ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÀI TẬP LỚN

MÔN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Đề tài: Thu thập và phân tích dữ liệu Twitter

Giảng viên hướng dẫn: TS. Trịnh Tuấn Đạt

Danh sách sinh viên thực hiện:

STT	Họ và tên	Mã sinh viên
1	Đồng Văn Hiếu	20224980
2	Nguyễn Hoàng Phương	20225070
3	Trần Đức Thái	20225085
4	Ngô Ngọc Đăng	20215564
5	Vũ Hoàng Tùng	20235241
6	Chhor Tithpanha	20211004

Mục lục

1 Giới thiệu **4**

2 Phân chia công việc và Đóng góp **5**

3 Nhật ký thảo luận nhóm **5**
3.1 Các buổi họp chính 5
3.2 Các buổi kiểm tra tiến độ 6

4 Thu thập dữ liệu **6**
4.1 Công cụ và công nghệ sử dụng 7
4.2 Thiết kế gói 7
4.3 Quy trình thu thập dữ liệu 9
4.4 Dữ liệu thu thập được 10
4.5 Ý nghĩa của dữ liệu 10

5 Xử lý dữ liệu **12**
5.1 Công cụ và công nghệ sử dụng 12
5.1.1 Cơ sở dữ liệu 13
5.2 Thiết kế gói 15
5.3 Quy trình xử lý dữ liệu 17
5.4 Kết quả xử lý dữ liệu 17
5.5 Ý nghĩa của dữ liệu 18

6 PageRank **19**
6.1 Công cụ và công nghệ sử dụng 19
6.2 Ý tưởng thành lập các loại cạnh 19
6.2.1 POST 20
6.2.2 FOLLOW 21
6.2.3 MENTION 21
6.2.4 RETWEET 22
6.2.5 REPLY và QUOTE 22
6.2.6 Cách hoạt động của bảng `User_Tweets` 23
6.3 Thiết kế Gói và Lớp 24
6.3.1 Công dụng của các gói và lớp 25
6.3.2 Các kỹ thuật lập trình hướng đối tượng (OOP) được sử dụng 26
6.4 Quy trình tính toán 27
6.5 Kết quả 29
6.6 Ý nghĩa kết quả nhận được 30

7 Visualization **31**
7.1 Công cụ và công nghệ sử dụng 31
7.2 Thiết kế gói và lớp 32
7.3 Quy trình hoạt động 32
7.4 Kết quả hiển thị 33
7.5 Ý nghĩa của Visualization 33

8 Quản lý cấu hình và tích hợp hệ thống **34**
8.1 Công cụ và công nghệ sử dụng 34
8.2 Thiết kế gói và lớp 34
8.3 Quy trình hoạt động 35
8.4 Kết quả 35
8.5 Ý nghĩa 35

9 Hướng dẫn sử dụng **36**
9.1 Chuẩn bị môi trường 36
9.2 Thu thập dữ liệu từ Twitter 36
9.3 Import dữ liệu và tính toán PageRank 37
9.4 Hiển thị kết quả PageRank (tùy chọn) 37

10 Ảnh demo sản phẩm **38**
10.1 Giao diện thu thập dữ liệu 38
10.2 Quá trình xử lý dữ liệu 40
10.3 Kết quả tính toán PageRank 41
10.4 Giao diện hiển thị đồ thị 42

Danh sách hình vẽ

1 Sơ đồ gói và tiện ích hỗ trợ . . . 7
2 Sơ đồ quan hệ cơ sở dữ liệu . . . 13
3 Sơ đồ gói xử lý dữ liệu và các thành phần liên quan . . . 15
4 Liên kết hình thành từ `POST` . . . 20
5 Liên kết hình thành từ `FOLLOW` . . . 21
6 Liên kết hình thành từ `MENTION` . . . 21
7 Liên kết hình thành từ `RETWEET` . . . 22
8 Liên kết hình thành từ `REPLY` và `QUOTE` . . . 23
9 Biểu đồ lớp của gói `caculator` . . . 24
10 Biểu đồ lớp của gói `adjacency_list_builder` . . . 25
11 Ví dụ hiển thị đồ thị tương tác bằng Cytoscape.js . . . 33
12 Giao diện Selenium tự động thu thập thông tin từ Twitter. . . . 38
13 Sử dụng TwitterAPI kèm để lấy thông tin retweeters. . . . 38
14 File JSON chứa danh sách người theo dõi (`followers.json`). . . . 39
15 File JSON chứa danh sách người đang theo dõi bởi người dùng (`following.json`). 39
16 File JSON chứa danh sách các tweet (`retweetData.json`). . . . 40
17 Bảng `Users` được nhập vào cơ sở dữ liệu PostgreSQL. . . . 40
18 File danh sách kề (`directedSimpleGraph.json`) được xuất ra từ quá trình xử lý. 41
19 Kết quả tính toán PageRank trong file `pageRankPoints.json`. . . . 41
20 Giao diện hiển thị đồ thị với Cytoscape.js. Các node và edges được hiển thị rõ ràng, với kích thước và màu sắc dựa trên giá trị PageRank. . . . 42

Danh sách bảng

1 Phân chia công việc và đóng góp của các thành viên . . . 5
2 Dữ liệu thu thập được . . . 11
3 Kết quả xử lý dữ liệu . . . 18
4 Bảng kết quả PageRank tính toán được . . . 29

Tóm tắt nội dung

Trong kỷ nguyên bùng nổ thông tin, công nghệ blockchain nổi lên như một biểu tượng của sự tin cậy và minh bạch. Dự án này trình bày phương pháp thu thập và phân tích dữ liệu trên Twitter nhằm tìm kiếm những KOL (Key Opinion Leaders) trong lĩnh vực blockchain, giúp người dùng có cái nhìn tổng quan về các xu hướng và phát triển mới nhất.

Báo cáo này trình bày cách thiết kế, triển khai, và đánh giá một hệ thống tự động hóa tìm kiếm và xếp hạng các KOL bằng thuật toán PageRank, đồng thời xây dựng đồ thị kết nối giữa người dùng và tweet, hỗ trợ phân tích dữ liệu hiệu quả. Hệ thống này hứa hẹn là công cụ mạnh mẽ trong việc khám phá những tác động của blockchain trong thời đại số.

1 Giới thiệu

Mục tiêu

Mục tiêu chính của dự án này là phát triển một hệ thống thu thập và phân tích dữ liệu từ nền tảng Twitter, tập trung vào việc khám phá các nhân vật KOL có ảnh hưởng trong lĩnh vực blockchain. Dự án tập trung vào các mục tiêu cụ thể sau:

1. Xác định danh sách các KOL trong lĩnh vực blockchain thông qua các từ khóa và hashtag phù hợp.
2. Thu thập dữ liệu về các tweet và followers của các KOL này để xây dựng đồ thị mạng xã hội.
3. Tự động hóa quá trình thu thập và phân tích dữ liệu bằng cách sử dụng Selenium và thuật toán PageRank.
4. Xây dựng đồ thị kết nối giữa người dùng và tweet với số lượng node và cạnh đủ lớn.
5. Tính toán và xếp hạng mức độ ảnh hưởng của các KOL dựa trên đồ thị đã xây dựng.

Dự án không chỉ mang lại góc nhìn toàn diện về các KOL mà còn cung cấp công cụ hỗ trợ phân tích dữ liệu hữu ích cho lĩnh vực blockchain.

2 Phân chia công việc và Đóng góp

No.	Thành Viên	Công việc	Đóng góp (%)
1	Đồng Văn Hiếu	Quản lý dự án, Thiết kế hệ thống, Xử lý dữ liệu, Xây dựng thuật toán PageRank, Viết báo cáo	20%
2	Nguyễn Hoàng Phương	Quản lý dự án, Thiết kế hệ thống, Thu thập dữ liệu, Xử lý dữ liệu, Viết báo cáo và Demo dự án	20%
3	Trần Đức Thái	Visualization	15%
4	Ngô Ngọc Đăng	Thu thập dữ liệu	15%
5	Vũ Hoàng Tùng	Thu thập dữ liệu	15%
6	Chhor Tithpanha	Visualization	15%

Bảng 1: Phân chia công việc và đóng góp của các thành viên

3 Nhật ký thảo luận nhóm

3.1 Các buổi họp chính

- **Ngày 28/09/2024 (Discord, 19:30 – 21:00):**
  - Thảo luận về yêu cầu dự án và mục tiêu chính: Thu thập dữ liệu từ Twitter, xử lý, và phân tích thuật toán PageRank.
  - Phân công nhiệm vụ ban đầu:
    * Nguyễn Hoàng Phương phụ trách tìm hiểu và xây dựng gói `scraper`.
    * Đồng Văn Hiếu đảm nhận thiết kế cơ sở dữ liệu và thuật toán PageRank.
    * Ngô Ngọc Đăng và Vũ Hoàng Tùng hỗ trợ thu thập dữ liệu thực tế.
    * Trần Đức Thái và Chhor Tithpanha phụ trách trực quan hóa kết quả.
- **Ngày 10/10/2024 (Discord, 20:00 – 21:30):**
  - Nguyễn Hoàng Phương trình bày tiến độ xây dựng gói `scraper` với các lớp như `FollowsScraper`, `TweetIDScraper`.
  - Đồng Văn Hiếu thảo luận về cấu trúc cơ sở dữ liệu SQL với các bảng chính như `Users`, `Tweets`, và `User_Follows`.
  - Đưa ra hướng dẫn sử dụng lớp `RetweetUserInfo` để thu thập thông tin retweeters và mentions.
  - Phân tích các lỗi giới hạn API Twitter và đề xuất cách khắc phục.
- **Ngày 25/10/2024 (Discord, 19:30 – 21:00):**
  - Ngô Ngọc Đăng và Vũ Hoàng Tùng báo cáo tiến độ thu thập dữ liệu followers, following, tweet ID, và danh sách retweeters.

  - Đồng Văn Hiếu trình bày về tiến trình xây dựng thuật toán PageRank, cách chuyển đổi dữ liệu từ cơ sở dữ liệu sang danh sách kề.
  - Thảo luận và thống nhất cấu trúc dữ liệu đầu ra phục vụ cho phần trực quan hóa.

- **Ngày 15/11/2024 (Discord, 20:00 – 21:30):**
  - Trần Đức Thái và Chhor Tithpanha trình bày tiến độ xây dựng công cụ trực quan hóa bằng `Cytoscape.js`.
  - Thảo luận về giao diện đồ thị, bao gồm màu sắc node, kích thước dựa trên PageRank, và bố cục đồ thị.
  - Đồng Văn Hiếu báo cáo kết quả tính toán PageRank, tập trung vào các user có ảnh hưởng lớn.

- **Ngày 10/12/2024 (Discord, 19:30 – 21:00):**
  - Tổng hợp báo cáo từ các thành viên, kiểm tra các phần đã viết.
  - Nguyễn Hoàng Phương và Đồng Văn Hiếu trình bày phần tổng quan về thiết kế và thuật toán PageRank.
  - Kiểm tra phần trực quan hóa, đảm bảo kết quả hiển thị chính xác trước buổi báo cáo cuối kỳ.

3.2 Các buổi kiểm tra tiến độ

Ngoài các buổi họp chính, nhóm tổ chức các buổi kiểm tra tiến độ tại lớp vào thứ Tư hàng tuần trong tháng 10 và 11. Nội dung chủ yếu bao gồm:

- Theo dõi tiến độ thực hiện của từng thành viên.
- Kiểm tra và khắc phục lỗi trong các phần đã triển khai, như thu thập dữ liệu và tính toán PageRank.
- Đưa ra ý kiến và góp ý cải thiện phần trực quan hóa.

Các buổi kiểm tra tiến độ đã giúp đảm bảo rằng dự án được thực hiện đúng kế hoạch và các thành viên phối hợp hiệu quả.

4 Thu thập dữ liệu

Thu thập dữ liệu là một phần quan trọng trong dự án, giúp tạo cơ sở dữ liệu để xây dựng đồ thị và phân tích thuật toán PageRank. Mục tiêu là thu thập thông tin từ Twitter, tập trung vào các tài khoản liên quan đến blockchain, để phân tích các mối quan hệ và mức độ ảnh hưởng của các KOL trong cộng đồng này.

Phần này được triển khai bằng ngôn ngữ Java, sử dụng các công cụ mạnh mẽ như Selenium và Twitter API. Dữ liệu được thu thập và lưu trữ dưới dạng JSON với các trường thông tin bao gồm:

- **Danh sách các người theo dõi (followers) và người đang theo dõi (following)** của một người dùng.
- **Danh sách các Tweet**: Bao gồm ID, các retweeters và cả các mention người dùng khác (nếu có).

Dữ liệu này sẽ đóng vai trò làm nền tảng để xây dựng đồ thị mạng xã hội, hỗ trợ phân tích mức độ ảnh hưởng của các KOL.

4.1 Công cụ và công nghệ sử dụng

Quá trình thu thập dữ liệu sử dụng nhiều công cụ và thư viện để đảm bảo tính hiệu quả và chính xác. Các công cụ chính bao gồm:

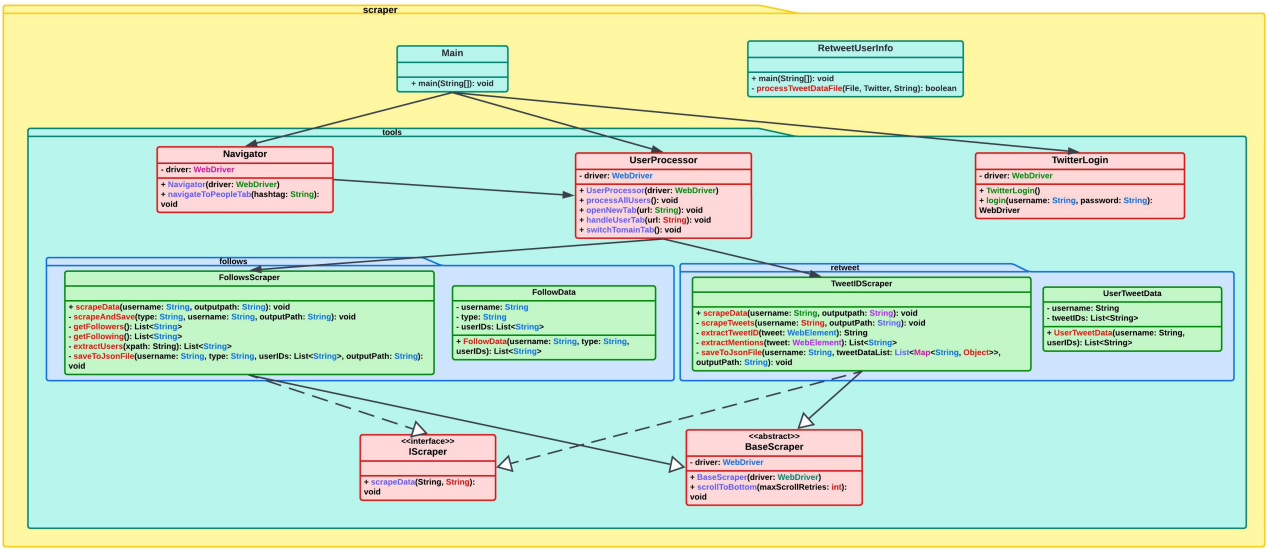
Selenium: Selenium là một công cụ tự động hóa trình duyệt web, giúp xử lý các tác vụ tương tác động như đăng nhập, cuộn trang, và thu thập dữ liệu từ giao diện web. Với Selenium, dự án có thể tự động hóa việc tìm kiếm tài khoản Twitter liên quan đến từ khóa `#blockchain` và thu thập thông tin như danh sách followers, following và tweet ID. Trong dự án, trình duyệt Chrome được sử dụng để thực hiện các tác vụ tự động hóa.

