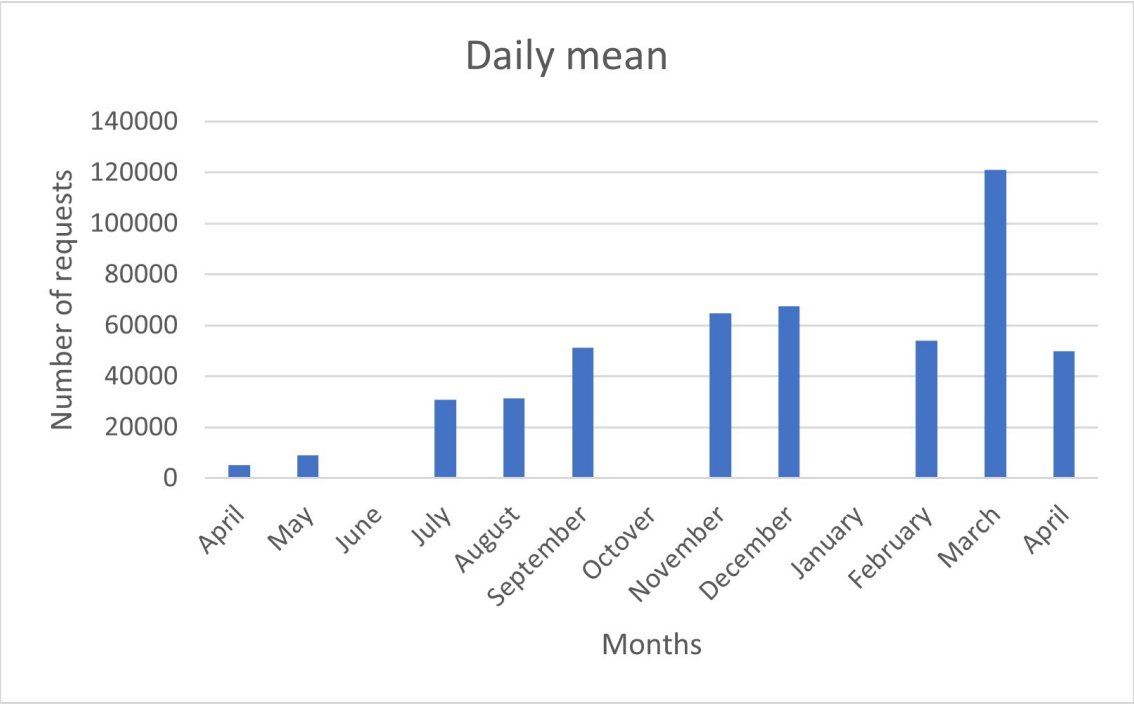


Fig. 5.3. Mean data extracted per Day

data points were permanently lost and needed to be requested again.

This incident highlighted the importance of robust data storage and backup mechanisms, as well as the need for thorough testing and validation of script updates to ensure the integrity of the data extraction process. Measures were subsequently implemented to prevent similar incidents from occurring in the future and to enhance the overall reliability of the system.

6. COSTS

In this chapter, we will perform a comprehensive economic analysis of the automated system, considering a range of cost factors and resource demands. Through a thorough examination of expenses related to human resources, materials, and assets, we aim to gain valuable insights into the financial considerations involved in the implementation of the system.

6.1. Estimation

In this section, we will provide an estimation of the actual expenses incurred in the development of this project. The total estimated cost amounts to €15000. A comprehensive breakdown and analysis of these costs can be found in the subsequent sections.

6.1.1. Human Resources cost

The initial aspect we will examine is the expenditure associated with human resources engaged in the design and implementation of the web scrapper. The table presented below outlines the projected time invested and the respective salaries for the individuals involved.

Employee	**Time Spend (h)**	**Salary**	**Total**
David Rico	1440	11.25€/h	16200€
Ángel Cuevas	150	15€/h	2250€
			18450€

TABLE 6.1. HUMAN RESOURCES EXPENSES

It is important to note that these costs are specific to the development and initial operation phase of the web scrapper and data gather.

6.1.2. Materials and assets

Furthermore, apart from human resources, the financial evaluation encompasses the expenses related to materials and assets essential for the successful implementation and test of the scrapper built. This entails hardware, software licenses, cloud computing services, and other tangible or intangible resources imperative for the system's operation.

