Meme Trend Analysis System: Implementation Report

Amruta Chaudhari
State University of New York, Binghamton
Binghamton, New York, USA
achaudhari@binghamton.edu

Priyanka Nandwani
State University of New York, Binghamton
Binghamton, New York, USA
pnandwani@binghamton.edu

Narendra Khatpe
State University of New York, Binghamton
Binghamton, New York, USA
nkhatpe@binghamton.edu

Abstract

This report details the implementation of a continuous data collection system designed to harvest data from Reddit and 4chan. The system employs APIs provided by both platforms to collect posts and comments in real-time, storing them in a MongoDB database. We outline the system architecture, discuss the implementation specifics, highlight challenges faced during development, and provide preliminary data exploration.

CCS Concepts

• **Information systems → Data management systems**; • **Computing methodologies → Distributed computing methodologies**.

Keywords

Data Collection, Reddit API, 4chan Scraper, Distributed Systems, MongoDB, Faktory, Python

ACM Reference Format:
Amruta Chaudhari, Priyanka Nandwani, and Narendra Khatpe. 2024. Meme Trend Analysis System: Implementation Report. In . ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Data collection is a critical component of any data science pipeline. Without robust and efficient data gathering mechanisms, subsequent analysis and insights become impossible. This project aims to build a continuously operating data collection system that harvests data from Reddit and 4chan. The collected data will serve as the foundation for further analysis in subsequent projects

2 Data Sources

2.1 Reddit

Reddit is a vast platform with numerous subreddits covering a wide range of topics. Due to the high volume of content, we focus on specific subreddits to manage the data collection effectively. The subreddits monitored are:

- memes
- dankmemes
- AdviceAnimals

We utilize Reddit's OAuth2 API for authentication and access:

- **API Endpoint for Posts:** https://oauth.reddit.com/r/{subreddit}/new
- **API Endpoint for Comments:** https://oauth.reddit.com/comments/{post_id}

2.2 4chan

4chan is an imageboard platform where users post anonymously. We focus on two of its boards:

- pol (Politically Incorrect)
- b (Random)

Data is collected using 4chan's public API:

- **API Endpoint for Catalog:** https://a.4cdn.org/{board}/catalog.json
- **API Endpoint for Threads:** https://a.4cdn.org/{board}/thread/{thread_id}.json

3 System Architecture and Implementation

The system is designed to operate continuously, fetching new data as it becomes available. The architecture comprises the following:

3.1 Core Components:

- **Data Producers:** Scripts that enqueue jobs to fetch posts and comments from Reddit and threads from 4chan.
- **Job Queue:** Implemented using Faktory, a background job processing system that manages job distribution.
- **Workers:** Processes that execute the jobs, fetching data from the APIs and storing it in MongoDB.
- **MongoDB:** A NoSQL database used to store the collected data in a structured format.

3.2 Project Structure:

The project is organized into distinct modules:

- config.py: Central configuration management
- enqueue_board_jobs.py: 4chan job producer
- enqueue_post_jobs.py: Reddit post job producer
- enqueue_comment_jobs.py: Reddit comment job producer
- utils.py: Shared utilities
- worker_fetch_boards.py: 4chan board collection worker
- worker_fetch_posts.py: Reddit post collection worker
- worker_fetch_comments.py: Reddit comment collection worker