Twitter API (Twitter4J-v2): API của Twitter cung cấp khả năng lấy thông tin nâng cao, chẳng hạn như danh sách người dùng đã retweet một bài viết cụ thể. Công cụ này rất cần thiết để bổ sung dữ liệu từ các tweet đã thu thập.

Jackson ObjectMapper: Thư viện này được sử dụng để xử lý dữ liệu JSON. Nó hỗ trợ đọc, ghi và định dạng dữ liệu JSON, giúp lưu trữ thông tin đã thu thập vào các file JSON như `followers.json`, `following.json`, và `tweetID.json`.

Các công cụ này kết hợp với nhau để tạo ra một quy trình thu thập dữ liệu hiệu quả, cho phép dự án đạt được mục tiêu xây dựng một cơ sở dữ liệu phong phú phục vụ phân tích.

4.2 Thiết kế gói



Hình 1: Sơ đồ gói và tiện ích hỗ trợ

Như được thể hiện trong Hình 1, mã nguồn của phần này được tổ chức trong gói chính `scraper`, bao gồm nhiều gói con và lớp để quản lý các khía cạnh khác nhau của quy trình thu thập và xử lý dữ liệu.

Gói `tools` là trung tâm của toàn bộ quá trình, bao gồm các gói con `follows` và `retweet`, được thiết kế để xử lý dữ liệu người theo dõi và bài đăng trên Twitter. Mỗi gói con chứa các lớp chuyên biệt để đảm nhiệm các chức năng cụ thể, được trình bày chi tiết dưới đây:

- `BaseScraper`: Một lớp trừu tượng cung cấp các phương thức cơ bản như `scrollToBottom`, cho phép thao tác cuộn trang trên giao diện Twitter. Lớp này được kế thừa bởi các lớp như `FollowsScraper` và `TweetIDScraper`.

- `IScraper`: Một giao diện định nghĩa phương thức `scrapeData`, đảm bảo rằng tất cả các lớp triển khai sẽ tuân thủ một cấu trúc chung trong quy trình thu thập dữ liệu.

Gói `follows` chứa lớp `FollowsScraper`, chịu trách nhiệm thu thập danh sách người theo dõi (*followers*) và người đang theo dõi (*following*) của một người dùng Twitter. Dữ liệu được lưu trữ dưới định dạng JSON thông qua các phương thức chuyên biệt như `scrapeAndSave`.

Gói `retweet` chứa lớp `TweetIDScraper`, tập trung vào thu thập thông tin về tweet ID và danh sách mentions từ các bài đăng trên Twitter. Cả hai lớp này đều kế thừa từ `BaseScraper` và triển khai giao diện `IScraper`.

Ngoài ra, các lớp hỗ trợ khác bao gồm:

- `Navigator`: Hỗ trợ điều hướng đến các trang cụ thể trên Twitter dựa trên đầu vào như hashtag. Ví dụ, điều hướng đến tab *People*.
- `TwitterLogin`: Cung cấp khả năng đăng nhập tự động vào Twitter, tạo một môi trường Selenium ổn định cho các scraper hoạt động.
- `UserProcessor`: Quản lý toàn bộ luồng xử lý chính, phối hợp với các lớp `FollowsScraper` và `TweetIDScraper` để thu thập dữ liệu từ danh sách người dùng.

Các kỹ thuật lập trình hướng đối tượng (OOP) được áp dụng

Thiết kế gói này áp dụng nhiều kỹ thuật OOP quan trọng, giúp tăng tính hiệu quả, dễ bảo trì, và khả năng mở rộng của hệ thống:

1. Trừu tượng hóa (Abstraction): Lớp `BaseScraper` là một lớp trừu tượng, đóng vai trò định nghĩa các chức năng chung cần thiết cho tất cả các lớp scraper, như `scrollToBottom`. Thay vì để các lớp con phải tự triển khai lại logic này, `BaseScraper` tập trung tất cả các chức năng chung, giúp loại bỏ sự lặp lại và giảm độ phức tạp của mã nguồn. Trừu tượng hóa giúp lập trình viên chỉ cần tập trung vào những gì cần thực hiện, thay vì chi tiết cách nó được thực hiện.

2. Kế thừa (Inheritance): Cơ chế kế thừa được sử dụng để tái sử dụng mã nguồn một cách hiệu quả. Các lớp như `FollowsScraper` và `TweetIDScraper` kế thừa từ `BaseScraper`, giúp sử dụng lại các phương thức như `scrollToBottom` mà không cần triển khai lại. Điều này đảm bảo rằng nếu cần thay đổi logic chung, chỉ cần cập nhật trong lớp `BaseScraper`, các lớp con sẽ tự động nhận được thay đổi.

3. Đa hình (Polymorphism): Giao diện `IScraper` định nghĩa phương thức `scrapeData`, được triển khai riêng trong từng lớp scraper như `FollowsScraper` và `TweetIDScraper`. Nhờ đa hình, hệ thống có thể gọi phương thức `scrapeData` trên đối tượng kiểu `IScraper` mà không cần biết lớp cụ thể của nó. Điều này giúp mã nguồn linh hoạt hơn khi cần mở rộng thêm các scraper khác trong tương lai.

4. Đóng gói (Encapsulation): Các thuộc tính như `driver` trong các lớp scraper được khai báo ở mức `private` hoặc `protected`, đảm bảo rằng chúng chỉ có thể được truy cập hoặc chỉnh sửa thông qua các phương thức được định nghĩa rõ ràng. Điều này ngăn ngừa việc truy cập trái phép và bảo vệ tính toàn vẹn của dữ liệu. Đồng thời, việc đóng gói giúp giảm thiểu lỗi không mong muốn do tương tác không kiểm soát giữa các thành phần.

Lợi ích của việc áp dụng OOP

- **Tái sử dụng mã nguồn:** Nhờ cơ chế kế thừa và trừu tượng hóa, các logic chung được tập trung trong `BaseScraper`, cho phép tái sử dụng dễ dàng trong các lớp khác.
- **Dễ dàng mở rộng:** Việc thêm các scraper mới chỉ cần kế thừa `BaseScraper` và triển khai các phương thức cụ thể mà không làm thay đổi mã nguồn hiện tại.
- **Dễ bảo trì:** Với việc sử dụng đóng gói, các thành phần được tách biệt rõ ràng, giảm thiểu rủi ro gây lỗi khi thay đổi mã nguồn.
- **Tính linh hoạt:** Đa hình cho phép xử lý các đối tượng scraper một cách tổng quát, mà không cần biết lớp cụ thể của chúng. Điều này đặc biệt hữu ích trong các trường hợp như gọi `scrapeData` trên nhiều loại scraper khác nhau.
- **Tăng tính tổ chức:** Mã nguồn được chia thành các lớp và gói rõ ràng, giúp lập trình viên dễ dàng điều hướng và hiểu cấu trúc của dự án.

Thiết kế gói này không chỉ đảm bảo tính module hóa cao mà còn tận dụng tối đa sức mạnh của lập trình hướng đối tượng, giúp hệ thống dễ dàng mở rộng, bảo trì và nâng cấp trong tương lai.

4.3 Quy trình thu thập dữ liệu

Quy trình thu thập dữ liệu bao gồm các bước chính sau:

1. **Đăng nhập vào Twitter**: Sử dụng Selenium để tự động hóa quá trình đăng nhập vào tài khoản Twitter.
2. **Tìm kiếm và thu thập thông tin người dùng**:
   - Sử dụng Selenium để tự động tìm kiếm các tài khoản liên quan đến từ khóa `#blockchain`.
   - Thu thập danh sách followers và following của các tài khoản này.
3. **Thu thập tweet ID và mentions**:
   - Duyệt qua timeline của người dùng để thu thập tweet ID và mentions trong bài viết.
4. **Thu thập danh sách retweeters**:
   - Sử dụng Twitter API để duyệt qua các TweetID đã thu thập được và lấy danh sách người dùng đã retweet các bài viết đó.
5. **Lưu trữ dữ liệu**:
   - Lưu dữ liệu vào các file JSON bao gồm `followers.json`, `following.json`, `tweetID.json`, và `retweetData.json`.

4.4 Dữ liệu thu thập được

Dữ liệu thu thập được bao gồm:

- `followers.json`: Danh sách followers của mỗi tài khoản.
- `following.json`: Danh sách following của mỗi tài khoản.
- `retweetData.json`: Danh sách các thông tin liên quan Tweet bao gồm tweetID, những người retweet, người dùng được mention trong bài viết.

Ví dụ mẫu dữ liệu:
followers.json

```
{
  "username" : "@ArgoBlockchain",
  "type" : "Followers",
  "userIDs" : ["@TuukkaJoutsenl1", "@GarethAyres", "@EmilAharonov"]
}
```

following.json

```
{
  "username" : "@ArgoBlockchain",
  "type" : "Following",
  "userIDs" : [ "@staratlas", "@dfinity", "@cosmos"]
}
```

retweetData.json

```
{
"tweets" : [ {
    "mentions" : [ ],
    "retweeters" : [ "@JamesIIIIIII", "@alboyBill", "@tddivine", "@shanksyp" ],
    "tweetID" : "1654440807598882816"
  }, {
    "mentions" : [ "@ePICBlockchain" ],
    "retweeters" : [ "@TokaliMich26615", "@babatundemuham6", "@tacticalinvest_"],
    "tweetID" : "1654021792493707266"
    } ]
"username" : "@ArgoBlockchain"
}
```

4.5 Ý nghĩa của dữ liệu

Dữ liệu này được sử dụng để xây dựng đồ thị xã hội, trong đó:

- **Node**: Có thể là người dùng hoặc bài viết.
- **Edge**: Kết nối thể hiện mối quan hệ giữa người dùng (followers, following, retweet) và giữa người dùng với bài viết (post, mentions, retweet).

Các dữ liệu này sẽ được đưa vào trong một hệ quản trị cơ sở dữ liệu để đảm bảo dữ liệu luôn được chuẩn hoá. Sau đó sẽ được trích xuất và phân tích bằng thuật toán PageRank để đánh giá mức độ ảnh hưởng của các KOL trong lĩnh vực Blockchain.

Danh mục	**Mô tả**
2180 người dùng	Được thu thập từ 11 tài khoản KOL ban đầu trong đó có **144 người dùng được mention**, còn lại là những người đã retweet và followers/following.
376 quan hệ theo dõi	Bao gồm các kết nối giữa người dùng trong danh sách followers và following.
88 bài tweet	Được thu thập từ 11 tài khoản KOL.
4742 loại tương tác	Bao gồm retweet, mentions, post.

Bảng 2: Dữ liệu thu thập được

5 Xử lý dữ liệu

Xử lý dữ liệu là một bước quan trọng trong dự án, nhằm chuyển đổi dữ liệu thô được thu thập từ Twitter thành một cơ sở dữ liệu chuẩn hóa. Quá trình này bao gồm việc nhập dữ liệu vào hệ quản trị cơ sở dữ liệu, sau đó trích xuất dữ liệu để thực hiện các phân tích bởi thuật toán PageRank.

Phần này được triển khai bằng ngôn ngữ Java, với các gói và lớp được thiết kế rõ ràng để đảm bảo hiệu quả và dễ dàng bảo trì. Dữ liệu thô được lưu trong các tệp JSON sẽ được nhập vào cơ sở dữ liệu thông qua các lớp nhập dữ liệu (`dataimporterservices`), sau đó được lưu trữ trong cơ sở dữ liệu được thiết kế theo mô hình quan hệ.

5.1 Công cụ và công nghệ sử dụng

Quá trình xử lý dữ liệu được thực hiện bằng cách kết hợp các công cụ và thư viện mạnh mẽ, đảm bảo hiệu quả và tính toàn vẹn của dữ liệu trong suốt quá trình từ thu thập, nhập liệu, đến lưu trữ và xử lý. Các công cụ chính bao gồm:

Jackson ObjectMapper: Jackson là một thư viện xử lý JSON mạnh mẽ được sử dụng để phân tích cú pháp (parsing) và ghi dữ liệu JSON trong Java. Với `ObjectMapper`, chúng tôi có thể:

- Chuyển đổi trực tiếp từ tệp JSON thành các đối tượng Java, hỗ trợ kiểu dữ liệu phức tạp như danh sách (`List`) và bản đồ (`Map`).
- Xử lý các cấu trúc JSON không cố định, dễ dàng tương thích với dữ liệu thô được thu thập từ các nguồn khác nhau.
- Định dạng lại dữ liệu JSON trước khi lưu trữ hoặc xử lý, đảm bảo tuân thủ các yêu cầu của cơ sở dữ liệu.

Jackson ObjectMapper giúp giảm đáng kể thời gian và công sức xử lý dữ liệu JSON, đảm bảo tính chính xác và dễ dàng bảo trì.

JDBC (Java Database Connectivity): JDBC là API tiêu chuẩn của Java để kết nối và tương tác với cơ sở dữ liệu. Trong dự án này, JDBC được sử dụng để thực hiện:

- Kết nối với cơ sở dữ liệu PostgreSQL thông qua lớp `DatabaseConnector`, đảm bảo thiết lập một kết nối ổn định.
- Thực thi các câu lệnh SQL, bao gồm truy vấn (`SELECT`) và cập nhật dữ liệu (`INSERT`, `UPDATE`).
- Sử dụng các câu lệnh `PreparedStatement` để tránh các lỗi liên quan đến SQL Injection, đảm bảo an toàn cho dữ liệu.

Với sự hỗ trợ của JDBC, hệ thống có thể xử lý hàng ngàn giao dịch dữ liệu một cách nhanh chóng và đáng tin cậy.

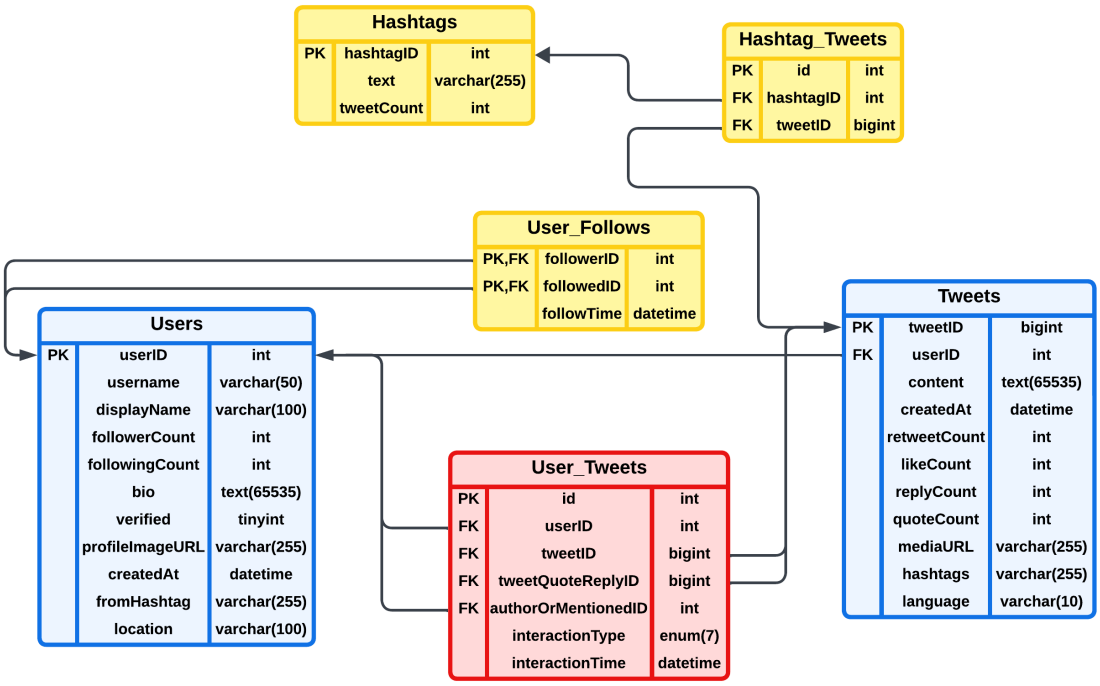
PostgreSQL: PostgreSQL là hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở được sử dụng trong dự án để lưu trữ dữ liệu đã được chuẩn hóa. PostgreSQL được lựa chọn nhờ các đặc điểm vượt trội sau:

- **Hỗ trợ ràng buộc toàn vẹn dữ liệu:** Các bảng như `Users`, `Tweets`, và `User_Follows` được thiết kế với các khóa ngoại (`FOREIGN KEY`) để đảm bảo tính toàn vẹn tham chiếu giữa các bảng.
- **Khả năng xử lý khối lượng dữ liệu lớn:** PostgreSQL tối ưu hóa hiệu suất xử lý với các chỉ mục (`INDEX`) trên các cột thường xuyên được truy vấn như `userID`, `tweetID`.
- **Cú pháp SQL hiện đại:** PostgreSQL hỗ trợ cú pháp SQL tiên tiến, cho phép sử dụng các tính năng như câu lệnh `ON CONFLICT` để tránh trùng lặp dữ liệu trong quá trình nhập liệu.
- **Khả năng mở rộng:** PostgreSQL có thể mở rộng để xử lý các yêu cầu phức tạp, như tính toán dữ liệu theo thời gian thực hoặc tích hợp với các công cụ phân tích dữ liệu khác.