Employee	**Cost**	**Depreciation**	**Total**
My personal laptop	900€	16 months	820€
Server	2400€	16 months	2300€
			3120€

TABLE 6.2. MATERIALS EXPENSES

In the context of this specific project, the primary assets utilized were my personal device and the server employed to execute our script.

By taking into account the expenses associated with both human resources and materials/assets, we can obtain a holistic assessment of the financial implications involved in the implementation and usage of the scrapper over the lifetime of the project.

7. PLANNING

To visually depict the different project tasks and their respective durations, a detailed Gantt diagram has been generated (see Figure 7.1). The project officially commenced on June 1st and reached the conclusion of its testing and evaluation phase in December of the following year.

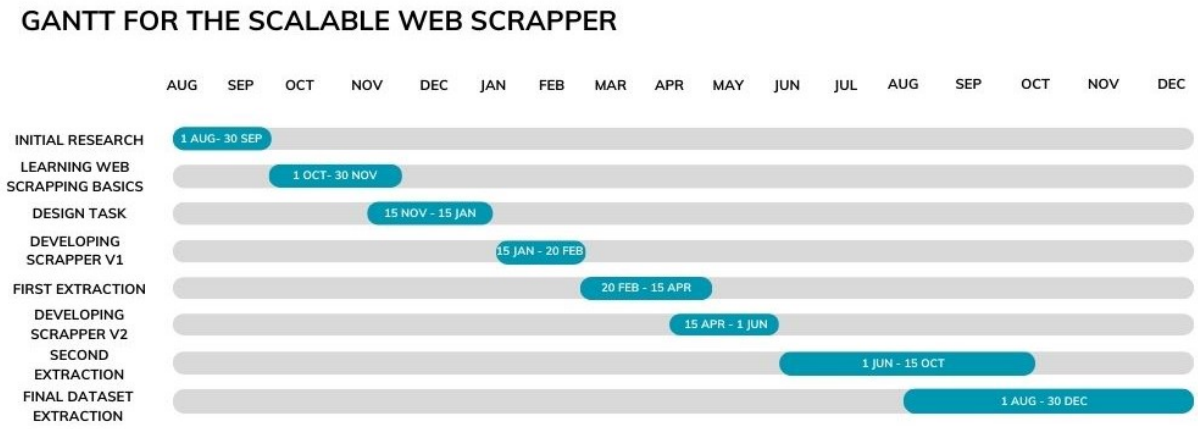


Fig. 7.1. Gantt diagram with the phases of the project

The provided timeline aptly captures the development process undertaken during years 2020 to end of 2021. It is important to note that the mentioned dates are approximations, as the original plan underwent adjustments and refinements to accommodate the specific demands and complexities encountered throughout each phase of the project.

8. SOCIO-ECONOMIC ANALYSIS

Web scraping tools have emerged as powerful tools that have revolutionized socio-economic research. By providing access to large-scale aggregated data in short time, we empower this revolution. This section examines the impact of our tool and web scraping on research and knowledge advancement, and their role in accelerating data-driven decision-making, and its contribution to bridging the data gap.

8.1. Impact on Research and Knowledge Advancement

The availability of large-scale aggregated data through web scraping has transformed socio-economic research across various disciplines. Researchers in economics, sociology, marketing, and public policy have leveraged these vast datasets (Big Data) to address complex research questions and gain valuable insights into socio-economic phenomena.[22] By analyzing comprehensive datasets obtained through web scraping, researchers have been able to uncover hidden patterns, identify trends, and enhance their understanding of societal dynamics. This wealth of data has not only expanded the scope of research but has also contributed to the advancement of knowledge in diverse fields.

Web scraping has enabled researchers to access real-time and up-to-date information, providing a more accurate representation of socio-economic trends. By utilizing aggregated data from Facebook through our tool, researchers can capture a holistic view of social and economic behaviors, facilitating the identification of patterns and correlations that were previously unattainable. The ability to analyze large-scale datasets has paved the way for innovative research methodologies and has significantly enriched the empirical evidence base.[23]

In line with the open-source nature of our project, we are committed to pushing the boundaries of our tool's development and expanding its capabilities to encompass other social network platforms and databases. Our aim is to provide researchers worldwide, regardless of their economic background or technical capabilities, with the means to access and utilize these valuable resources.

By fostering a collaborative and inclusive environment, we encourage contributions from a diverse range of developers, researchers, and users. This collective effort will drive the continuous improvement and evolution of our tool, ensuring its adaptability to emerging technologies and data sources.

