

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

---□□□---



**BÁO CÁO BÀI TẬP LỚN**  
**ĐỀ TÀI**  
**PHÂN TÍCH VÀ XỬ LÝ DỮ LIỆU REDDIT**

**Nhóm sinh viên thực hiện:** Nhóm 12

**Mã lớp :** 154050

**GVHD:** TS.Trần Việt Trung

**Thành viên nhóm:** Ngô Văn Tân - 20210769

Trần Mạnh Cường - 20215325

Đinh Ngọc Toàn - 20215486

Nguyễn Mạnh Hiếu - 20210347

Nguyễn Viết Quang - 20215462

# MỤC LỤC

<b>MỤC LỤC.....</b>	<b>2</b>
<b>I. Đặt vấn đề.....</b>	<b>3</b>
1. Tổng quan bài toán.....	3
2. Phân tích bài toán.....	3
3. Giới hạn và Phạm vi của báo cáo.....	4
<b>II. Kiến trúc và thiết kế.....</b>	<b>5</b>
<b>III. Triển khai hệ thống.....</b>	<b>7</b>
1. Kafka.....	7
1.1 Chi tiết về Cấu Hình Kafka Cluster.....	7
1.2 Luồng xử lý.....	9
2. Spark.....	11
2.1 Làm sạch và biến đổi dữ liệu.....	11
2.2 Xử lý và lưu trữ dữ liệu.....	15
3. Airflow.....	16
4. Streamlit.....	19
5. Các bước chạy hệ thống.....	21
<b>IV. Bài học kinh nghiệm.....</b>	<b>22</b>
Kinh nghiệm 1: Xử lý Docker Socket Permission trong Airflow	
DockerOperator.....	22
Các giải pháp đã thử.....	22
Giải pháp cuối cùng.....	23
Bài học rút ra.....	23
Kinh nghiệm 2: Kết nối Database từ Container tới Host Machine.....	23
Mô tả vấn đề.....	23
Giải pháp:.....	24
<b>V. Kết luận.....</b>	<b>24</b>

# I. Đặt vấn đề

## 1. Tổng quan bài toán

Hiện nay, bóng đá là một trong những môn thể thao nổi tiếng nhất, thu hút hàng tỷ người hâm mộ trên toàn thế giới. Các chủ đề như chuyển nhượng cầu thủ, diễn biến giải đấu, và các sự kiện bên lề luôn nhận được sự quan tâm đặc biệt từ cộng đồng. Trong thời đại công nghệ, với sự ra đời của các nền tảng mạng xã hội, trong đó có Reddit, đã trở thành một nền tảng đông đảo người hâm mộ thảo luận, chia sẻ tin tức và bày tỏ quan điểm về môn thể thao này. Reddit không chỉ là nơi bày tỏ ý kiến của các cá nhân tham gia mà còn là một kho dữ liệu khổng lồ với nội dung phong phú, từ các phân tích chuyên sâu đến những tin đồn nóng hổi.

Việc xây dựng một hệ thống tự động crawl dữ liệu từ Reddit để thu thập thông tin liên quan đến bóng đá không chỉ giúp khai thác hiệu quả nguồn dữ liệu này mà còn mở ra cơ hội ứng dụng trong các lĩnh vực như phân tích xu hướng, dự đoán sự kiện, và hỗ trợ nghiên cứu chuyên sâu.

## 2. Phân tích bài toán

Trong bối cảnh sự phát triển mạnh mẽ của các nền tảng mạng xã hội và nhu cầu theo dõi, phân tích thông tin nhanh chóng, nhất là trong thời đại bùng nổ truyền thông, mọi thứ đều có thể truyền nhau thông qua mạng xã hội, bài toán đặt ra những thách thức :

- **Lưu trữ và xử lý dữ liệu với tính cập nhật cao:** Reddit là một mạng xã hội, do đó, Reddit liên tục được cập nhật với hàng nghìn bài viết và bình luận mỗi ngày. Hệ thống cần có khả năng lưu trữ và xử lý khối lượng dữ liệu lớn với tốc độ nhanh, đảm bảo không bỏ sót những thông tin mới và quan trọng.
- **Dữ liệu thiếu tính cấu trúc và không đồng nhất:** Reddit không có yêu cầu gì về cấu trúc của các bài viết của người dùng. Do đó, dữ liệu trên Reddit bao gồm văn bản, hình ảnh, video, liên kết và bình luận, thường thiếu tính đồng nhất và khó phân loại. Thêm vào đó, ngôn ngữ tự nhiên được sử dụng trong các thảo luận đôi khi chứa nhiều tiếng lóng, từ viết tắt, hoặc ngữ cảnh phức tạp, đặt ra thách thức cho việc tiền xử lý và làm sạch dữ liệu.