Trong dự án này, PostgreSQL được sử dụng để lưu trữ dữ liệu người dùng, tweet, và các mối quan hệ giữa chúng. Cấu trúc cơ sở dữ liệu được định nghĩa trong tệp `schema.sql`, đảm bảo rằng dữ liệu luôn được lưu trữ theo một mô hình chuẩn hóa, dễ dàng truy vấn và phân tích.

5.1.1 Cơ sở dữ liệu

Cơ sở dữ liệu trong hệ thống được thiết kế theo mô hình quan hệ, với cấu trúc chặt chẽ nhằm đảm bảo tính toàn vẹn và hiệu quả trong việc xử lý dữ liệu. Như thể hiện trong Hình 2, các bảng quan trọng trong hệ thống bao gồm:



Hình 2: Sơ đồ quan hệ cơ sở dữ liệu

`Users:` Bảng `Users` lưu trữ thông tin chi tiết về người dùng, với các trường quan trọng như:

- **userID**: Khóa chính, định danh duy nhất cho mỗi người dùng.
- **username**: Tên tài khoản Twitter của người dùng (duy nhất và không được để trống).
- **displayName, bio, location**: Các thông tin mô tả người dùng.
- **followerCount, followingCount**: Số lượng người theo dõi và đang theo dõi của người dùng.
- **verified**: Xác định người dùng đã được xác minh hay chưa.
- **createdAt**: Thời điểm tài khoản được thêm vào cơ sở dữ liệu.

`Tweets:` Bảng `Tweets` lưu trữ thông tin về các bài đăng:

- **tweetID**: Khóa chính, định danh duy nhất cho mỗi tweet.
- **userID**: Khóa ngoại, tham chiếu đến bảng `Users`, xác định người dùng đã đăng bài.
- **content**: Nội dung của tweet.
- **retweetCount, likeCount, replyCount, quoteCount**: Các số liệu thống kê về tương tác với tweet.
- **hashtags**: Danh sách hashtag liên quan đến tweet (nếu có).

`User_Tweets:` Bảng `User_Tweets` lưu trữ các tương tác giữa người dùng và tweet:

- **userID**: Khóa ngoại, tham chiếu đến người dùng thực hiện hành động.
- **tweetID**: Khóa ngoại, tham chiếu đến tweet được tương tác.
- **interactionType**: Loại tương tác, có thể là `MENTION`, `RETWEET`, `QUOTE`, hoặc `REPLY`.
- **interactionTime**: Thời gian xảy ra tương tác.

`User_Follows:` Bảng `User_Follows` lưu trữ mối quan hệ theo dõi giữa các người dùng:

- **followerID**: Khóa ngoại, định danh người theo dõi.
- **followedID**: Khóa ngoại, định danh người được theo dõi.
- **followTime**: Thời gian bắt đầu theo dõi.

`Hashtags` **và** `Hashtag_Tweets:` Hai bảng này lưu trữ thông tin về các hashtag và mối liên kết của chúng với các tweet:

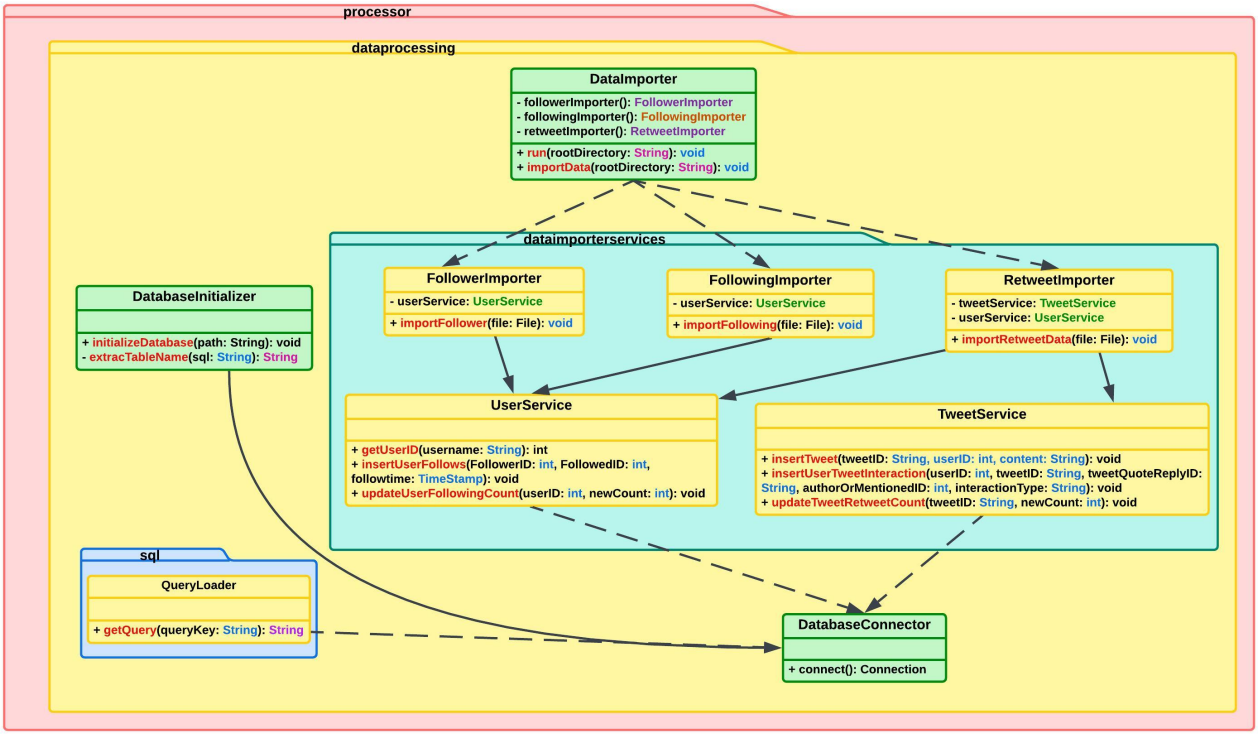
- **Hashtags**: Chứa các thông tin về hashtag như `text` (nội dung hashtag) và `tweetCount` (số lần xuất hiện trong các tweet).
- **Hashtag_Tweets**: Xác định mối quan hệ giữa một hashtag và các tweet mà nó xuất hiện.

Các ràng buộc và quan hệ quan trọng

- **Khóa ngoại:** Các bảng `Tweets`, `User_Tweets`, `User_Follows`, và `Hashtag_Tweets` đều có các khóa ngoại tham chiếu đến bảng `Users` hoặc `Tweets`, đảm bảo tính toàn vẹn dữ liệu.
- **Xóa lan truyền (ON DELETE CASCADE):** Khi một người dùng hoặc tweet bị xóa, tất cả các dữ liệu liên quan trong các bảng khác cũng được tự động xóa để tránh dữ liệu "mồ côi".
- **Ràng buộc kiểm tra:** Trường `interactionType` trong bảng `User_Tweets` chỉ cho phép các giá trị hợp lệ (`MENTION`, `RETWEET`, `QUOTE`, `REPLY`).

Thiết kế cơ sở dữ liệu này không chỉ đảm bảo tính toàn vẹn và logic dữ liệu mà còn hỗ trợ tối ưu hóa cho các truy vấn phân tích, giúp hệ thống vận hành một cách hiệu quả và chính xác.

5.2 Thiết kế gói



Hình 3: Sơ đồ gói xử lý dữ liệu và các thành phần liên quan

Như được thể hiện trong Hình 3, gói `processor.dataprocessing` được thiết kế để tổ chức toàn bộ quá trình xử lý dữ liệu từ việc khởi tạo cơ sở dữ liệu, nhập liệu từ các tệp JSON, đến việc thực thi các truy vấn SQL. Gói được chia thành các thành phần chính như sau:

Gói chính: `processor.dataprocessing` Gói chính chứa các lớp chịu trách nhiệm điều phối và xử lý dữ liệu cấp cao:

- `DatabaseConnector`: Lớp này chịu trách nhiệm kết nối với cơ sở dữ liệu PostgreSQL thông qua JDBC. Nó cung cấp phương thức `connect()`, trả về đối tượng `Connection` để thực hiện các thao tác truy vấn SQL. Lớp này áp dụng nguyên tắc **đóng gói** (Encapsulation)

để đảm bảo rằng các thông tin về kết nối (URL, username, password) được bảo vệ thông qua lớp cấu hình.

- `DatabaseInitializer`: Lớp đảm bảo rằng cơ sở dữ liệu được khởi tạo chính xác theo cấu trúc được định nghĩa trong tệp `schema.sql`. Phương thức `initializeDatabase()` đọc tệp SQL, phân tách các câu lệnh và thực thi chúng tuần tự. Việc trích xuất tên bảng từ câu lệnh SQL được thực hiện qua phương thức `extractTableName()`, giúp ghi log rõ ràng cho từng bước khởi tạo.

- `DataImporter`: Lớp này đóng vai trò điều phối quá trình nhập dữ liệu từ các tệp JSON vào cơ sở dữ liệu. Nó phối hợp với các lớp trong gói con `dataimporterservices` để xử lý từng loại tệp dữ liệu (`followers.json`, `following.json`, `retweetData.json`). Lớp sử dụng các nguyên tắc **phân chia trách nhiệm** (Separation of Concerns) để đảm bảo từng lớp nhập liệu chỉ chịu trách nhiệm xử lý loại dữ liệu cụ thể.

Gói con: `dataimporterservices` Gói này bao gồm các lớp hỗ trợ việc nhập dữ liệu cụ thể:

- `FollowerImporter`: Lớp nhập danh sách người theo dõi từ tệp `followers.json`. Nó sử dụng `UserService` để kiểm tra, thêm người dùng mới, và cập nhật thông tin trong cơ sở dữ liệu.

- `FollowingImporter`: Lớp nhập danh sách người đang theo dõi từ tệp `following.json`. Cơ chế xử lý tương tự như `FollowerImporter`.

- `RetweetImporter`: Lớp nhập danh sách người dùng retweet và mentions từ tệp `retweetData.json`. Nó phối hợp với cả `UserService` và `TweetService` để xử lý dữ liệu phức tạp hơn, bao gồm việc thêm các tweet mới, cập nhật retweet count, và thêm các tương tác giữa người dùng và tweet.

Gói con: `sql` Gói này chứa lớp `QueryLoader`, chịu trách nhiệm tải và quản lý các truy vấn SQL từ tệp `queries.sql`. Truy vấn được định nghĩa và đặt tên rõ ràng, giúp dễ dàng sử dụng và bảo trì.

Các kỹ thuật lập trình hướng đối tượng (OOP) được áp dụng

Thiết kế gói này tận dụng nhiều kỹ thuật OOP quan trọng để đảm bảo tính mở rộng, bảo trì và hiệu quả của hệ thống:

1. Trừu tượng hóa (Abstraction): Lớp `DatabaseConnector` và `DatabaseInitializer` che giấu chi tiết kết nối và khởi tạo cơ sở dữ liệu. Điều này giúp các lớp khác sử dụng mà không cần lo lắng về chi tiết cách thức hoạt động của cơ sở dữ liệu.

2. Đóng gói (Encapsulation): Các lớp như `FollowerImporter`, `FollowingImporter`, và `RetweetImporter` sử dụng các thuộc tính `private` hoặc `protected` để bảo vệ dữ liệu bên trong. Việc truy cập hoặc chỉnh sửa dữ liệu chỉ được thực hiện thông qua các phương thức được định nghĩa rõ ràng, đảm bảo tính toàn vẹn của dữ liệu.

3. Phân chia trách nhiệm (Separation of Concerns): Mỗi lớp trong gói con `dataimporterservices` chịu trách nhiệm xử lý một loại dữ liệu cụ thể, đảm bảo rằng mã nguồn dễ hiểu và dễ bảo trì. Ví dụ, `FollowerImporter` chỉ tập trung vào dữ liệu người theo dõi, trong khi `RetweetImporter` xử lý dữ liệu phức tạp hơn từ retweet.

4. Kế thừa (Inheritance): Lớp `DataImporter` không kế thừa trực tiếp các lớp importer, nhưng thông qua việc sử dụng chúng, nó thể hiện một cơ chế "tái sử dụng mã nguồn"tương tự như kế thừa.

5. Phụ thuộc (Dependency Injection): Các lớp như `FollowerImporter`, `FollowingImporter`, và `RetweetImporter` phụ thuộc vào `UserService` và `TweetService`. Phụ thuộc này được quản lý rõ ràng, giúp các lớp có thể phối hợp hoạt động mà không vi phạm tính module hóa.

Lợi ích của việc áp dụng OOP

- **Tính mở rộng:** Dễ dàng thêm các lớp mới, ví dụ như các importer khác, mà không cần thay đổi cấu trúc hiện tại.
- **Tái sử dụng mã nguồn:** Các chức năng chung như kết nối cơ sở dữ liệu được tập trung trong `DatabaseConnector`, giúp giảm sự trùng lặp.
- **Dễ bảo trì:** Việc chia gói và lớp rõ ràng giúp dễ dàng xác định và sửa lỗi khi cần.
- **Tính tổ chức cao:** Cấu trúc rõ ràng của gói và sự phân chia trách nhiệm giúp lập trình viên dễ dàng hiểu và làm việc với hệ thống.

Thiết kế gói này đảm bảo sự hiệu quả trong việc xử lý dữ liệu, đồng thời tạo nền tảng vững chắc để tích hợp và mở rộng trong tương lai.

5.3 Quy trình xử lý dữ liệu

Quy trình xử lý dữ liệu bao gồm các bước chính sau:

1. **Khởi tạo cơ sở dữ liệu**: Tệp `schema.sql` được sử dụng để thiết lập cấu trúc cơ sở dữ liệu ban đầu thông qua lớp `DatabaseInitializer`.
2. **Nhập dữ liệu từ tệp JSON**:
   - `FollowerImporter`: Đọc tệp `followers.json`, lấy danh sách người theo dõi và cập nhật vào bảng `User_Follows`.
   - `FollowingImporter`: Đọc tệp `following.json`, lấy danh sách người đang theo dõi và cập nhật vào bảng `User_Follows`.
   - `RetweetImporter`: Đọc tệp `retweetData.json`, lấy danh sách retweeters và mentions, cập nhật vào bảng `User_Tweets`.
3. **Cập nhật dữ liệu**: Sau khi nhập dữ liệu, các bảng như `Users` và `Tweets` được cập nhật để phản ánh thông tin mới.

5.4 Kết quả xử lý dữ liệu

Kết quả của quá trình xử lý dữ liệu được trình bày trong Bảng 3.