Our vision extends beyond the current scope of the project, as we recognize the ever-growing importance of social networks and databases in socio-economic research. We aspire to empower researchers with the tools and resources they need to conduct in-depth analyses and generate valuable insights on a global scale

8.2. Accelerating Data-Driven Decision-Making

With our script, we can also facilitate the access to big-data to anyone, and this could facilitate data-driven decision-making across various sectors. Businesses, governments, and organizations can leverage this wealth of information to make informed choices, develop effective strategies, and enhance socio-economic outcomes. For example, policy-makers can utilize web-scraped data to gain insights into consumer behavior, market trends, and public sentiment, enabling evidence-based policy formulation.[24]

In the realm of market research, web scraping has already revolutionized the collection of competitive intelligence and market insights. By extracting data from online sources, businesses can analyze market trends, monitor competitor activities, and identify emerging opportunities. An example of this is the analysis of Rental housing trends performed on 2016.[25]. This data-driven approach empowers organizations to make timely and informed decisions, leading to improved competitiveness and market positioning.

Furthermore, web scraping and big data has played a crucial role in resource allocation and optimization. By extracting relevant data from various sources, organizations can gain a comprehensive understanding of supply and demand dynamics, optimize production processes, and allocate resources effectively. This data-driven approach enhances efficiency, reduces costs, and promotes sustainable socio-economic development.

8.3. Bridging the Data Gap

One of the significant advantages of web scraping is its ability to bridge the data gap by providing access to valuable information that was previously inaccessible or limited in scope. Traditional data collection methods often suffer from limitations such as small sample sizes, outdated data, or high costs. Web scraping has democratized data access, enabling researchers from diverse backgrounds to analyze comprehensive datasets and gain a more accurate understanding of socio-economic trends and patterns.

Our script takes data scraping to the next level, surpassing the already beneficial aspects of this technique. With our tool, accessing data is not only faster and more efficient, but it also boasts an unprecedented throughput, enabling us to gather a wealth of information. The days of having to compromise between precision and size when collecting data could be forgotten. We also tackle the concerns about obtaining outdated information in rapidly changing and volatile markets.

Web scraping allows researchers to overcome geographical and temporal constraints by accessing data from various online platforms. By allowing them to increase their datasets, we also increase the reach of their investigations as well. This enables the analysis of real-world data without borders, capturing the complexities of socio-economic interactions in different contexts. By harnessing the power of web scraping, researchers can obtain valuable insights into many fields that were previously unattainable.[26]

Understanding cultural and social differences across the world is essential for the development of our society. By enabling the study and analysis of these differences through our tool, we can elevate awareness to unprecedented levels. This heightened awareness fosters a deeper understanding and appreciation of diverse cultures, leading to more inclusive and harmonious societies.[27]

The accessibility of comprehensive large datasets obtained through web scraping has empowered researchers to conduct robust empirical analyses and draw meaningful conclusions, This has led to a more nuanced understanding of socio-economic trends.

9. FUTURE WORK

In this chapter, we address the future work required to further enhance the usability, performance, and compatibility of our web crawler. While significant progress has been made in achieving our individual objectives, there are still areas that require attention to keep the crawler up-to-date with the latest technologies and ensure optimal functionality. The following sections outline specific areas where improvements can be made to elevate the capabilities of our crawler.

9.1. Finish score implementation and enhanced account management

As discussed in Section 4.5.4, one of the ongoing initiatives was the development of a scoring system for our accounts. However, due to time constraints, the implementation of this feature remains unfinished. In this section, we propose completing the score implementation and enhancing the account management module to leverage the scores effectively.

The first step would involve revisiting the parameters used in the scoring algorithm to ensure their continued relevance and effectiveness. This would require conducting an analysis of Facebook Ads Manager[28] and API regarding web scraping, considering factors such as new website policies, most recent anti-scraping measures, and other variables that impact the performance and availability of accounts.

Once the scoring system is in place, we can make intelligent and informed decisions regarding account rotation. Instead of randomly rotating accounts, we can prioritize those with higher scores, indicating better performance and reliability. This intelligent account management approach would optimize the usage of accounts and improve crawling efficiency. It will also let the worse score accounts more time into idle state, and this could potentially increase their score on future executions.

Furthermore, we can incorporate workload management into the account rotation strategy. By monitoring and tracking the workload of each account, we can dynamically adjust the rotation schedule. For example, accounts that have been idle for an extended period can be given priority in the rotation, allowing them to contribute while accounts with higher recent workloads are given sufficient rest. This workload-based rotation strategy ensures a fair distribution of tasks and maximizes their individual performance.

