Social Media Data Science Pipelines Project 1: Project Proposal

Devang Jagdale
djagdale@binghamton.edu
Binghamton University
Binghamton, New York, USA

Tejas Hiremath
thiremath@binghamton.edu
Binghamton University
Binghamton, New York, USA

Chaitanya Jha
cjha@binghamton.edu
Binghamton University
Binghamton, New York, USA

1 Introduction

In this project, we plan to build a continuous data collection system that gathers data from two data sources: Reddit and 4chan. The collected data will serve as the foundation for subsequent data science projects, focusing on sentiment analysis and the detection of harmful content, such as hate speech. The system will fetch data dynamically from selected subreddits and 4chan boards, process the data to remove irrelevant content, and store it in a PostgreSQL database for further analysis.

2 Data Sources

2.1 Reddit API

Reddit's API provides programmatic access to Reddit's extensive repository of user-generated content. Given the high volume of comments and posts, our crawler will target specific subreddits rather than attempting to collect all data from the platform.

Search API: https://oauth.reddit.com/r/{subreddit}/search retrieves posts, sorted by newest.

Post Details API: https://oauth.reddit.com/r/{subreddit}/comments/{post_id} retrieves full post content by post ID.

API Endpoints:

- /r/{subreddit}/new: Fetches new posts from a given subreddit.
- /r/{subreddit}/comments: Collects new comments under each post.

The set of subreddits will be configurable, allowing us to easily add or remove subreddits dynamically.

2.2 4chan API

The 4chan API provides access to various boards where users post anonymously. We will use this API to fetch threads and their corresponding posts in real-time from targeted boards.

Boards API: The ChanClient can be extended to call the 4chan Boards API to dynamically retrieve a list of all boards. For this project, we assume that /pol/ is statically defined as the target board.

Example Call: http://a.4cdn.org/boards.json

Catalog API: This API is used to collect metadata from all active threads on /pol/. The data includes thread numbers, post counts, and more.

Example Call: http://a.4cdn.org/pol/catalog.json

Thread API: This API is used to collect the content of individual threads by thread number, providing access to all posts within a specific thread.

Example Call: http://a.4cdn.org/pol/thread/{thread_number}.json

API Endpoints:

- /boards.json: Lists available boards.
- /pol/catalog.json: Fetches threads from a specific board.

3 Data Collection Workflow

3.1 Client Class Initialization

We create a ChanClient and RedditClient class to interact with the 4chan and Reddit APIs to fetch data.

Methods: `get_thread()`, `get_catalog()`, `get_latest_posts()`, `get_post_details()`.

3.2 API Calls for Data Retrieval

The following API endpoints are used to collect data:

Catalog API: http://a.4cdn.org/pol/catalog.json retrieves the catalog of threads from a specified board. This is implemented in the `get_catalog` method of the ChanClient class.

Thread API: http://a.4cdn.org/pol/thread/{thread_number}.json retrieves the full contents of a thread using its thread number.

Search API: https://oauth.reddit.com/r/{subreddit}/search retrieves posts based on search query.

Post Details API: https://oauth.reddit.com/r/{subreddit}/comments/{post_id} retrieves full post content.

3.3 Post/Thread Collection Process

(1) **Retrieve Latest Posts:** Use `get_latest_posts` to fetch active posts.
(2) **Detect Inactive Posts:** Compare current posts with previously fetched posts to identify inactive ones.
(3) **Process Inactive Posts:** Use `get_post_details` to retrieve details of inactive posts.

The `crawl_catalog` function handles the data collection process by interacting with the ChanClient. The process can be broken down into the following steps:

(1) **Retrieve Current Catalog:** The `ChanClient.get_catalog(board)` method fetches the current list of threads for the board. The `thread_numbers_from_catalog(catalog)` function extracts the thread numbers from the catalog JSON response.
(2) **Dead Thread Detection:** Using the `find_dead_threads` function, the system compares the thread numbers from the previous catalog crawl with the current catalog crawl.
(3) **Processing Dead Threads:** For each detected dead thread, a new task (crawl job) is enqueued for processing, which could involve retrieving the archived thread data for analysis.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3.4 Logging and Error Handling

A logger is used to log important events, such as the status of HTTP requests and the identification of dead threads. This helps track the system's progress and any potential issues during API interactions.

3.5 Scheduled Crawling

After each crawl, the system is designed to enqueue a new catalog crawling job to ensure continuous data collection.

4 System Architecture

The proposed system architecture consists of five key components:

4.1 Scheduler

The scheduler will trigger data collection at regular intervals, ensuring continuous data collection from both Reddit and 4chan.

4.2 Crawler

The crawler will handle API calls to Reddit and 4chan, fetching data as it becomes available. For Reddit, the crawler will collect new posts and comments from specified subreddits, while for 4chan, it will focus on specific boards. The subreddit and board lists will be configurable.

4.3 Data Preprocessing

After fetching the data, it will be cleaned and preprocessed. This involves filtering out irrelevant content (e.g., removing nontext data), parsing the fetched data, and integrating it with an external Hate Speech API to flag potential harmful content.

4.4 Database (PostgreSQL)

The preprocessed data will be stored in a PostgreSQL database. This structure ensures that the data is well organized and accessible for future data analysis and modeling projects.

4.5 Data Analytics

In future stages, this component will analyze the collected data. Potential analyses include sentiment analysis, frequency of hate speech, user engagement trends, and the creation of various types of charts to visualize the data, including bar charts, line graphs, and histograms. We will leverage the political data collected to create graphs that depict trends in political discussions over time, engagement metrics, and sentiment distribution.