Danh mục	Mô tả
2180 người dùng	Được lưu trữ trong bảng `Users`, bao gồm thông tin về followers, following, và retweeters.
376 quan hệ theo dõi	Được lưu trữ trong bảng `User_Follows`, thể hiện kết nối giữa các người dùng.
88 bài tweet	Được lưu trữ trong bảng `Tweets`, với đầy đủ thông tin về mentions và retweets.
4742 loại tương tác	Bao gồm mentions, retweets, và replies, được lưu trong bảng `User_Tweets`.

Bảng 3: Kết quả xử lý dữ liệu

5.5 Ý nghĩa của dữ liệu

Dữ liệu sau khi xử lý đóng vai trò nền tảng cho các phân tích tiếp theo, cụ thể là:

- **Xây dựng đồ thị xã hội**: Các bảng `Users`, `User_Follows`, và `User_Tweets` cung cấp thông tin để xây dựng đồ thị kết nối giữa người dùng và tweet.
- **Phân tích mức độ ảnh hưởng**: Dữ liệu được chuẩn hóa hỗ trợ thuật toán PageRank tính toán mức độ ảnh hưởng của từng người dùng trong cộng đồng blockchain.
- **Mở rộng ứng dụng**: Dữ liệu này có thể được sử dụng cho các phân tích khác như tìm kiếm các xu hướng (trending topics) và các KOL mới.

6 PageRank

Thuật toán PageRank là một phương pháp đánh giá mức độ ảnh hưởng của các nút trong đồ thị dựa trên cấu trúc liên kết giữa chúng. Trong dự án này, PageRank được sử dụng để xác định mức độ ảnh hưởng của người dùng trên mạng xã hội dựa trên các kết nối như người theo dõi (followers), người được theo dõi (following), và tương tác qua bài viết.

Quá trình tính toán PageRank được triển khai bằng ngôn ngữ Java, sử dụng các gói và lớp được thiết kế rõ ràng, đảm bảo khả năng mở rộng, hiệu quả, và dễ bảo trì. Các kết quả tính toán được lưu trữ dưới dạng tệp JSON để sử dụng trong các phân tích sau.

6.1 Công cụ và công nghệ sử dụng

Các công cụ và thư viện chính được sử dụng trong việc tính toán PageRank bao gồm:

Java Collections Framework: Java Collections Framework cung cấp các cấu trúc dữ liệu mạnh mẽ như `HashMap`, `List`, và `Set` để lưu trữ và xử lý dữ liệu đồ thị, đảm bảo hiệu suất tính toán PageRank ổn định và linh hoạt.

Jackson ObjectMapper: Jackson ObjectMapper được sử dụng để đọc và ghi dữ liệu các tệp JSON. Việc sử dụng thư viện này đảm bảo tính chính xác trong việc lưu trữ dữ liệu kết quả PageRank.

Apache Log4j: Apache Log4j hỗ trợ ghi nhật ký các sự kiện trong quá trình xử lý thuật toán PageRank, giúp theo dõi và khắc phục lỗi dễ dàng trong quá trình phát triển.

SQL: SQL được sử dụng để lưu trữ dữ liệu người dùng, bài đăng, và các tương tác giữa chúng. Kết quả tính toán PageRank cuối cùng cũng có thể được lưu trong cơ sở dữ liệu để phục vụ các phân tích mở rộng.

6.2 Ý tưởng thành lập các loại cạnh

Twitter là một mạng xã hội quy mô lớn với hàng triệu người dùng, nơi diễn ra nhiều loại tương tác khác nhau. Sau quá trình khảo sát thực tế, chúng em nhận diện được các loại tương tác tiêu biểu như: `FOLLOW`, `RETWEET` (`REPOST`), `POST`, `MENTION`, `LIKE`, `QUOTE`, `REPLY`, `BOOKMARK`, `VIEW`, `SHARE`,... Dựa trên quá trình phân tích kỹ lưỡng, chúng em đánh giá như sau:

LIKE : Thường ta sẽ nghĩ Twitter cũng giống FaceBook, họ sẽ cho xem những ai đã `LIKE` bài viết. Nhưng không, ở đây họ không cho xem những ai đã thực hiện loại tương tác này. Nhưng nếu ta dùng Twitter API, ta vẫn có thể lấy thông tin về người `LIKE` bài viết. Nhưng điều này không có ý nghĩa trong thuật toán PageRank, một cạnh xuất hiện giữa node A đến node B thì tức là phải có liên kết để khi người dùng đứng ở vị trí của node A có thể nhấp vào liên kết đó để đi đến được node B, mà Twitter đã ẩn đi thông tin này, vậy lấy được dữ liệu đấy, nhưng nó không có ý nghĩa! Chúng em không chọn loại tương tác này trong đồ thị của mình.

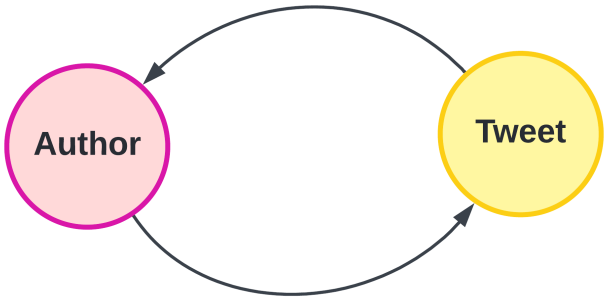
VIEW : Tương tự `LIKE`, Twitter chỉ cung cấp tổng số lượt xem một bài viết nhưng không tiết lộ danh tính người xem. Do đó, loại tương tác này không phù hợp để sử dụng.

BOOKMARK : Loại tương tác này sẽ đánh dấu một bài viết để người dùng có thể xem lại nó sau, hoặc nó đóng vai trò như là một lời nhắc nhở, ... Nhưng rất ít khi người dùng mạng xã hội sử dụng loại tương tác này, xác suất họ nhấp vào liên kết để xem lại là rất thấp. Nhiều khi họ xem lại họ cũng không biết họ đánh dấu để làm gì. Nói chung là loại tương tác này có ý nghĩa không cao. Chúng em quyết định không sử dụng loại tương tác này.

SHARE : `SHARE` có khả năng gia tăng đáng kể phạm vi tiếp cận của bài viết. Tuy nhiên, hầu hết người dùng chia sẻ bài viết ra ngoài Twitter, làm mất liên kết nội bộ. Do đó, tương tác này không được chọn.

Những loại tương tác còn lại phù hợp với đồ thị PageRank của chúng em bao gồm: `POST`, `FOLLOW`, `RETWEET`, `REPLY`, `QUOTE`, `MENTION`.

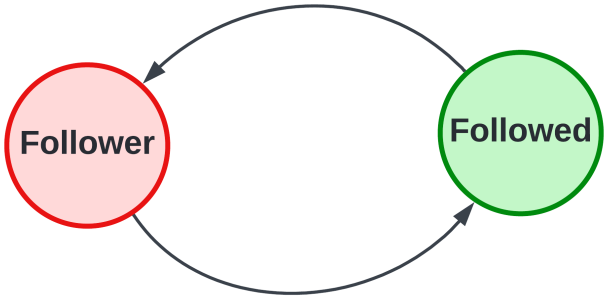
6.2.1 POST



Hình 4: Liên kết hình thành từ `POST`

Trong loại tương tác `POST` trên Twitter, khi người dùng tạo một bài đăng mới, hệ thống ghi nhận người dùng (`userID`) là tác giả của bài đăng (`tweetID`), tạo ra hai liên kết có hướng trong đồ thị mạng xã hội. Liên kết thứ nhất đi từ người dùng đến bài đăng, thể hiện rằng người dùng đã tạo bài đăng này. Liên kết thứ hai đi từ bài đăng trở lại người dùng, chỉ ra rằng bài đăng thuộc về người dùng đó. Cả hai liên kết này được lưu trữ trong bảng `Tweets`, nơi các trường như `tweetID`, `userID`, `content` và `createdAt` ghi nhận chi tiết về bài đăng và tác giả của nó. Khi duyệt Twitter, bạn có thể kiểm chứng các liên kết này thông qua trang cá nhân người dùng (hiển thị các bài đăng mà họ đã tạo) và trang bài đăng cụ thể (hiển thị tác giả của bài đăng). Những liên kết này không chỉ giúp quản lý dữ liệu trên nền tảng mà còn đóng vai trò quan trọng trong các thuật toán phân tích và xếp hạng, như PageRank hoặc MultiRelational PageRank, để xác định mức độ ảnh hưởng và độ tin cậy của người dùng và nội dung trong mạng xã hội.

6.2.2 FOLLOW



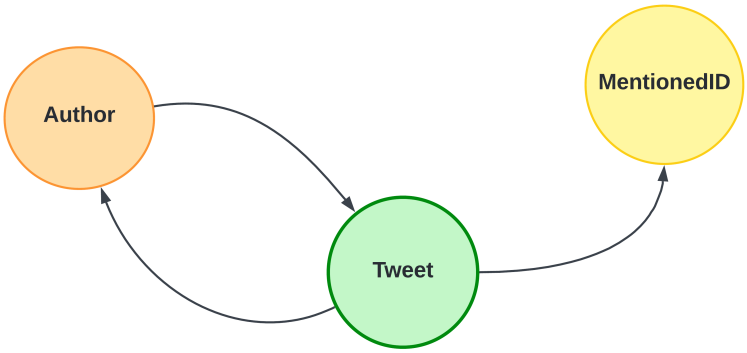
Hình 5: Liên kết hình thành từ `FOLLOW`

Trong loại tương tác `FOLLOW` trên `Twitter`, khi một người dùng (`followerID`) theo dõi người dùng khác (`followedID`), hệ thống ghi nhận hai liên kết có hướng trong đồ thị mạng xã hội. Liên kết thứ nhất xuất phát từ `followerID` đến `followedID`, thể hiện rằng người dùng thứ nhất đã thực hiện hành động theo dõi người dùng thứ hai. Ngược lại, liên kết thứ hai xuất phát từ `followedID` đến `followerID`, đại diện cho việc người dùng thứ hai có thêm một người theo dõi.

Các liên kết này được lưu trữ trong bảng `User_Follows`, với các trường như `followerID`, `followedID` và `followTime`, ghi nhận thông tin chi tiết về mối quan hệ theo dõi giữa hai người dùng. Khi sử dụng `Twitter`, bạn có thể kiểm chứng các liên kết này bằng cách truy cập vào danh sách người theo dõi của một người dùng bất kỳ hoặc danh sách những người mà họ đang theo dõi.

Những liên kết này không chỉ giúp hệ thống quản lý mối quan hệ giữa các người dùng mà còn đóng vai trò quan trọng trong các thuật toán phân tích mạng xã hội PageRank để đánh giá mức độ ảnh hưởng và sự kết nối giữa các người dùng trong nền tảng mạng xã hội.

6.2.3 MENTION



Hình 6: Liên kết hình thành từ `MENTION`

Trong loại tương tác `MENTION` trên `Twitter`, khi một người dùng đề cập đến người khác trong một bài đăng, ba liên kết có hướng được hình thành trong đồ thị mạng xã hội.

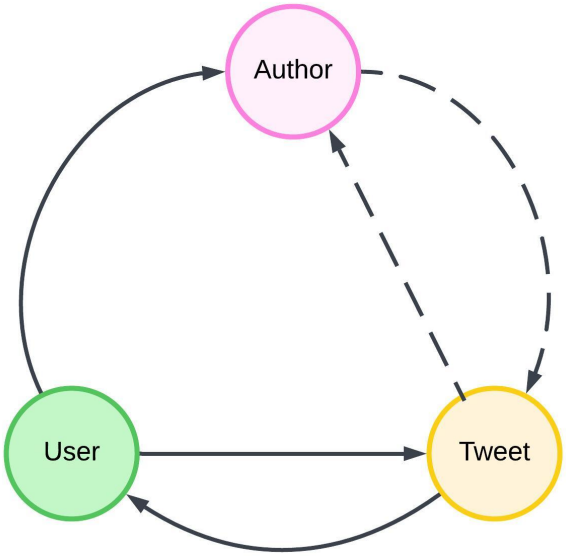
Liên kết thứ nhất: Xuất phát từ người đăng bài (`userID`) đến bài đăng (`tweetID`) của họ, thể hiện rằng người dùng này đã tạo ra nội dung.

Liên kết thứ hai: Ngược lại, từ bài đăng (`tweetID`) trở về người đăng bài (`userID`), thể hiện mối quan hệ sở hữu giữa bài đăng và người dùng. Hai liên kết này là đặc điểm chung của loại tương tác `POST`, mô tả việc một người dùng tạo nội dung trên nền tảng.

Liên kết thứ ba: Xuất phát từ bài đăng (`tweetID`) đến người được đề cập (`mentionedUserID`), thể hiện rằng bài đăng này có nhắc đến người dùng khác. Khi nhấp vào liên kết, hệ thống dẫn đến hồ sơ của người được đề cập, phản ánh sự kết nối thông qua tính năng `MENTION`.

Vì hai liên kết đầu tiên của `MENTION` hoàn toàn giống với loại tương tác `POST`, việc để lớp `MENTION` kế thừa lớp `POST` là hợp lý và giúp giảm thiểu việc lặp lại mã nguồn. Điều này cũng đảm bảo rằng mọi thuộc tính và phương thức liên quan đến bài đăng sẽ được áp dụng một cách nhất quán khi mở rộng thêm các tính năng tương tác khác.

6.2.4 RETWEET



Hình 7: Liên kết hình thành từ `RETWEET`

Loại tương tác `RETWEET` trên `Twitter` không tạo ra bài đăng mới mà chỉ chia sẻ lại nội dung đã có. Khi một người dùng thực hiện hành động `RETWEET`, hai liên kết có hướng được hình thành trong đồ thị mạng xã hội.

Liên kết thứ nhất: Từ người dùng đã đăng lại bài viết (`userID`) đến bài đăng gốc (`tweetID`) và tiếp tục đến tác giả của bài đăng (`authorOrMentionedID`). Điều này cho phép người khác truy cập bài đăng gốc và hồ sơ của tác giả khi nhấp vào bài `RETWEET`.

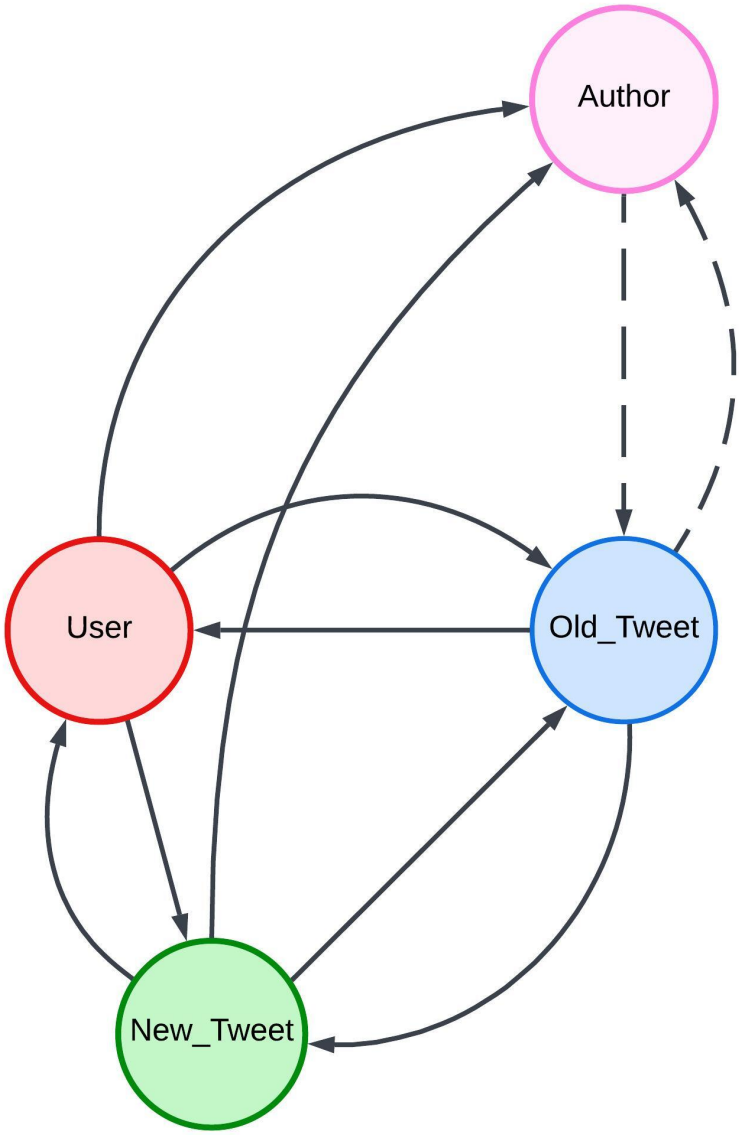
Liên kết thứ hai: Ngược lại, từ bài đăng gốc (`tweetID`) đến người đã đăng lại bài viết (`userID`). Liên kết này phản ánh rằng bài viết đã được chia sẻ và hiển thị trong danh sách những người đã `RETWEET` bài đăng, thông tin này hoàn toàn công khai.