By completing the score implementation and enhancing account management with intelligent rotation and workload considerations, we can further optimize the efficiency and effectiveness of our web crawler. These enhancements will contribute to improved performance, better utilization of resources, and ultimately enhance the overall crawling capabilities of our system.

9.2. Advanced data storage management

The current approach of storing scraped data in individual raw text files presents limitations in terms of efficiency and accessibility. To address this issue, we propose implementing more sophisticated data storage and management solutions. While creating a full-fledged database might be overly complex for our specific needs, we can explore alternative approaches that offer improved storage efficiency and easier data retrieval.

One option is to leverage existing database technologies, such as relational databases or NoSQL databases, to store and organize the scraped data. Databases provide efficient indexing and querying capabilities, allowing for faster and more targeted data retrieval. Additionally, they offer built-in mechanisms for handling large volumes of data and can support concurrent access by multiple users. By adopting a database solution, we can eliminate the need for manual post-processing and enable real-time access to the extracted data.

Another consideration is the implementation of a distributed file system. Distributed file systems, such as Hadoop Distributed File System (HDFS)[29] or Apache Cassandra[30], are designed to handle large-scale data storage and processing across multiple machines. By distributing the data across a cluster of nodes, these systems provide fault tolerance, scalability, and efficient data management. Adopting a distributed file system would enable us to handle the increasing volume of scraped data while ensuring high availability and improved performance.

In addition to exploring alternative storage solutions, we can also automate the post-processing of the extracted data. Rather than waiting until the entire extraction process is complete, we can implement parallel processing techniques that allow for simultaneous extraction and post-processing. This would significantly reduce the time required to access the data and make it available for analysis. By automating the post-processing step, researchers can access and analyze the data as soon as it is extracted, enhancing the efficiency of their workflow.

Overall, the focus of advanced data storage management is to improve storage efficiency, accessibility, and data processing capabilities. By adopting appropriate storage technologies and automating post-processing, we can enhance the effectiveness of our web crawler and facilitate seamless data retrieval and analysis.

9.3. Integration with Data Analysis Tools

While our web scraper primarily focuses on gathering raw data for further investigation, integrating data analysis tools can provide additional value to researchers. By incorporating simple data analysis functionalities within our web crawler, we can empower researchers to gain insights from the extracted data in real-time.

One possible enhancement is to implement basic data analysis features using popular data manipulation libraries such as NumPy[31] or pandas[32]. These libraries offer a wide range of functions and tools for data analysis, including calculations of means, ranges, statistical measures, and the generation of visualizations and graphs. By integrating these capabilities into our web scraper, researchers can gain immediate insights and preliminary findings from the data as it is being extracted.

Furthermore, the integration of data analysis tools can facilitate data validation and quality control. By implementing data consistency checks and applying predefined rules or constraints to the extracted data, researchers can identify and address data anomalies or errors in real-time. This feature ensures the integrity and reliability of the collected data, providing researchers with more accurate results and reducing potential errors in their analyses.

By offering basic data analysis functionalities within our web crawler, we enable researchers to obtain actionable insights from the extracted data while the crawling process is ongoing. This integration enhances the usability and value of our web scraper as a comprehensive research tool.

9.4. User Interface and Visualization

To enhance the accessibility and usability of our web scraper, we recognize the importance of developing a clear and intuitive user interface (UI) that provides users with a seamless experience. A well-designed UI enables users to interact with the tool effortlessly and effectively, improving their overall satisfaction and productivity.

One aspect of the UI enhancement is to provide users with a visual representation of the ongoing procedures. While we already have the "tqdm" library[33] implemented to display the extraction status, we can further enhance the visualization by incorporating additional elements. For instance, having a dedicated dashboard that shows the status of each account in real-time, including whether they are currently working, active, or idle, would provide users with a comprehensive overview of the system's progress. This real-time visualization allows users to monitor the performance of individual accounts and make informed decisions based on their current status.

Another valuable addition to the UI would be a stop and resume option. This feature allows users to pause the extraction process temporarily and resume it at a later time without losing progress. The stop and resume functionality provides flexibility and convenience, particularly in scenarios where users may need to interrupt the extraction process due to external factors or other priorities. Implementing this option empowers users to have better control over the extraction process and optimizes their workflow.