- **Phân tích xu hướng và trực quan hóa dữ liệu:** Việc phân tích dữ liệu Reddit không chỉ dừng lại ở việc thu thập thông tin mà còn phải phát hiện các xu hướng nổi bật như tin chuyển nhượng “hot”, cầu thủ được quan tâm nhất, hoặc diễn biến các giải đấu lớn. Các công cụ phân tích và trực quan hóa như Python, Streamlit sẽ giúp tổng hợp dữ liệu một cách quả và hiển thị các kết quả trực quan, dễ hiểu, phục vụ cho việc ra quyết định hoặc nghiên cứu sâu hơn.

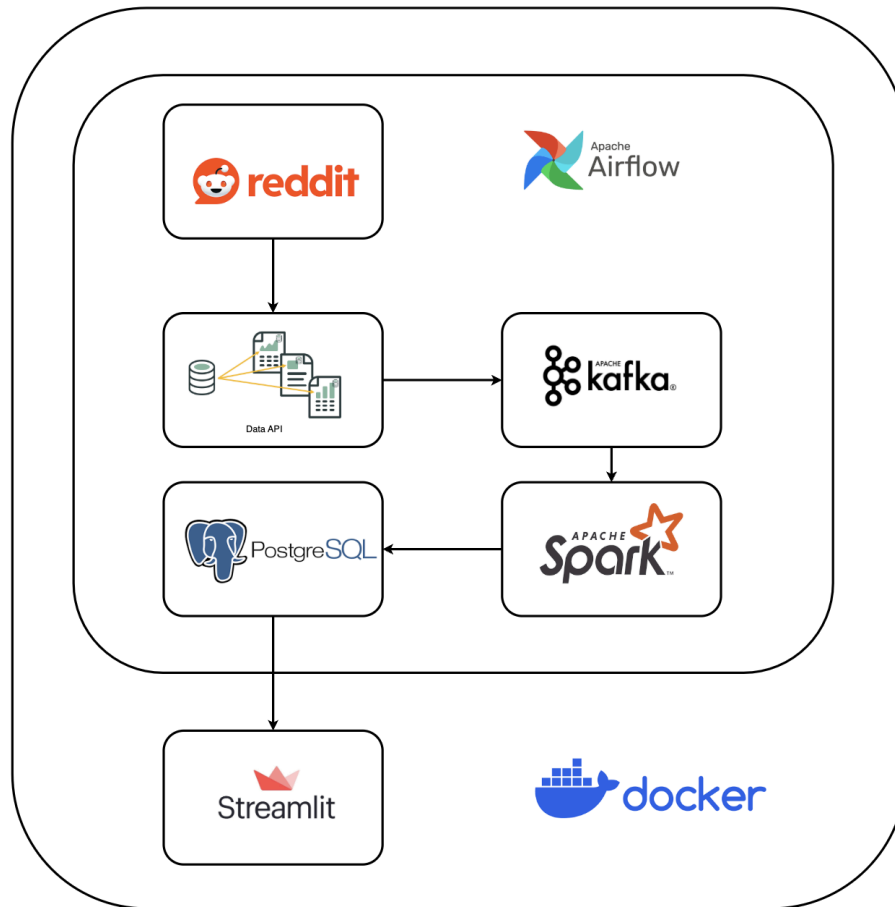
Với sự hỗ trợ của các hệ thống phân tích dữ liệu lớn, việc khai thác thông tin từ Reddit không chỉ dừng lại ở việc thu thập và tổng hợp dữ liệu mà còn giúp nhận diện xu hướng, dự đoán các sự kiện quan trọng trong bóng đá, từ đó phục vụ cho việc nghiên cứu, phân tích và đưa ra các quyết định hiệu quả.