Liên kết thứ ba là từ người dùng đến tác giả gốc Liên kết từ `userID` đến `authorOrMentionedID` thể hiện rằng người dùng không chỉ chia sẻ bài đăng mà còn tạo kết nối trực tiếp với tác giả của bài viết, phản ánh sự tương tác tiềm năng giữa hai người.

Tuy nhiên, vì danh sách những người `RETWEET` thường không hiển thị đầy đủ và chỉ cung cấp số lượng hạn chế khi xem thủ công, chúng em đánh giá trọng số của liên kết này không cao trong việc xác định mức độ ảnh hưởng và kết nối giữa các thực thể trong đồ thị mạng xã hội.

6.2.5 REPLY và QUOTE

Đây là 2 loại tương tác phức tạp nhất và chúng tạo ra các cạnh tương tự nhau. Khi người dùng thực hiện các hành động như `REPLY` hoặc `QUOTE` trên `Twitter`, hệ thống sẽ tạo một bài đăng mới, dẫn đến việc hình thành nhiều liên kết trong đồ thị mạng xã hội. Cụ thể, người dùng tạo liên kết từ `userID` đến bài đăng mới `tweetReplyQuoteID` và ngược lại, thể hiện người dùng



Hình 8: Liên kết hình thành từ `REPLY` và `QUOTE`

là tác giả của bài đăng. Bài đăng mới cũng liên kết hai chiều với bài đăng gốc `tweetID`, phản ánh rằng đó là một phản hồi hoặc trích dẫn từ nội dung ban đầu. Người dùng và bài đăng cũ cũng có liên kết đến nhau. Ngoài ra, cả người dùng và bài đăng mới đều liên kết đến tác giả gốc `authorOrMentionedID`, thể hiện sự tương tác trực tiếp với người đăng bài ban đầu. Hai liên kết giữa tác giả gốc và bài đăng gốc đã được lưu trước trong bảng `Tweets`, trong khi các liên kết còn lại được lưu trong bảng `User_Tweets`, giúp theo dõi chính xác và chi tiết các tương tác phức tạp trong hệ thống.

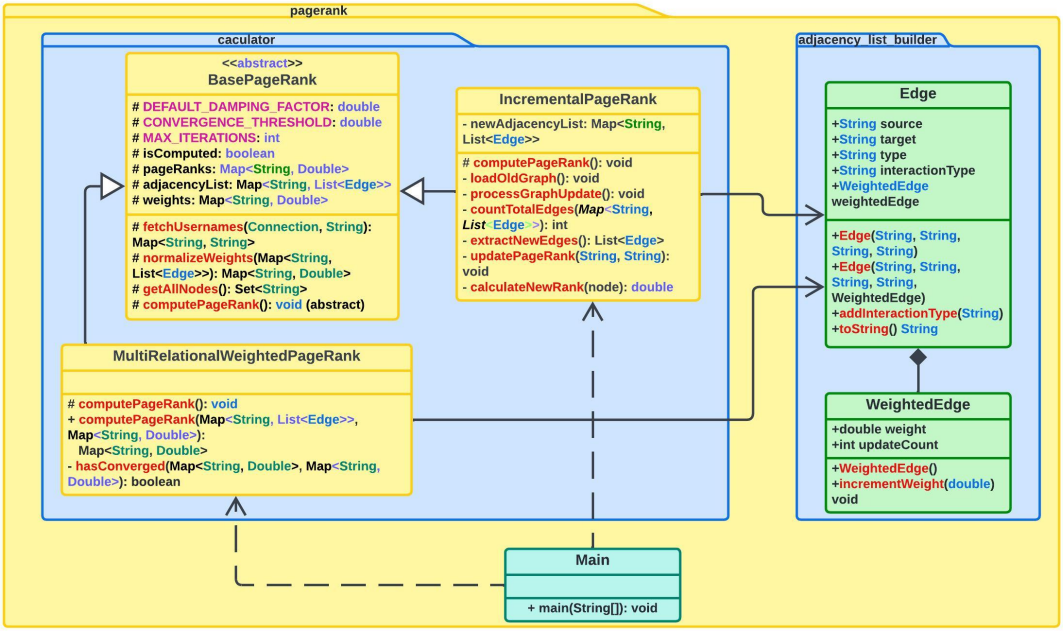
6.2.6 Cách hoạt động của bảng `User_Tweets`

Đây là bảng chứa nhiều trường thôn tin quan trọng nhất của dự án, cũng là bảng phức tạp nhất, bảng này sẽ lưu 4 loại tương tác: `RETWEET`, `MENTION`, `REPLY`, `QUOTE`. Chúng được hiểu như sau:

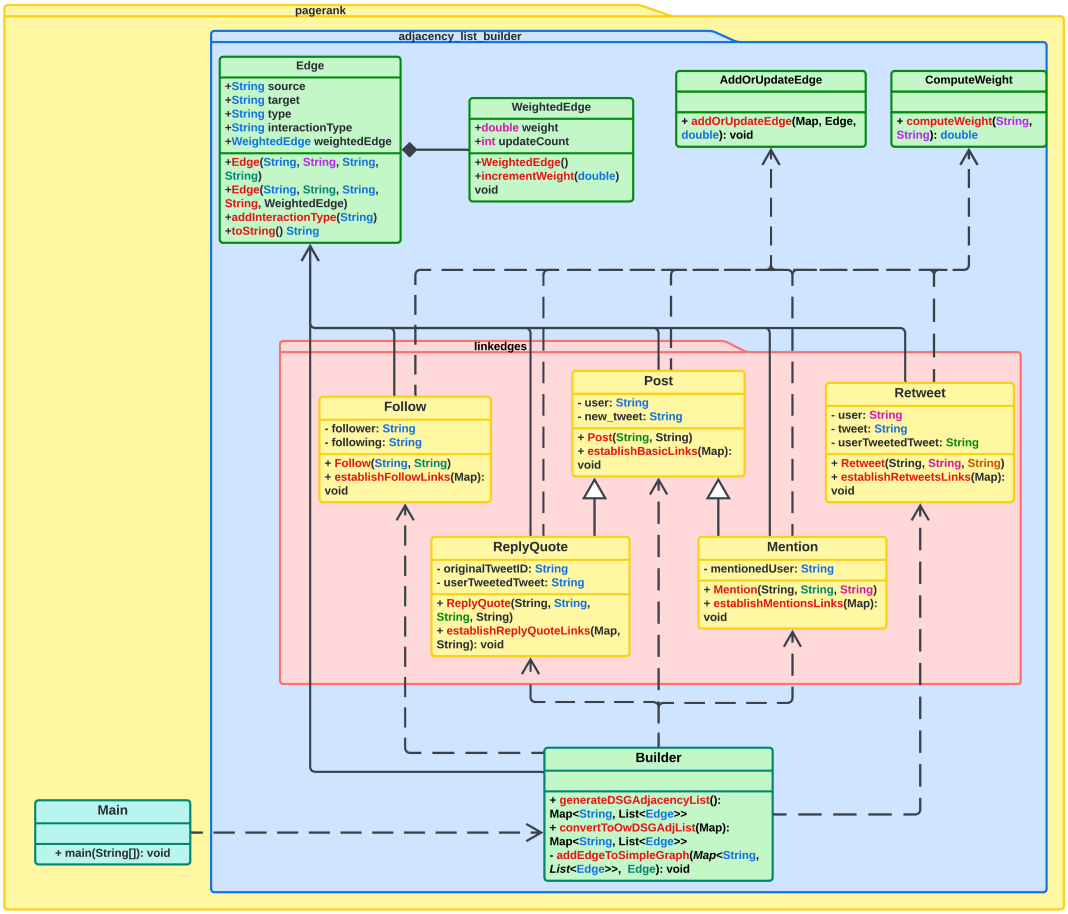
- `RETWEET`: `userID` đã đăng lại `tweetID` của tác giả `authorOrMentionedID`.
- `MENTION`: `userID` đã đề cập trong bài viết `tweetID` đến `authorOrMentionedID`.

- `REPLY/QUOTE`: `userID` đã trả lời/trích dẫn bài `tweetID` của tác giả `authorOrMentionedID` bằng bài viết mới `TweetReplyQuoteID`.

6.3 Thiết kế Gói và Lớp



Hình 9: Biểu đồ lớp của gói `caculator`



Hình 10: Biểu đồ lớp của gói `adjacency_list_builder`

6.3.1 Công dụng của các gói và lớp

Như thể hiện trong Hình 9 và Hình 10, các gói và lớp được thiết kế theo các nhiệm vụ chính như sau:

- `pagerank.calculator.BasePageRank`: Lớp trừu tượng định nghĩa các phương thức cơ bản để khởi tạo, tính toán và lưu kết quả PageRank. Lớp này cung cấp khung làm việc chung cho các thuật toán cụ thể.
- `pagerank.calculator.IncrementalPageRank`: Lớp kế thừa từ `BasePageRank`, triển khai thuật toán PageRank tăng dần nhằm tối ưu hóa việc cập nhật khi có thay đổi nhỏ trong đồ thị.
- `pagerank.calculator.MultiRelationalWeightedPageRank`: Lớp kế thừa từ `BasePageRank`, triển khai thuật toán PageRank có trọng số đa quan hệ cho đồ thị phức tạp với nhiều loại cạnh khác nhau.
- `pagerank.adjacency_list_builder.Builder`: Lớp chính chịu trách nhiệm xây dựng danh sách kề từ dữ liệu đầu vào. Nó tổng hợp các thông tin từ các cạnh khác nhau và ước lượng trọng số cho các loại quan hệ.
- `pagerank.adjacency_list_builder.AddOrUpdateEdge`: Lớp xử lý việc thêm hoặc cập nhật các cạnh vào danh sách kề, đảm bảo đồ thị luôn ở trạng thái chính xác nhất.
- `pagerank.adjacency_list_builder.ComputeWeight`: Lớp thực hiện tính toán trọng số cho các cạnh dựa trên các tiêu chí cụ thể như mức độ tương tác hoặc số lần xuất hiện.

- `pagerank.adjacency_list_builder.Edge`: Lớp mô tả một cạnh trong đồ thị, bao gồm các thuộc tính như nút nguồn, nút đích, loại cạnh, loại tương tác và trọng số.
- `pagerank.adjacency_list_builder.WeightedEdge`: Lớp mở rộng từ `Edge`, bổ sung các thuộc tính liên quan đến trọng số, hỗ trợ các thuật toán PageRank có trọng số.
- `pagerank.adjacency_list_builder.linkedges.Follow`: Lớp đại diện cho mối quan hệ người theo dõi giữa hai người dùng.
- `pagerank.adjacency_list_builder.linkedges.Mention`: Lớp mô tả mối quan hệ đề cập (mention) giữa người dùng và bài viết.
- `pagerank.adjacency_list_builder.linkedges.Post`: Lớp đại diện cho việc đăng bài của người dùng, liên kết người dùng với nội dung họ tạo ra.
- `pagerank.adjacency_list_builder.linkedges.ReplyQuote`: Lớp đại diện cho các mối quan hệ trả lời hoặc trích dẫn bài đăng giữa người dùng.
- `pagerank.adjacency_list_builder.linkedges.Retweet`: Lớp mô tả hành động đăng lại (retweet) của người dùng, tạo mối liên kết ngược trong đồ thị mạng xã hội.

6.3.2 Các kỹ thuật lập trình hướng đối tượng (OOP) được sử dụng

Trừu tượng hóa (Abstraction) Trừu tượng hóa được sử dụng để định nghĩa các khái niệm tổng quát mà không đi vào chi tiết cụ thể. Lớp `BasePageRank` là một ví dụ điển hình, định nghĩa các phương thức trừu tượng như `computePageRank`, và các phương thức chung như `fetchUsernames`, và `normalizeWeights`. Các lớp con như `IncrementalPageRank` và `MultiRelationalWeightedPageRank` kế thừa và triển khai các phương thức này với thuật toán cụ thể.

Kế thừa (Inheritance) Kế thừa cho phép mở rộng chức năng từ các lớp cơ sở. Lớp `IncrementalPageRank` và `MultiRelationalWeightedPageRank` kế thừa từ `BasePageRank`, giúp tránh lặp lại mã nguồn.

Đa hình (Polymorphism) Đa hình cho phép sử dụng cùng một giao diện nhưng với các cách triển khai khác nhau. Trong hệ thống này, các lớp kế thừa từ `BasePageRank` có thể được xử lý đồng nhất thông qua các phương thức trừu tượng. Ví dụ, hàm tính PageRank có thể gọi chung cho cả `IncrementalPageRank` và `MultiRelationalWeightedPageRank` mà không cần quan tâm đến chi tiết bên trong.

Đóng gói (Encapsulation) Các thuộc tính và phương thức của các lớp được đóng gói để bảo vệ dữ liệu bên trong. Chẳng hạn, các thuộc tính như `originalTweetID`, `userTweetedTweet` trong lớp `ReplyQuote` được khai báo là `private`, chỉ có thể truy cập thông qua các phương thức công khai như `establishReplyQuoteLinks`. Điều này đảm bảo rằng dữ liệu không bị thay đổi một cách tùy tiện từ bên ngoài.

Tái sử dụng mã nguồn (Code Reusability) Thông qua việc kế thừa và sử dụng các lớp con, hệ thống tối ưu hóa việc tái sử dụng mã nguồn. Lớp `Builder` tận dụng các lớp liên quan đến đồ thị như `Edge`, `WeightedEdge` và `ComputeWeight` để xây dựng đồ thị mà không phải viết lại logic tính toán.

Những kỹ thuật lập trình hướng đối tượng này giúp tăng cường tính linh hoạt, dễ bảo trì và mở rộng của hệ thống PageRank, đảm bảo rằng hệ thống có thể xử lý các yêu cầu phức tạp một cách hiệu quả và dễ hiểu.

Thiết kế này tuân thủ các nguyên tắc lập trình hướng đối tượng, đảm bảo khả năng mở rộng và dễ dàng bảo trì khi cần nâng cấp hệ thống.

6.4 Quy trình tính toán

Quy trình tính toán PageRank bao gồm các bước chính sau:

1. **Khởi tạo đồ thị**

   Sau khi đẩy dữ liệu lên cơ sở dữ liệu, hai phương thức `generateDSGAdjacencyList()` và `convertToOwDSGAdjList(Map<String, List<Edge»)` trong lớp `Builder` sẽ được gọi để xây dựng danh sách kề cho đồ thị. Phương thức thứ hai nhằm tạo danh sách kề phục vụ cho thuật toán PageRank cổ điển, nhưng thuật toán này đã bị loại bỏ vì không phản ánh đầy đủ tính chất của đồ thị mạng xã hội.

   Ước lượng Trọng số Các Loại Tương tác Trong quá trình xây dựng danh sách kề, với mỗi loại tương tác, chúng em ước lượng trọng số cho từng loại cạnh bằng cách sử dụng phương thức `computeWeight()`.