In addition to the UI improvements, we should also consider reorganizing the input file system to make it more user-friendly. This could involve implementing a structured folder hierarchy or adopting standardized naming conventions for input files. By organizing the

input files systematically, users can easily locate and manage the necessary files without confusion or manual effort. This enhancement simplifies the setup process and reduces the chances of errors or omissions in file selection.

Furthermore, we can explore the integration of data visualization capabilities within the user interface. This feature would enable users to visualize and analyze the extracted data directly within the tool, eliminating the need to export the data to external visualization software. By incorporating data visualization functionalities, such as interactive charts, graphs, or maps, users can gain deeper insights into the extracted data and perform exploratory analysis efficiently.

Overall, focusing on the user interface and visualization aspects of our web scraper contributes to its accessibility and usability. By providing a clear and intuitive UI, real-time status visualization, stop and resume functionality, optimized file organization, and integrated data visualization, we create a tool that is more user-friendly and capable of meeting the diverse needs of researchers and users.

10. CONCLUSION

The primary objective of this thesis was successfully achieved, which was to develop an autonomous web scraper capable of efficiently obtaining large-scale datasets of aggregated data in a significantly reduced timeframe. After a year and a half of dedicated work, development, and rigorous testing in real-world scenarios and research projects, we have been able to gather over 9 million rows of data.

Throughout the course of this project, we encountered several challenges and obstacles. We had to overcome various anti-scraping mechanisms implemented by Facebook, address the loss of nearly 3 million rows of data resulting from a bug after a script update, and continuously manage token refreshment. These challenges tested our resolve and problem-solving skills, but ultimately contributed to the improvement and refinement of our web scraping tool.

It is important to recognize that this tool can serve as a foundation for building more powerful and advanced scraping tools in the future. By incorporating enhanced account management techniques, expanding the scope of account groups, and implementing insightful analysis tools, researchers can benefit from a robust infrastructure that provides valuable insights and facilitates their investigations.

As technology continues to evolve and the trend of data scraping grows, we firmly believe that our tool has the potential to evolve and adapt accordingly. By utilizing this tool, we have successfully achieved the main objective of providing a scalable and multidisciplinary solution that empowers researchers to enhance the quality and reach of their investigations. This has the potential to create a positive impact not only within the academic community but also in society as a whole.

Overall, this thesis has contributed to advancing the field of web scraping and has demonstrated the potential of automated data extraction in enabling groundbreaking research. The journey has been challenging, but the results obtained and the future possibilities that lie ahead make it a worthwhile endeavor. With further improvements and advancements in this technology, we are confident that web scraping will continue to revolutionize research practices and contribute to positive societal change.

BIBLIOGRAPHY

[1] N. Obradovich *et al.*, *Expanding the measurement of culture with a sample of two billion humans*, 2020. DOI: 10.3386/w27827.

[2] F. Liberini, M. Redoano, A. Russo, Á. Cuevas, and R. Cuevas, "Politics in the facebook era - evidence from the 2016 us presidential elections," *SSRN Electronic Journal*, 2020. DOI: 10.2139/ssrn.3584086.

[3] A. Mehrjoo, R. Cuevas, and Á. Cuevas, "A new methodology to measure faultlines at scale leveraging digital traces," *EPJ Data Science*, vol. 11, no. 1, 2022. DOI: 10.1140/epjds/s13688-022-00350-w.

[4] Á. Cuevas, R. Cuevas, K. Desmet, and I. Ortuño-Ortín, *The gender gap in preferences: Evidence from 45,397 facebook interests*, 2021. DOI: 10.3386/w29451.

[5] Statista. "Facebook - statistics facts." (2023), [Online]. Available: https://www.statista.com/topics/751/facebook/ (visited on 02/07/2023).

[6] J. Kauflin. "Facebook's biggest acquisitions." (2017), [Online]. Available: https://www.forbes.com/sites/jeffkauflin/2017/12/15/facebooks-biggest-acquisitions/?sh=5a9c1f0b6f63 (visited on 02/07/2023).

[7] B. News. "Facebook changes company name to meta." (2021), [Online]. Available: https://www.bbc.com/news/technology-59102131 (visited on 02/07/2023).

[8] Scrapy. "Scrapy | a fast and powerful scraping and web crawling framework." (), [Online]. Available: https://scrapy.org/ (visited on 02/07/2023).