### **3. Giới hạn và Phạm vi của báo cáo**

**Phạm vi:** Báo cáo tập trung vào việc thu thập và phân tích dữ liệu từ nền tảng Reddit liên quan đến các chủ đề bóng đá như chuyển nhượng cầu thủ, diễn biến giải đấu, và sự kiện nổi bật. Mục tiêu chính là nhận diện xu hướng thảo luận, đánh giá mức độ quan tâm của cộng đồng người hâm mộ và phát hiện các nội dung nổi bật trong khoảng thời gian cụ thể. Nội dung nhấn mạnh vào việc áp dụng các phương pháp và công nghệ xử lý dữ liệu lớn nhằm cung cấp góc nhìn tổng quan và khách quan về các vấn đề bóng đá trên Reddit.

**Giới hạn:** Báo cáo sử dụng dữ liệu được thu thập từ các subreddit công khai liên quan đến bóng đá, không bao gồm các nguồn dữ liệu riêng tư. Kết quả phân tích mang tính tham khảo và phụ thuộc vào chất lượng dữ liệu thu thập được từ Reddit. Do đó, báo cáo không đảm bảo tính đầy đủ tuyệt đối và có thể gặp các hạn chế về tính chính xác cũng như thời gian cập nhật dữ liệu.

## II. Kiến trúc và thiết kế



**Kiến trúc tổng thể:** Hệ thống sử dụng kiến trúc Lambda, tập trung vào xử lý batch và streaming.

### Batch Layer:

- **Apache Kafka:** Là hệ thống message broker giúp truyền dữ liệu theo thời gian thực từ Data API.
- **Apache Spark:** Thực hiện xử lý dữ liệu batch và streaming định kỳ từ Kafka và PostgreSQL.

### Serving Layer:

- **PostgreSQL**: Lưu trữ kết quả xử lý từ Apache Spark.
- **Streamlit**: Cung cấp giao diện visualization để hiển thị kết quả và báo cáo.
- **Dữ liệu được cập nhật theo chu kỳ batch và real-time.**

#### **Luồng dữ liệu:**

- **Dữ liệu thu thập từ Reddit**: Thông qua **Reddit API**, dữ liệu được gửi đến **Apache Spark** thông qua **Apache Kafka** để xử lý rồi lưu trực tiếp vào **PostgreSQL**.
- **Xử lý dữ liệu**: **Apache Spark** lấy dữ liệu từ Kafka để thực hiện các tác vụ ETL (Extract, Transform, Load).
- **Lưu trữ kết quả**: Dữ liệu sau khi được xử lý sẽ được lưu trữ trong **PostgreSQL**.
- **Visualization**: **Streamlit** truy cập vào PostgreSQL để trực quan hóa kết quả và báo cáo cho người dùng cuối.
- **Điều phối công việc**: **Apache Airflow** được sử dụng để quản lý, lập lịch và giám sát toàn bộ các pipeline xử lý dữ liệu.
- **Triển khai hệ thống**: Tất cả các dịch vụ trong hệ thống được đóng gói và triển khai trong môi trường **Docker**.

### III. Triển khai hệ thống

#### 1. Kafka

Trong dự án này, Apache Kafka đóng vai trò quan trọng như một distributed streaming platform, kết nối giữa Reddit API và hệ thống xử lý dữ liệu Spark. Hệ thống Kafka được triển khai với các thành phần sau:

- **Kafka Cluster:**
  - 3 broker độc lập để đảm bảo fault tolerance
  - Mỗi broker được cấu hình riêng biệt với ID và port khác nhau
  - Sử dụng bitnami/kafka:3.4 image để đảm bảo tính ổn định
- **Zookeeper:**
  - Quản lý cluster state và broker coordination
  - Sử dụng bitnami/zookeeper:3.8
  - Cấu hình anonymous access cho môi trường development
- **Kafka UI:**
  - Dashboard cho monitoring và quản lý
  - Triển khai qua provectuslabs/kafka-ui
  - Cung cấp giao diện trực quan để theo dõi hoạt động của cluster
- **Topic Configuration:**
  - Topic "reddit\_data" với 3 partitions
  - Replication factor 2 cho data redundancy
  - Được tối ưu cho việc xử lý song song