   Phương thức `computeWeight()` Phương thức này nhận vào ba tham số chính:

   - `type`: Loại cạnh (edge), ví dụ: `User -> User`, `User -> Tweet`, `Tweet -> Tweet`, hoặc `Tweet -> User`.
   - `interactionType`: Loại tương tác giữa các nút, ví dụ: `FOLLOW+`, `REPLY`, `RETWEET`, v.v.
   - `interactionTime`: Thời gian của sự kiện tương tác (ở định dạng `DATETIME`).

   Dựa trên loại cạnh và loại tương tác, phương thức tính toán một trọng số cơ bản (`baseWeight`) cho mỗi tương tác. Ví dụ, với loại tương tác `FOLLOW+`, trọng số sẽ là 7.0, trong khi `RETWEET+` sẽ có trọng số là 4.0. Các trọng số này được xác định dựa trên mức độ ảnh hưởng của từng loại tương tác.

   Hàm Phân rã Thời gian Không phải tất cả các tương tác đều có giá trị trọng số không đổi. Để mô phỏng sự suy giảm ảnh hưởng theo thời gian (time decay), phương thức `computeWeight()` sử dụng một hàm phân rã.

   Cụ thể, thời gian ảnh hưởng của các tương tác sẽ giảm dần theo thời gian, và yếu tố này được điều chỉnh qua một hằng số phân rã λ.

   Phương thức tính toán sự thay đổi của trọng số theo thời gian bằng công thức phân rã:

   $$\text{decayFactor} = e^{-\lambda \times \Delta t}$$

   Trong đó:

   - λ là hằng số phân rã thời gian, có giá trị mặc định là 0.00001, nhưng có thể điều chỉnh tuỳ theo yêu cầu của mô hình.
   - Δt là sự chênh lệch thời gian giữa thời điểm của tương tác và thời điểm hiện tại (tính bằng phút).

Hàm phân rã đảm bảo rằng các tương tác cũ sẽ có ảnh hưởng nhỏ hơn đến trọng số tổng quát, giúp mô phỏng chính xác hơn sự thay đổi trong ảnh hưởng của các tương tác theo thời gian.

Cuối cùng, trọng số được tính toán chính là trọng số cơ bản đã được điều chỉnh bởi yếu tố phân rã thời gian. Điều này giúp giảm ảnh hưởng của những tương tác cũ và làm nổi bật các tương tác gần đây hơn.

2. **Tính toán PageRank**

   Lần lượt gọi hai thuật toán:

   (a) `Incremental PageRank`

   (b) `Multi-Relational Weighted PageRank`

   Trong thuật toán `Incremental PageRank`, có hai đầu vào:

   - Danh sách kề đọc từ tệp JSON.
   - Dữ liệu từ cơ sở dữ liệu.

   Nếu số lượng cạnh mới từ cơ sở dữ liệu nhiều hơn số lượng cạnh từ danh sách kề cũ 10%, thuật toán `Incremental PageRank` sẽ được triển khai; ngược lại, nếu số lượng cạnh lớn hơn 10%, thuật toán `Multi-Relational Weighted PageRank` sẽ được áp dụng.

Phương trình PageRank có trọng số: Công thức chuẩn của PageRank có trọng số được biểu diễn như sau:

$$PR(i) = \frac{1-d}{N} + d \cdot \sum_{j \in \text{In}(i)} \frac{w_{ji} \cdot PR(j)}{\sum_{k \in \text{Out}(j)} w_{jk}}$$

Trong đó:

- $PR(i)$: Giá trị PageRank của nút i.
- d: Hệ số suy giảm (damping factor), thường được đặt là $d = 0.85$.
- N: Tổng số nút trong đồ thị.
- $\text{In}(i)$: Tập hợp các nút có liên kết đến nút i.
- $\text{Out}(j)$: Tập hợp các nút mà nút j liên kết đến.
- w_{ji}: Trọng số của cạnh từ nút j đến nút i.
- $\sum_{k \in \text{Out}(j)} w_{jk}$: Tổng trọng số của các cạnh đi ra từ nút j.

Giải thích: Thành phần ngẫu nhiên:

$$\frac{1-d}{N}$$

Biểu thức này đại diện cho xác suất người dùng nhảy đến một nút ngẫu nhiên bất kỳ trong đồ thị khi không có liên kết trực tiếp.

Thành phần liên kết:

$$d \cdot \sum_{j \in \text{In}(i)} \frac{w_{ji} \cdot PR(j)}{\sum_{k \in \text{Out}(j)} w_{jk}}$$

Thành phần này thể hiện sự lan truyền điểm PageRank từ các nút liên kết đến nút i, có xét trọng số của các liên kết.

Điều kiện hội tụ: Quá trình tính toán tiếp tục lặp lại cho đến khi độ chênh lệch giữa hai lần tính liên tiếp nhỏ hơn ngưỡng xác định ($\epsilon = 10^{-6}$).

Xử lý nút treo (Dangling Nodes): Nút treo là các nút không có cạnh đi ra, tức là tập $\text{Out}(j) = \emptyset$. Khi gặp nút treo, không có cách nào để lan truyền giá trị PageRank của nút đó đến các nút khác, dẫn đến thất thoát điểm PageRank trong đồ thị. Để giải quyết vấn đề này, khi gặp một nút treo j, giá trị PageRank của nút đó được phân phối đều đến tất cả các nút khác trong đồ thị. Điều này tương đương với việc giả định nút treo liên kết đến tất cả các nút khác, bao gồm chính nó. Công thức được điều chỉnh như sau:

$$PR(i) = \frac{1-d}{N} + d \cdot \left(\sum_{j \in \text{In}(i)} \frac{w_{ji} \cdot PR(j)}{\sum_{k \in \text{Out}(j)} w_{jk}} + \frac{\sum_{j \in D} PR(j)}{N} \right)$$

Thành Phần Từ Nút Treo

$$\frac{\sum_{j \in D} PR(j)}{N}$$

Đây là phần điểm PageRank được phân phối đều từ tất cả các nút treo $j \in D$ đến tất cả các nút trong đồ thị, bao gồm chính nó.

Chuẩn hóa trọng số đầu vào: Do đầu vào có thể chứa các cạnh với trọng số rất cao, gây ảnh hưởng đến kết quả tính toán, chúng em đã chuẩn hóa trọng số bằng cách sử dụng phương thức `normalizeWeights(Map<String, List<Edge»)` trước khi thực hiện tính toán PageRank. Quy tắc chuẩn hóa là tổng trọng số của tất cả các cạnh đi ra của một nút luôn bằng 1.

3. **Lưu kết quả** Kết quả tính toán PageRank được ghi vào tệp JSON hoặc cơ sở dữ liệu SQL để phục vụ cho các phân tích khác.

4. **Cập nhật PageRank** Khi có thay đổi nhỏ trong đồ thị, thuật toán `Incremental PageRank` sẽ được áp dụng để tối ưu hóa thời gian tính toán, thay vì thực hiện lại toàn bộ quy trình.

6.5 Kết quả

Kết quả tính toán PageRank có dạng như trong Bảng 4:

Bảng 4: Bảng kết quả PageRank tính toán được

Node	PageRank
T138274	0.0039495
@crypto	0.1743023
@ice_blockchain	0.3843527
...	...

Kết quả trên cho thấy các tài khoản hoặc bài viết có ảnh hưởng lớn trên mạng xã hội sẽ có giá trị PageRank cao hơn, thể hiện tầm quan trọng của chúng trong hệ thống. Điều này cho thấy rằng thuật toán PageRank được áp dụng đã phản ánh chính xác các tương tác và sự ảnh hưởng trong đồ thị mạng xã hội mà chúng tôi xây dựng.

6.6 Ý nghĩa kết quả nhận được

Các kết quả PageRank thu được có ý nghĩa quan trọng trong việc đánh giá mức độ ảnh hưởng của người dùng và bài viết trên mạng xã hội:

- **Phân tích ảnh hưởng cá nhân:** Người dùng có giá trị PageRank cao nhất được xác định là những người có vai trò quan trọng trong mạng lưới xã hội.
- **Xác định các KOL tiềm năng:** PageRank hỗ trợ xác định các cá nhân có tiềm năng trở thành người dẫn đầu dư luận (Key Opinion Leaders - KOLs) thông qua mức độ kết nối rộng rãi.
- **Phân tích chiến lược tiếp thị:** Kết quả PageRank có thể giúp các tổ chức tối ưu hóa chiến dịch tiếp thị, tập trung vào những người có tầm ảnh hưởng lớn nhất trong cộng đồng.
- **Ứng dụng trong các hệ thống khác:** Thuật toán PageRank không chỉ áp dụng trong mạng xã hội mà còn hữu ích trong các lĩnh vực như tìm kiếm web, đề xuất sản phẩm, và phân tích mạng lưới phức tạp.

7 Visualization

Visualization là bước hiển thị dữ liệu sau khi đã được xử lý và tính toán bằng thuật toán PageRank. Mục tiêu chính của bước này là cung cấp một giao diện trực quan để người dùng có thể dễ dàng phân tích và tương tác với dữ liệu đồ thị.

Hệ thống được chia thành hai thành phần chính:

- Backend sử dụng Spring Boot để xây dựng API cung cấp dữ liệu đồ thị.
- Frontend sử dụng HTML, JavaScript và Cytoscape.js để hiển thị đồ thị tương tác.

7.1 Công cụ và công nghệ sử dụng

Quá trình xây dựng Visualization sử dụng nhiều công cụ và công nghệ hiện đại, bao gồm cả backend và frontend.

Backend (Spring Boot)

Spring Boot: Spring Boot là một framework mạnh mẽ trong Java, hỗ trợ xây dựng các ứng dụng web nhanh chóng. Trong dự án này, Spring Boot được sử dụng để xây dựng REST API cung cấp dữ liệu đồ thị.

- Tích hợp sẵn máy chủ nhúng (Tomcat) giúp triển khai dễ dàng.
- Quản lý các thư viện phụ thuộc thông qua Maven.
- Tương thích tốt với các ứng dụng lớn nhờ kiến trúc module hóa.

REST API: REST API là phong cách thiết kế giao tiếp giữa frontend và backend. Trong hệ thống này, các API chính bao gồm:

- `/api/nodes`: Trả về danh sách các node trong đồ thị.
- `/api/edges`: Trả về danh sách các cạnh trong đồ thị.
- `/api/average-pagerank`: Trả về giá trị trung bình của PageRank.

Jackson: Thư viện Jackson được sử dụng để đọc và xử lý dữ liệu từ file `pageRankPoints.json`. Nó hỗ trợ chuyển đổi giữa JSON và đối tượng Java một cách nhanh chóng và hiệu quả.

Streams API: Java Streams API được sử dụng để xử lý dữ liệu một cách linh hoạt, như lọc các node có giá trị PageRank cao hơn trung bình.

Frontend (HTML, JavaScript)

HTML: HTML được sử dụng để xây dựng giao diện người dùng cơ bản. Giao diện bao gồm các thành phần:

- Dropdown để chọn bố trí biểu đồ.
- Phần hiển thị biểu đồ (`<div id="cy">`).
- Thông báo chi tiết khi hover hoặc click vào node.

Cytoscape.js: Cytoscape.js là một thư viện mạnh mẽ để vẽ và hiển thị đồ thị tương tác. Trong dự án này, Cytoscape.js được sử dụng để:

- Hiển thị đồ thị với các node và cạnh.
- Thay đổi kích thước và màu sắc node dựa trên giá trị PageRank.
- Hỗ trợ các layout khác nhau như `cose`, `grid`, `concentric`.

JavaScript: JavaScript chịu trách nhiệm xử lý logic phía frontend, bao gồm:

- Fetch dữ liệu từ backend qua API.
- Khởi tạo và quản lý Cytoscape.js.
- Xử lý sự kiện như hover và click vào node.

CSS: CSS được sử dụng để tùy chỉnh giao diện của ứng dụng, đảm bảo giao diện đẹp và dễ sử dụng.

7.2 Thiết kế gói và lớp

Thiết kế gói

Hệ thống được tổ chức thành các gói chính:

- `com.example.visualizer`: Gói gốc chứa toàn bộ mã nguồn backend.
- `VisualizerApplication`: Lớp khởi động ứng dụng Spring Boot.
- `GraphController`: Lớp quản lý các API REST.

Thiết kế lớp

VisualizerApplication: Lớp khởi động của ứng dụng, chứa phương thức `main()` để chạy ứng dụng Spring Boot.

GraphController: Lớp quản lý các API REST, xử lý các yêu cầu từ frontend và trả về dữ liệu cần thiết. Cấu trúc chính của lớp:

- `userData`: Lưu trữ dữ liệu đồ thị từ file JSON.
- `averagePageRank`: Giá trị trung bình của PageRank.
- API `/api/nodes`: Trả về danh sách các node.
- API `/api/edges`: Trả về danh sách các cạnh.
- API `/api/average-pagerank`: Trả về giá trị trung bình PageRank.

7.3 Quy trình hoạt động

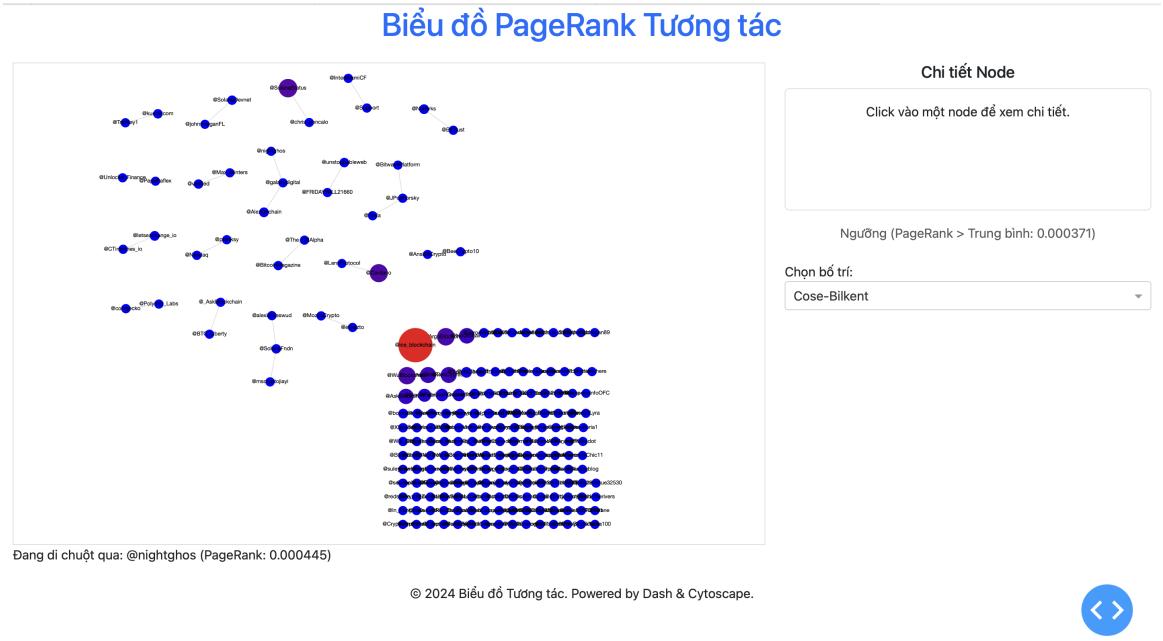
Quy trình Visualization bao gồm các bước:

1. **Xử lý dữ liệu backend:** Đọc dữ liệu từ file JSON, tính toán các giá trị như PageRank trung bình, và chuẩn bị dữ liệu để cung cấp qua API REST.
2. **Hiển thị frontend:** Fetch dữ liệu từ API backend, hiển thị đồ thị bằng Cytoscape.js, và xử lý các sự kiện người dùng như hover hoặc click.