[9] Octoparse. "Web scraping limitations | octoparse." (), [Online]. Available: https://www.octoparse.com/blog/web-scraping-limitations (visited on 02/07/2023).

[10] ParseHub. "Parsehub | free web scraping - the most powerful web scraper." (), [Online]. Available: https://www.parsehub.com/ (visited on 02/07/2023).

[11] Apify. "Apify: The web scraping and automation platform." (), [Online]. Available: https://apify.com/ (visited on 02/07/2023).

[12] M. Khder, "Web scraping or web crawling: State of art, techniques, approaches and application," *International Journal of Advances in Soft Computing and its Applications*, vol. 13, no. 3, pp. 145–168, 2021. DOI: 10.15849/ijasca.211128.11.

[13] R. Mitchell, *Web scraping with python: Collecting more data from the modern web*. O'Reilly Media, Inc., 2018.

[14] K. D. Gorro, M. J. Sabellano, K. Gorro, C. Maderazo, and K. Capao, "Classification of cyberbullying in facebook using selenium and svm," *2018 3rd International Conference on Computer and Communication Systems (ICCCS)*, 2018. DOI: 10.1109/ccoms.2018.8463326.

[15] N. A. Utami, W. Maharani, and I. Atastina, "Personality classification of facebook users according to big five personality using svm (support vector machine) method," *Procedia Computer Science*, vol. 179, pp. 177–184, 2021. DOI: 10.1016/j.procs.2020.12.023.

[16] M. Mancosu and F. Vegetti, "What you can scrape and what is right to scrape: A proposal for a tool to collect public facebook data," *Social Media + Society*, vol. 6, no. 3, p. 205 630 512 094 070, 2020. DOI: 10.1177/2056305120940703.

[17] F. Caravaca, J. González-Cabañas, Á. Cuevas, and R. Cuevas, "Estimating ideology and polarization in european countries using facebook data," *EPJ Data Science*, vol. 11, no. 1, 2022. DOI: 10.1140/epjds/s13688-022-00367-1.

[18] *Rfc 2616*. [Online]. Available: https://www.w3.org/Protocols/rfc2616/rfc2616.html.

[19] [Online]. Available: https://www.facebook.com/business/help/975570072950669?id=434838534925385.

[20] C. Kozierok, *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*. No Starch Press, 2005.

[21] "Facebook developers," Facebook. (), [Online]. Available: https://developers.facebook.com/.

[22] *Big Data, Big Analytics*, pp. 1–18, 2013. DOI: 10.1002/9781118562260.ch1.

[23] C. K. Leung, F. Jiang, T. W. Poon, and P.-É. Crevier, "Big data analytics of social network data: Who cares most about you on facebook?" *Studies in Big Data*, pp. 1–15, 2017. DOI: 10.1007/978-3-319-60255-4_1.

[24] F. Provost and T. Fawcett, "Data science and its relationship to big data and data-driven decision making," *Big Data*, vol. 1, no. 1, pp. 51–59, 2013. DOI: 10.1089/big.2013.1508.

[25] G. Boeing and P. Waddell, "New insights into rental housing markets across the united states: Web scraping and analyzing craigslist rental listings," *Journal of Planning Education and Research*, vol. 37, no. 4, pp. 457–476, 2016. DOI: 10.1177/0739456x16664789.

[26] S. Giest and A. Samuels, "'for good measure': Data gaps in a big data world," *Policy Sciences*, vol. 53, no. 3, pp. 559–569, 2020. DOI: 10.1007/s11077-020-09384-1.

[27] A. Gal and A. Senderovich, "Process minding: Closing the big data gap," *Lecture Notes in Computer Science*, pp. 3–16, 2020. DOI: 10.1007/978-3-030-58666-9_1.

[28] F. Meta, *Measuring using the ads manager documentation*. [Online]. Available: https://developers.facebook.com/docs/app-ads/measuring/ads-manager/.

[29] Oscarfmdc, *¿qué es hdfs? introducción 2023*, Dec. 2022. [Online]. Available: https://aprenderbigdata.com/hdfs/.

[30] *Apache cassandra manual page*. [Online]. Available: https://cassandra.apache.org/_/cassandra-basics.html.

[31] *Numpy documentation*. [Online]. Available: https://numpy.org/doc/stable/.

[32] *Pandas python library documentation*. [Online]. Available: https://pandas.pydata.org/docs/.

[33] *Tqdm project, usage and documentation*. [Online]. Available: https://pypi.org/project/tqdm/.