Additionally, we will focus on time series data analysis to observe user engagement patterns over time. This will allow us to identify peak activity periods related to specific political events or discussions. By analyzing the frequency of posts and replies, we can infer moments of heightened interest and engagement within the community. Furthermore, we will calculate and visualize hate speech scores over time, examining correlations between political discussions and the prevalence of harmful language. This comprehensive approach will provide insights into the dynamics of online political discourse and the role of user engagement in shaping these conversations.

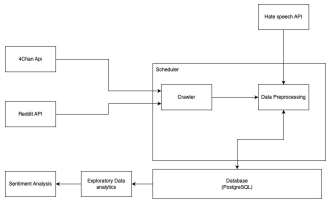

Figure 1: Enter Caption

5 Libraries and Tools

The project will utilize several key libraries to support data collection, processing, and storage:

- **Programming Language:** Python.
- **Requests:** For making HTTP requests to APIs.
- **Postman:** For testing API.
- **py-postgresql:** For interacting with the PostgreSQL database.

6 4Chan napkin math

6.1 Data Collection Process

1. **Identifying Data Sources:** The data is collected from specific boards on 4chan, focusing on politically incorrect discussions.

2. **Counting Thread IDs:** We count the number of thread IDs from the start of the day to the end of the day for the political incorrect categories of 4chan boards. - For September 17, around 2,500 threads were counted. - For September 18, around 2,700 threads were counted.

3. **Estimating Future Data:** Given the upcoming political season, it is predicted that the number of threads will increase. Therefore, we are taking an average of 3,000 threads per day for future estimates.

4. **Average Replies:** On average, each thread receives approximately 100 replies.

6.2 Napkin Math for Storage Requirements

1. **Estimate Size per Thread:** Using the earlier estimate, the average size of a thread with 100 replies is approximately 47,968 bytes (or about 47.97 KB).

2. **Daily Data Size:**

- **September 17 Estimate:** - Threads: 2,500 - Daily Size:

$2,500 \text{ threads} \times 47,968 \text{ bytes/thread} \approx 119,920,000 \text{ bytes} \approx 114.19 \text{ MB}$

- **September 18 Estimate:** - Threads: 2,700 - Daily Size:

$2,700 \text{ threads} \times 47,968 \text{ bytes/thread} \approx 129,505,600 \text{ bytes} \approx 123.23 \text{ MB}$

- **Average Daily Size (for future estimates):** - Threads: 3,000 - Daily Size:

$3,000 \text{ threads} \times 47,968 \text{ bytes/thread} \approx 143,904,000 \text{ bytes} \approx 137.52 \text{ MB}$

3. **Weekly Data Size:** Assuming an average of 3,000 threads per day for the next week:

- Calculate Daily Size:

$3,000 \text{ threads/day} \times 47,968 \text{ bytes/thread} = 143,904,000 \text{ bytes/day}$

- Calculate Weekly Size:

$$143,904,000 \text{ bytes/day} \times 7 \text{ days} = 1,007,328,000 \text{ bytes}$$

- Convert Bytes to Megabytes:

$$1,007,328,000 \text{ bytes} \div (1024^2) \approx 960.12 \text{ MB}$$

4. **Monthly Data Size:** Assuming 30 days in a month and maintaining the same average of 3,000 threads per day:

- Calculate Daily Size:

$3,000 \text{ threads/day} \times 47,968 \text{ bytes/thread} = 143,904,000 \text{ bytes/day}$

- Calculate Monthly Size:

$$143,904,000 \text{ bytes/day} \times 30 \text{ days} = 4,317,120,000 \text{ bytes}$$

- Convert Bytes to Gigabytes:

$$4,317,120,000 \text{ bytes} \div (1024^3) \approx 4.02 \text{ GB}$$

5. **Summary of Storage Estimates:** - Estimated Daily Size: - September 17: 114.19 MB - September 18: 123.23 MB - Average for future estimates: 137.52 MB - Estimated Weekly Size: Approximately 960.12 MB - Estimated Monthly Size: Approximately 4.02 GB

7 Reddit napkin math

7.1 Reddit Data Estimates

We have considered a 25% growth rate per day as the U.S. elections are approaching in two months.

After using the Reddit API https://oauth.reddit.com/r/politics, we observed the number of posts per day to be 400 posts.

Daily exponential growth rate: 25% = 0.25.

(1) **Estimating the number of posts for one day after exponential growth:**

$$400 \times (1 + 0.25)^1 = 400 \times 1.25 = 500 \text{ posts/day}.$$

(2) **Estimating the number of posts for a week after exponential growth:**

$$400 \times (1 + 0.25)^7 \approx 2,500 \text{ posts/week}.$$

(3) **Estimating the number of posts for a month after exponential growth:**

$$400 \times (1 + 0.25)^{30} \approx 130,000 \text{ posts/month}.$$

Summary of estimates with a 25% growth rate:

- Posts per day after 1 day: ~ 500 posts/day.
- Posts per week after 7 days: ~ 2,500 posts/week.
- Posts per month after 30 days: ~ 130,000 posts/month.

This shows the significantly larger impact of a 25% daily growth rate as the elections approach. The growth rate causes the number of posts to increase rapidly, especially over a longer period like a month.

8 Conclusion

This project aims to create a scalable and efficient data collection pipeline that can adapt to the evolving needs of research in social media analytics. By focusing on politically charged discussions, we hope to provide valuable insights that can inform further analyses and contribute to the broader field of data science.