7.4 Kết quả hiển thị

Hệ thống Visualization cung cấp các tính năng trực quan như:

- Hiển thị đồ thị tương tác với các node và cạnh đại diện cho các tài khoản Twitter và mối quan hệ của chúng.
- Tùy chỉnh giao diện đồ thị theo các layout khác nhau.
- Thông báo chi tiết khi người dùng tương tác với node, bao gồm giá trị PageRank.



Hình 11: Ví dụ hiển thị đồ thị tương tác bằng Cytoscape.js

7.5 Ý nghĩa của Visualization

Visualization giúp người dùng:

- Dễ dàng phân tích dữ liệu PageRank của các tài khoản Twitter.
- Hiểu được mối quan hệ giữa các tài khoản và tương tác của chúng.
- Hỗ trợ ra quyết định dựa trên các biểu đồ trực quan.

8 Quản lý cấu hình và tích hợp hệ thống

Phần quản lý cấu hình và tích hợp hệ thống đóng vai trò quan trọng trong việc đảm bảo hoạt động ổn định và hiệu quả của toàn bộ ứng dụng. Gói `others` được thiết kế để xử lý các tác vụ liên quan đến cấu hình, tiện ích hỗ trợ, và tích hợp hệ thống, giúp các thành phần khác của dự án vận hành mượt mà.

8.1 Công cụ và công nghệ sử dụng

Các công cụ và công nghệ chính được sử dụng trong phần này bao gồm:

Tess4J: Thư viện mã nguồn mở hỗ trợ OCR (Optical Character Recognition), sử dụng Tesseract để nhận dạng văn bản từ hình ảnh. Tess4J được tích hợp trong lớp `OCRUtils` để thực hiện trích xuất văn bản tự động. Mục đích sử dụng ở đây để phục vụ viêc giải Captcha khi đăng nhập Twitter.

Jackson ObjectMapper: Thư viện mạnh mẽ để đọc và ghi dữ liệu JSON. Nó được tích hợp trong lớp `FileUtils` để đọc tệp JSON và ánh xạ dữ liệu sang đối tượng Java.

Java Networking (HttpURLConnection): Java hỗ trợ kết nối mạng thông qua `HttpURLConnection`, được sử dụng trong lớp `NetworkUtils` để gửi và nhận yêu cầu HTTP.

Log4j: Thư viện ghi log, cung cấp khả năng ghi lại các thông báo, lỗi và thông tin xử lý trong hệ thống. Nó được sử dụng trong tất cả các lớp của gói `others.utils` để ghi lại trạng thái và hành vi của chương trình.

Các công cụ này phối hợp chặt chẽ để thực hiện các tác vụ liên quan đến quản lý tệp, xử lý mạng, và nhận dạng văn bản.

8.2 Thiết kế gói và lớp

Gói `others.utils` bao gồm ba lớp chính: `FileUtils`, `NetworkUtils`, và `OCRUtils`. Mỗi lớp đảm nhận các chức năng riêng biệt, phục vụ cho các nhu cầu khác nhau của dự án. Dưới đây là chi tiết từng lớp:

- `FileUtils`**:** Quản lý các tác vụ liên quan đến tệp, bao gồm:
  - Ghi dữ liệu vào tệp (`writeFile`).
  - Ghi đối tượng Java vào file JSON (`writeJsonToFile`).
  - Đọc tệp JSON và ánh xạ sang đối tượng Java (`readJsonFile`).
  - Kiểm tra sự tồn tại của tệp (`fileExists`).
  - Đảm bảo thư mục cha tồn tại (`ensureParentDirectoryExists`).
- `NetworkUtils`**:** Xử lý các tác vụ mạng như:
  - Gửi yêu cầu HTTP GET (`sendGetRequest`).
  - Gửi yêu cầu HTTP POST (`sendPostRequest`).
  - Xử lý lỗi mạng (`handleNetworkError`).
- `OCRUtils`**:** Nhận dạng văn bản từ hình ảnh, với phương thức:
  - `recognizeTextFromImage`: Trích xuất văn bản từ hình ảnh bằng thư viện Tess4J.

Các kỹ thuật lập trình hướng đối tượng (OOP) được áp dụng

Thiết kế gói `others.utils` áp dụng nhiều kỹ thuật OOP quan trọng, bao gồm:

1. Trừu tượng hóa (Abstraction): Các chức năng phổ biến như đọc/ghi tệp hoặc gửi yêu cầu HTTP được trừu tượng hóa thành các phương thức đơn giản, giúp lập trình viên dễ dàng sử dụng mà không cần quan tâm đến chi tiết bên trong.

2. Đóng gói (Encapsulation): Các thuộc tính và phương thức trong các lớp như `FileUtils`, `NetworkUtils` được đóng gói cẩn thận, chỉ cho phép truy cập qua các phương thức công khai.

3. Tái sử dụng mã nguồn: Nhờ việc tách biệt các chức năng thành từng lớp riêng biệt, các thành phần như `writeJsonToFile` hoặc `sendGetRequest` có thể được tái sử dụng ở bất kỳ đâu trong dự án.

8.3 Quy trình hoạt động

Dưới đây là quy trình sử dụng các tiện ích trong gói `others.utils`:

1. **Quản lý tệp:**
   - Sử dụng `writeJsonToFile` để lưu trữ dữ liệu đầu ra từ hệ thống dưới dạng tệp JSON.
   - Kiểm tra sự tồn tại của tệp với `fileExists` trước khi thao tác.
2. **Xử lý mạng:**
   - Gửi yêu cầu HTTP GET hoặc POST tới API bên ngoài để lấy hoặc gửi dữ liệu.
   - Xử lý lỗi mạng và ghi log các thông tin lỗi.
3. **Nhận dạng văn bản từ hình ảnh:**
   - Đọc hình ảnh đầu vào và trích xuất văn bản với `OCRUtils.recognizeTextFromImage`.

8.4 Kết quả

Các lớp trong gói `others.utils` đã được sử dụng hiệu quả trong toàn dự án để hỗ trợ các chức năng liên quan đến quản lý tệp, mạng, và OCR. Dưới đây là một số kết quả chính:

- **Xử lý tệp JSON:** Hàng nghìn tệp JSON đã được đọc, ghi và xử lý thành công, đảm bảo tính toàn vẹn của dữ liệu.
- **Yêu cầu mạng:** Các API bên ngoài được kết nối thành công, cung cấp dữ liệu cần thiết cho các giai đoạn tiếp theo.
- **Nhận dạng văn bản:** Các hình ảnh chứa văn bản quan trọng đã được xử lý để trích xuất dữ liệu, cải thiện tính tự động hóa của hệ thống.

8.5 Ý nghĩa

Gói `others.utils` đóng vai trò như một bộ công cụ hỗ trợ quan trọng, giúp giảm độ phức tạp của mã nguồn, nâng cao hiệu suất và đảm bảo tính ổn định của toàn bộ hệ thống. Các tiện ích này không chỉ giúp tăng năng suất phát triển mà còn mở ra khả năng mở rộng và tích hợp thêm các tính năng trong tương lai.

9 Hướng dẫn sử dụng

Dưới đây là các bước chi tiết để sử dụng dự án, từ thu thập dữ liệu, xử lý, đến tính toán PageRank:

9.1 Chuẩn bị môi trường

1. **Mở dự án trong Java IDE:**
   - Sử dụng IDE hỗ trợ Java, như IntelliJ IDEA hoặc Eclipse.
   - Tải và mở toàn bộ dự án trong IDE.
2. **Đảm bảo cài đặt đầy đủ các thư viện:**
   - Kiểm tra file `pom.xml` (nếu sử dụng Maven) hoặc `build.gradle` (nếu sử dụng Gradle) để đảm bảo tất cả các thư viện phụ thuộc đã được tải xuống và cài đặt đúng cách.
   - Các thư viện chính bao gồm: Selenium, Jackson, Log4j, PostgreSQL JDBC Driver, Cytoscape.js (nếu chạy phần Visualization).
3. **Cấu hình file `application.properties`:**
   - Điền đầy đủ thông tin kết nối đến cơ sở dữ liệu PostgreSQL, các đường dẫn tệp đầu vào/đầu ra, và thông tin API của Twitter.
   - File `application.properties` nằm trong thư mục `src/main/resources`.

9.2 Thu thập dữ liệu từ Twitter

1. **Chạy lớp Main trong gói `scraper`:**
   - Mở lớp `Main.java` trong gói `scraper`.
   - Chạy chương trình, quá trình này sẽ tự động:
     - Đăng nhập vào tài khoản Twitter bằng Selenium.
     - Tìm kiếm các tài khoản KOL liên quan đến lĩnh vực Blockchain dựa trên từ khóa và hashtag được cấu hình trước.
     - Thu thập dữ liệu followers, following và tweet IDs từ các tài khoản đã chọn.
   - Dữ liệu thu thập được sẽ được lưu trong thư mục `output/Data` dưới dạng các file JSON:
     - `followers.json`: Chứa danh sách followers của từng tài khoản.
     - `following.json`: Chứa danh sách following của từng tài khoản.
     - `tweetID.json`: Chứa danh sách ID của các bài viết (tweet).
2. **Lấy danh sách retweeters:**
   - Mở và chạy lớp `RetweetUserInfo.java` trong gói `scraper`.
   - Quá trình này sẽ:
     - Lấy danh sách người đã retweet (*retweeters*) cho từng tweet đã thu thập.
     - Lưu kết quả vào file `retweetData.json` trong thư mục `output/Data`.

9.3 Import dữ liệu và tính toán PageRank

1. **Chạy lớp Main trong gói `processor`:**
   - Mở lớp `Main.java` trong gói `processor`.
   - Chạy chương trình, quá trình này sẽ tự động:
     - Khởi tạo cơ sở dữ liệu bằng cách đọc file `schema.sql`.
     - Import dữ liệu từ các file JSON đã thu thập (`followers.json`, `following.json`, `retweetData.json`) vào cơ sở dữ liệu PostgreSQL.
     - Xây dựng đồ thị xã hội (Social Graph) từ dữ liệu trong cơ sở dữ liệu.
     - Tính toán điểm PageRank cho từng user bằng thuật toán PageRank.
2. **Kết quả đầu ra:**
   - Chương trình sẽ xuất kết quả vào thư mục `output`, với hai thư mục con:
     - `AdjList`: Chứa danh sách kề (*adjacency list*) của đồ thị được sử dụng trong thuật toán PageRank.
     - `PageRankPoints`: Chứa kết quả điểm PageRank của từng user.
   - Ví dụ:
     - File `directedSimpleGraph.json` lưu danh sách kề của đồ thị.
     - File `pageRankPoints.json` lưu điểm PageRank tương ứng với từng user.

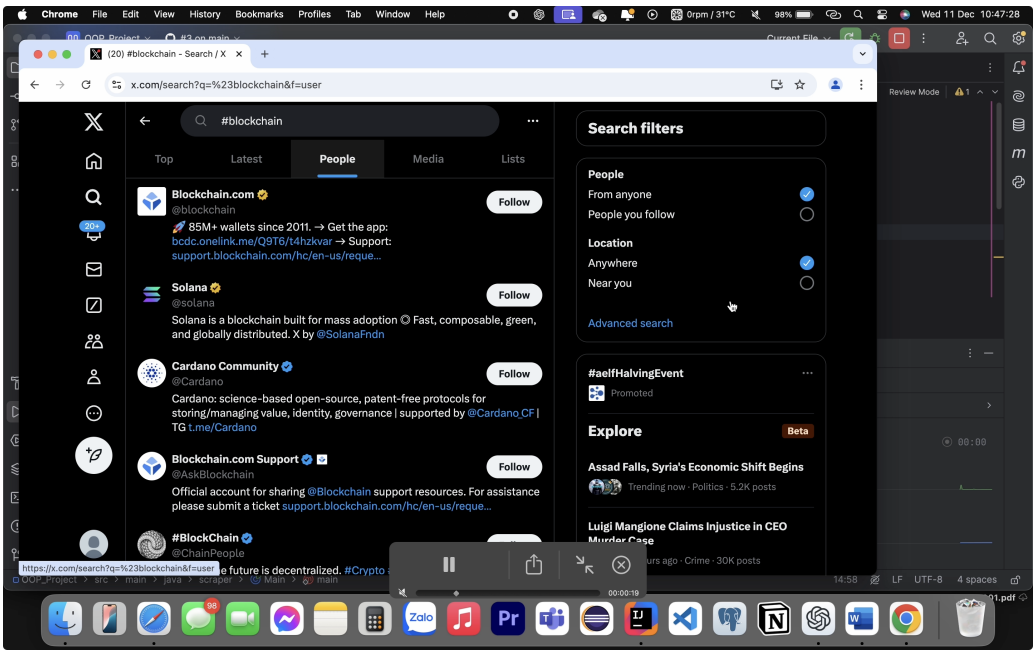
9.4 Hiển thị kết quả PageRank (tùy chọn)

Nếu cần hiển thị kết quả đồ thị PageRank, chạy gói file.pyvisualizer.

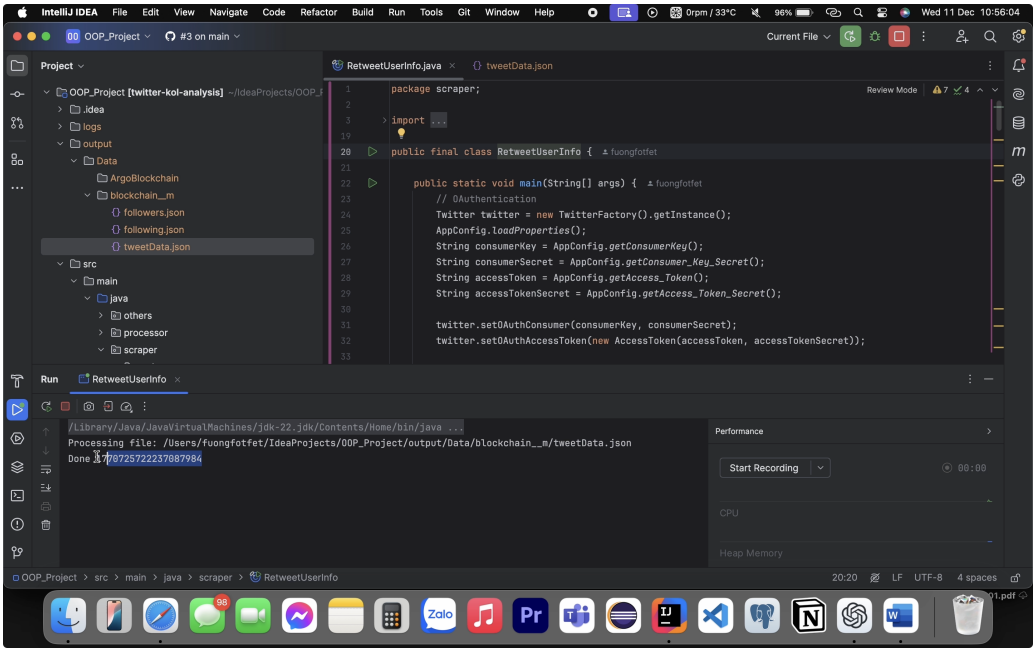
10 Ảnh demo sản phẩm

Phần này trình bày các ảnh minh họa cho các bước thực hiện và kết quả của sản phẩm. Các ảnh bao gồm giao diện thu thập dữ liệu, xử lý dữ liệu, kết quả tính toán PageRank, và hiển thị đồ thị trực quan.