Meme Trend Analysis System, July 2017, Washington, DC, USA

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

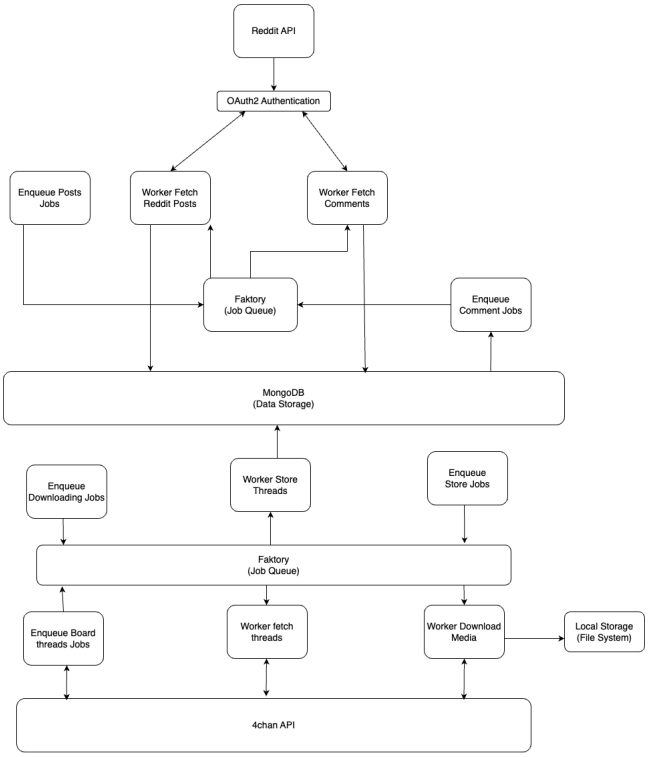

Figure 1: System Architecture

4 Implementation Details

4.1 Technologies and Libraries Used

- **Programming Language:** Python
- **HTTP Client:** requests library for making API calls.
- **Database:** pymongo library to interact with MongoDB.
- **Job Queue:** pyfaktory library to interact with Faktory.
- **Rate Limiting:** ratelimit library to adhere to API rate limits.

4.2 Reddit Data Collection

4.2.1 Posts Collection.

- **Authentication:** Utilizes OAuth2 to obtain an access token.
- **Job Enqueuing:** enqueue_posts_jobs.py script enqueues jobs every 5 minutes for each subreddit.
- **Worker Processing:** worker_fetch_posts.py fetches the latest posts and stores them in MongoDB, ensuring no duplicates by checking post IDs.

4.2.2 Comments Collection.

- **Job Enqueuing:** enqueue_comment_jobs.py identifies non-archived posts and enqueues jobs every hour.
- **Worker Processing:** worker_fetch_comments.py fetches comments for each post, handles pagination, and stores them hierarchically in MongoDB.

4.3 4chan Data Collection

- **Catalog Fetching:** enqueue_board_jobs.py fetches the catalog for each board every hour and enqueues jobs for each active thread.
- **Thread Processing:** worker_fetch_boards.py fetches thread data, downloads associated media, and stores structured data in MongoDB.
- **Media Handling:** Media files are downloaded and stored locally, with file paths saved in the database.

4.4 Continuous Operation

The system is designed to run indefinitely, with producers and workers operating in parallel:

- **Producers:** Enqueue jobs at regular intervals.
- **Workers:** Continuously poll the Faktory queue for new jobs.
- **Concurrency:** Managed using multiple worker processes for scalability.

5 Data Collection Statistics

5.1 Data Collected Over Time

Over the past three days, the system has collected:

- **Reddit Posts:** 861 posts.
- **Reddit Comments:** 13139 comments.
- **4chan Threads:** 1955 threads.

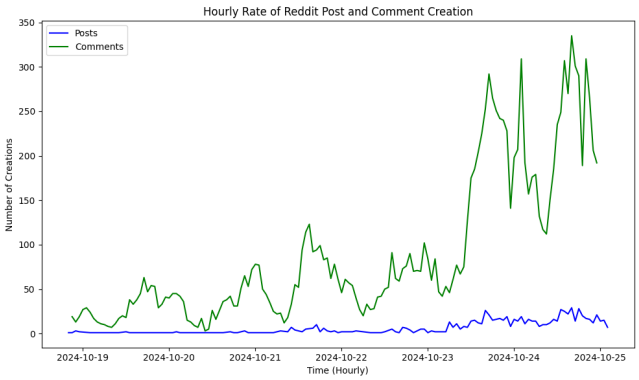

Figure 2: Hourly Rate of Reddit Post and Comment Creation

5.2 Projected Data Volume

Assuming a consistent data collection rate:

- **Daily Averages:**
  - Reddit Posts: 287 posts/day
  - Reddit Comments: 4,380 comments/day
  - 4chan Threads: 652 threads/day
- **Two-Month Projection (60 days):**
  - Reddit Posts: 17,220 posts
  - RReddit Comments: 262,800 comment
  - R4chan Threads: 39,120 threads

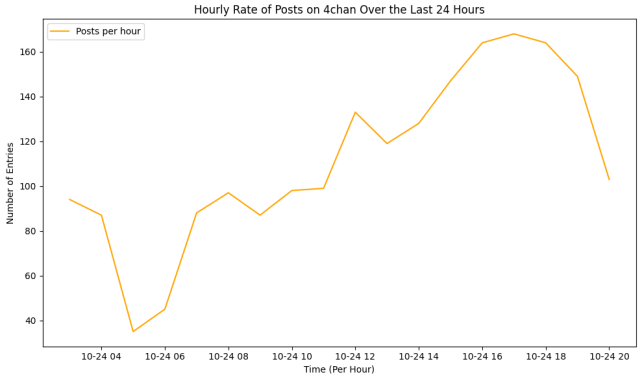

Figure 3: Hourly Rate of Posts on 4chan Over The Last 24 Hours

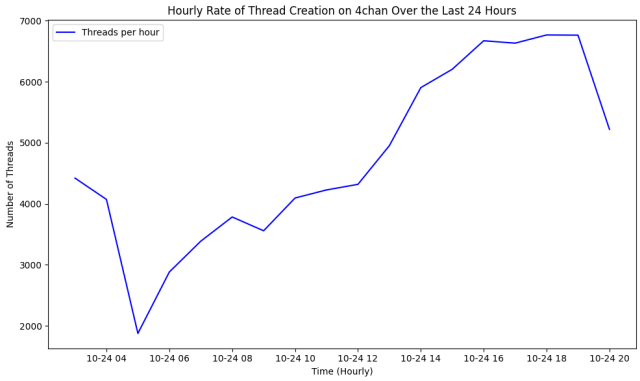

Figure 4: Hourly Rate of Thread Creation on 4chan Over The Last 24 Hours

6 Changes Since Proposal

- **Subreddit Selection:** Initially planned to monitor five subreddits; reduced to three due to data volume considerations.
- **Data Storage:** Switched from storing media files in the database to the file system for efficiency.
- **Rate Limiting:** Implemented stricter rate limiting to comply with API usage policies.

7 Challenges and Solutions

- **Faktory Setup and Integration:** Faced initial difficulties in setting up Faktory and integrating it with Python. Resolved by carefully configuring the pyfaktory client and ensuring proper authentication.
- **Reddit Authentication:** Managing OAuth2 tokens was challenging due to token expiration. Implemented a token refresh mechanism to maintain continuous operation.
- **API Rate Limits:** Encountered rate limit errors from Reddit API. Introduced the ratelimit library to throttle requests appropriately.
- **Data Duplication:** Duplicate entries were a concern. Enforced unique indexes in MongoDB collections to prevent duplicates.
- **Media Storage:** Downloading and storing media from 4chan threads required significant storage. Addressed by organizing media files systematically and monitoring disk usage.

8 Conclusion

The data collection system built for Reddit and 4chan demonstrates a scalable and efficient approach to continuous data gathering. While initial challenges included handling API authentication and rate limiting, the system is now functioning smoothly, with data projections showing significant volumes over the next two months. This project lays the groundwork for further data analysis, offering robust datasets for trend analysis and social behavior modeling.

References

[1] Reddit API Documentation. Reddit, Inc. Available at: https://www.reddit.com/dev/api/
[2] 4chan API Documentation. 4chan, LLC. Available at: https://github.com/4chan/4chan-API
[3] RFC 6749: The OAuth 2.0 Authorization Framework. IETF. Available at: https://datatracker.ietf.org/doc/html/rfc6749