## Introduction

The main goal of the project is to create a monitoring tool to gather information about Facebook posts in certain groups. We remarked that this kind of tool already exists for other social medias, but we did not find anything for Facebook, so we decided to make our own. The final objective of this type of tool is to analyze trends and posts’ performances and make content creation easier. The information that we obtained in this project were:

* Content of the post (Text, images, links)
* Performance indicators (Reactions, number of comments, shares)
* Date
* Author
* Link of the post

The goal of this project is not to obtain a particular dataset for later use but rather to create a method that can be used by others to get their own dataset. The dataset that we created is mostly composed of non-related posts about cats and dogs. The list of groups we used for our testing can be found in the groups.json file.

## Scrapping Method

Since Facebook has a lot of defense mechanisms against scrapping, we couldn’t use the usual methods like Scrappy. The feed of a Facebook page is almost empty when the user doesn't scroll to load the posts. A library that can interact with the browser is more suited than a simple web scrapper in order to interact with social media pages. We also thought about using the Facebook API, but it only allows people to interact with their own account to publish posts and cannot be used to gather information about posts in specific groups as we wanted. In sum, we decided to use a browser simulator library like Puppeteer, Beautiful Soup, or Selenium. We ended up choosing Playwright as it seemed simpler and quite suited to our needs.

Due to the nature of our project, the data is meant to be updated every day or more and the user can indicate which group they want to scrape using a json file.

## Encountered problems

The first problems that we have encountered were the cookie page that we managed to bypass by a simple click on "accept only essential cookies". We were faced with a connection page next that we first ignored since it is not mandatory to be connected to look at public groups’ content. However, after a few tests, Facebook asked us to connect to see these groups, likely because we were detected as bots. We then created tests accounts to log in and we changed our strategy of directly going into groups to scrape them and we logged in using the Facebook.com main page.

Our first account was quickly detected as a bot and was restricted, preventing us to connect and continue our scrapping. When creating a new one, Facebook was even able to prevent us from doing so because we were using the same password and connection. But after choosing a new password and using a VPN to create the account, we were able to pursue. The objective was then to not be detected again so we slowed down our clicking speed as well as our typing speed to enter the login information. We also used random delays to make it harder for Facebook to detect us.

Another problem that we encountered was that the names of the classes used in the HTML page were always changing, so we couldn’t use them as a reference for scrapping. To solve this, we had to reference the tree structure of the feed to scrape each post. This was very unpractical because each time we encountered a new type of post it would change the structure and we would need to differentiate the cases. For this reason, and by lack of time, we chose not to scrape each type of post. Our scrapper supports the scrapping of text, images, links, and the special posts with text on a colored background. Hence, we ignored Reals (short videos), videos, polls, quotes, posts that are automatically translated and probably other possibilities that we did not encounter. This still allows us to scrape more than 95% of posts successfully. When encountering a problematic post our scrapper is able to get all the available information and ignore problematic ones indicating “#Error, couldn’t scrape” in the fields that it was not able to gather.

## Data cleaning

Our method allowed us to choose how and what to extract. The cleaning is by construction already taken care of. A lack of information being represented by an empty cell in the created dataset. Given our goal to gather information and offer a catalogue of Facebook posts, we do not need to take care of null values.

To give an example of how the scrapping itself take care of the cleaning of the data, we were faced with duplicate inside posts' text when the "See more" button needed to be clicked. Indeed, since the tree structure was deeper, using the text\_content method would return the text of the current node as well as its children’s, duplicating it. The function we used to get the text (get\_content) requires to be precise regarding the HTML tag targeted, and we need to adapt our function to each specific case Facebook posts can have.

The rest of the data transformation we did was rather simple such as transforming KPI counts into numbers (1.2K needs to be translated to 1200), transforming the date into a datetime object (taking care of all date formats possible or separating the author’s name from the added data for example feelings or place.

## Database

To store our data, we chose an SQL database because it was the best suited for our needs:

* It allows us to store efficiently relations between posts, links and images that may be many to many relationships.
* It can efficiently edit existing data using an index on the posts’ links to update KPI’s values of an already encountered post.

Keeping our data inside text file is not adaptable enough and would be slow when needing to update KPIs, our data being structured also limits our interest in NO SQL databases.

The structure of our SQL database is as follows:

Une image contenant texte, Logiciel multimédia, logiciel, Logiciel de graphisme

Description générée automatiquement

## Conclusion

To conclude, this project was very interesting thanks to a relevant choice of subject. It allowed us to work with a large quantity and diversity of data. The data could be used in different applications such as training a language model, training image generation models or even just for monitoring purposes and learn about trends. We were able to overcome most of the issues we faced but we would have wanted to be able to scrape more kinds of posts even if we can cover about 95% of existing posts. Our method could also be used to expand the scrapping to Facebook pages or even the Facebook feed later instead of just groups since the structure is similar.