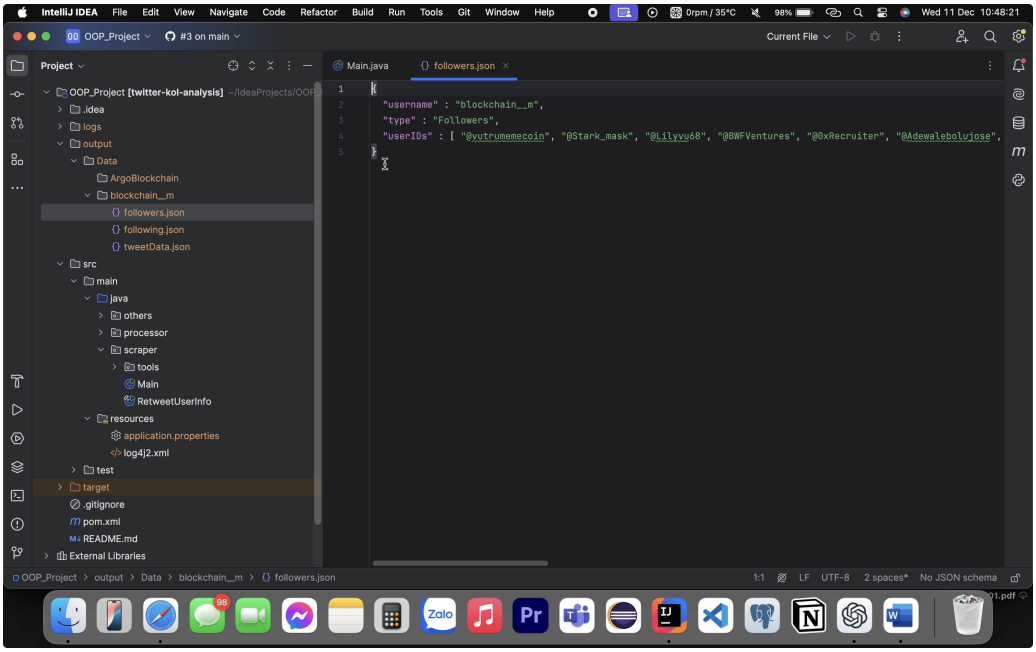
10.1 Giao diện thu thập dữ liệu



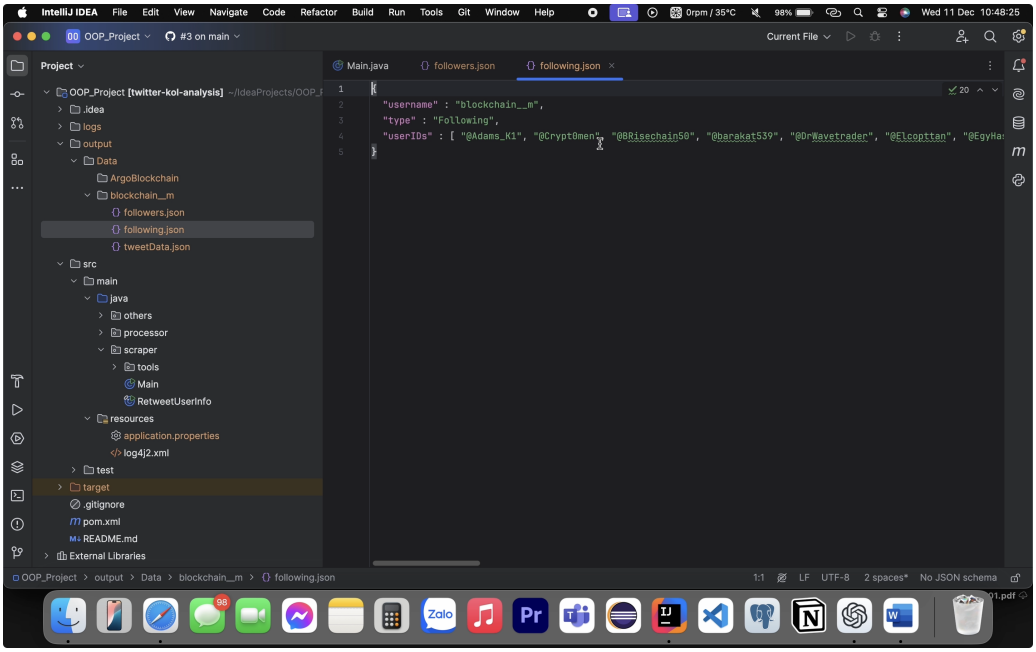
Hình 12: Giao diện Selenium tự động thu thập thông tin từ Twitter.



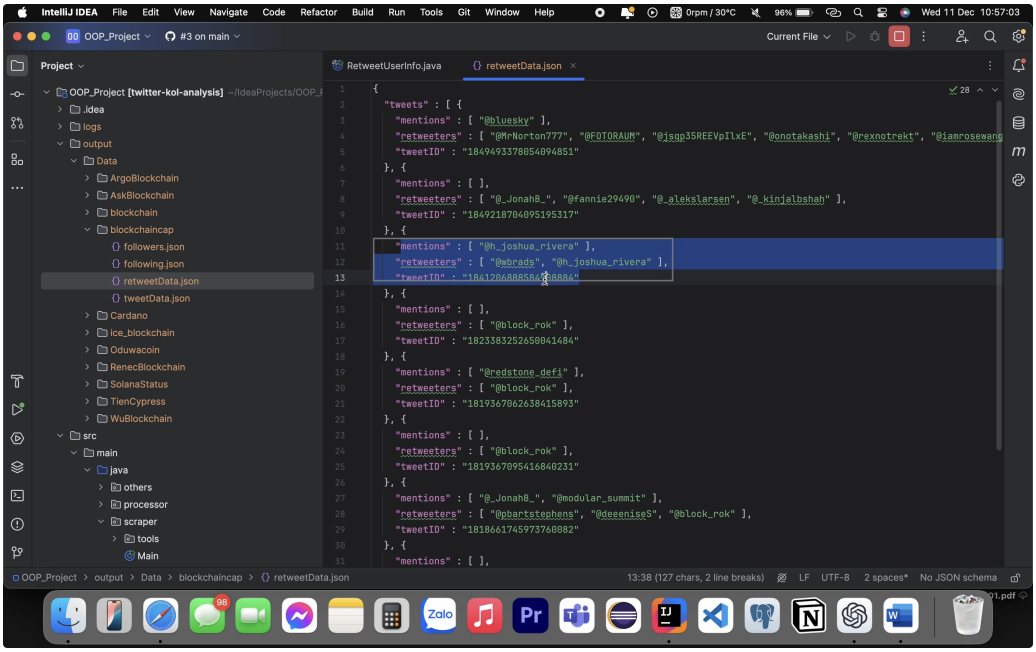
Hình 13: Sử dụng TwitterAPI kèm để lấy thông tin retweeters.



Hình 14: File JSON chứa danh sách người theo dõi (`followers.json`).

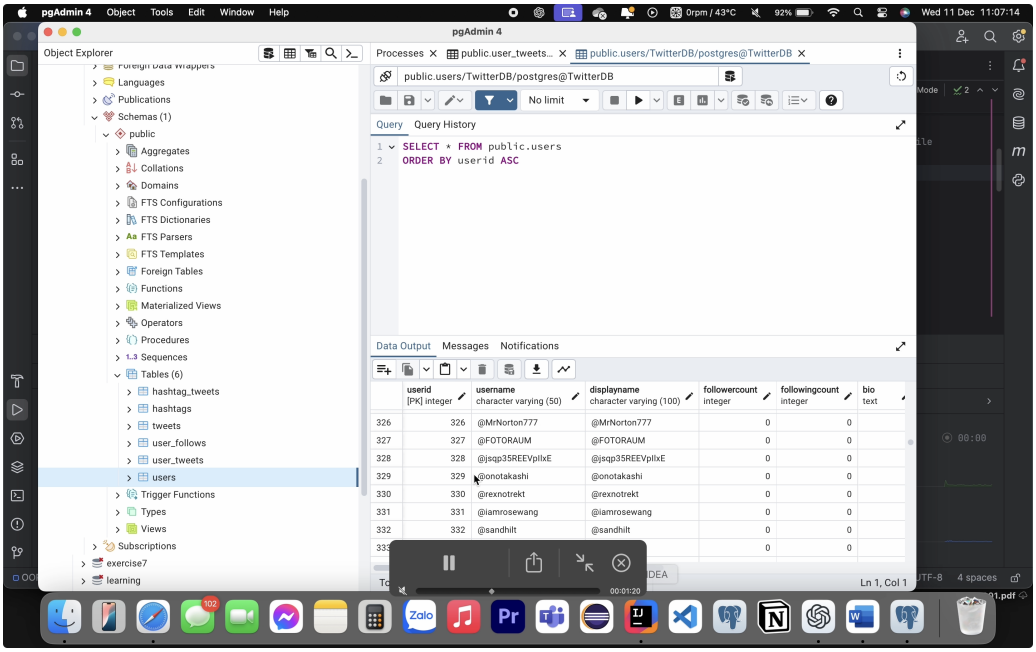


Hình 15: File JSON chứa danh sách người đang theo dõi bởi người dùng (`following.json`).

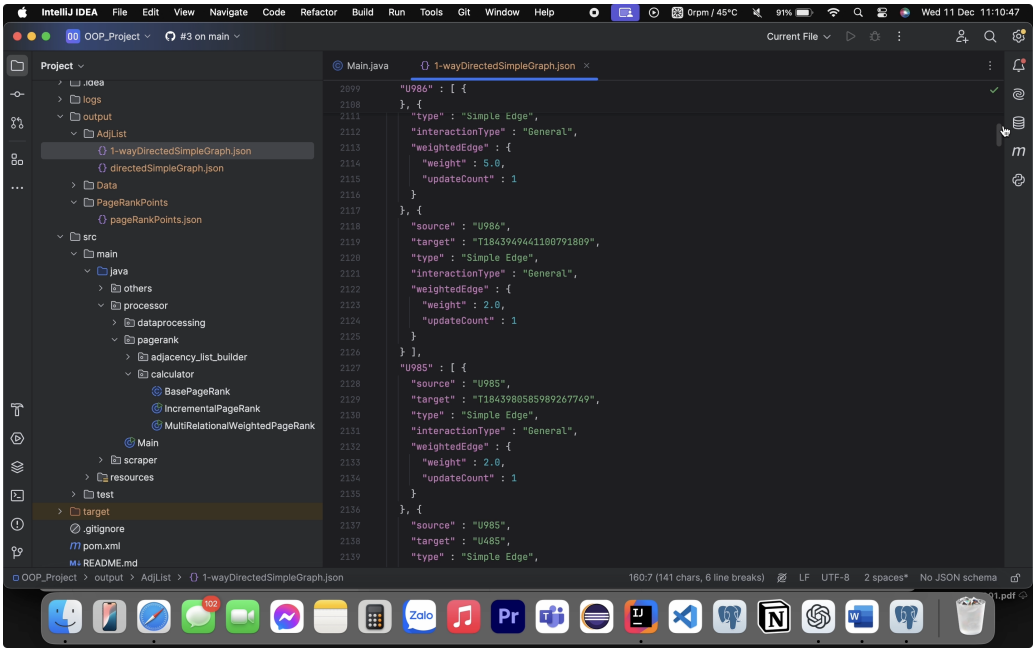


Hình 16: File JSON chứa danh sách các tweet (`retweetData.json`).

10.2 Quá trình xử lý dữ liệu

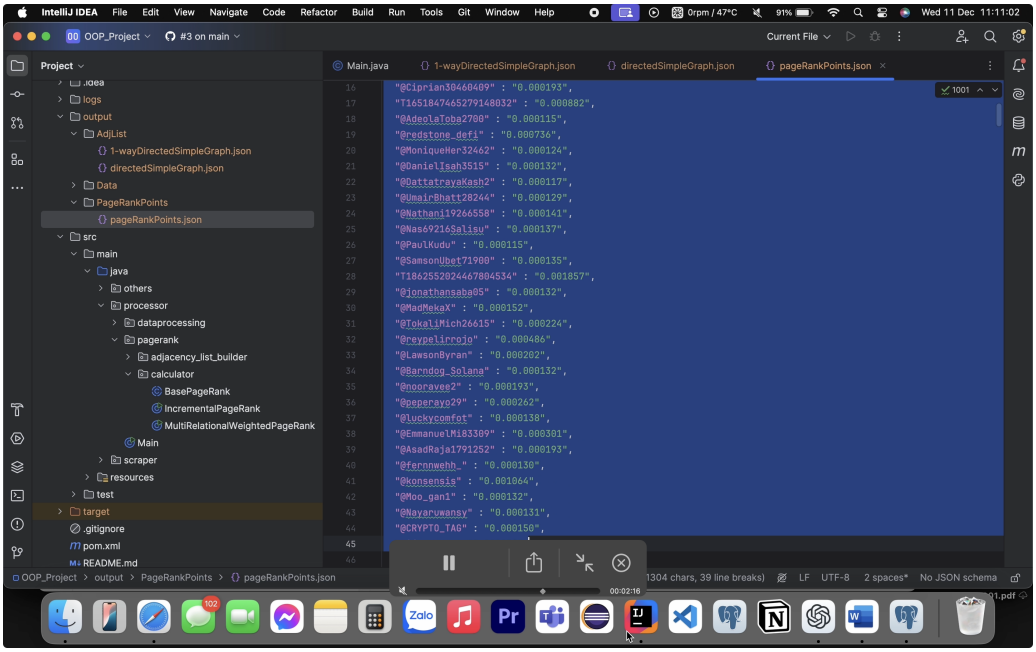


Hình 17: Bảng `Users` được nhập vào cơ sở dữ liệu PostgreSQL.



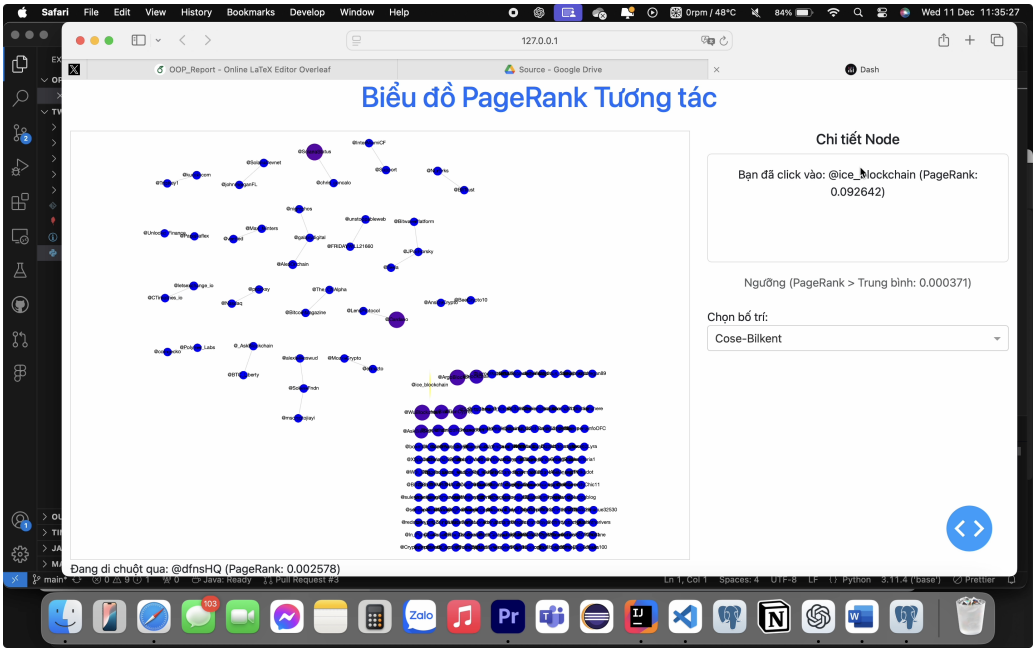
Hình 18: File danh sách kề (`directedSimpleGraph.json`) được xuất ra từ quá trình xử lý.

10.3 Kết quả tính toán PageRank



Hình 19: Kết quả tính toán PageRank trong file `pageRankPoints.json`.

10.4 Giao diện hiển thị đồ thị



Hình 20: Giao diện hiển thị đồ thị với Cytoscape.js. Các node và edges được hiển thị rõ ràng, với kích thước và màu sắc dựa trên giá trị PageRank.