##### 1.1 Chi tiết về Cấu Hình Kafka Cluster

Mỗi broker trong cluster được cấu hình với các tham số riêng biệt để đảm bảo hoạt động hiệu quả:

##### **Broker 1 (kafka1):**

```

kafka1:
  image: 'bitnami/kafka:3.4'
  container_name: kafka1
  ports:
    - '9094:9094'
  networks:
    - airflow-kafka
  environment:
    - KAFKA_BROKER_ID=1
    - KAFKA_CFG_LISTENERS=PLAINTEXT://:9092,EXTERNAL://:9094
    - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://kafka1:9092,EXTERNAL://localhost:9094
    - KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper:2181
    - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=PLAINTEXT:PLAINTEXT,EXTERNAL:PLAINTEXT
    - ALLOW_PLAINTEXT_LISTENER=yes
    - KAFKA_CFG_DEFAULT_REPLICATION_FACTOR=2
    - KAFKA_CFG_NUM_PARTITIONS=3
  volumes:
    - kafka1_data:/bitnami/kafka
  depends_on:
    - zookeeper

```

- Port 9094 cho external access
- Port 9092 cho internal communication
- Cấu hình dual listener cho phép kết nối từ cả trong và ngoài docker network

### Broker 2 và Broker 3:

```

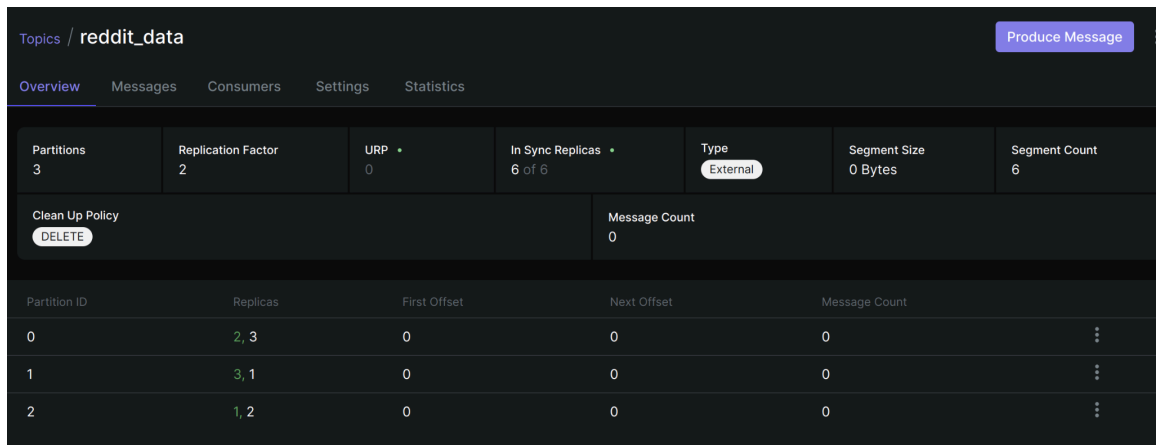
kafka2:
  image: 'bitnami/kafka:3.4'
  container_name: kafka2
  networks:
    - airflow-kafka
  environment:
    - KAFKA_BROKER_ID=2
    - KAFKA_CFG_LISTENERS=PLAINTEXT://:9092
    - KAFKA_CFG_ADVERTISED_LISTENERS=PLAINTEXT://kafka2:9092
    - KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper:2181
    - KAFKA_CFG_LISTENER_SECURITY_PROTOCOL_MAP=PLAINTEXT:PLAINTEXT
    - ALLOW_PLAINTEXT_LISTENER=yes
  volumes:
    - kafka2_data:/bitnami/kafka
  depends_on:
    - zookeeper

```



- Chỉ phục vụ internal communication
- Đóng vai trò backup và load balancing

## Kafka Topic



The screenshot shows the Kafka UI for the 'reddit\_data' topic. The 'Overview' tab is selected, displaying various configuration metrics and a table of partitions.

Partitions	Replication Factor	URP	In Sync Replicas	Type	Segment Size	Segment Count
3	2	0	6 of 6	External	0 Bytes	6

Partition ID	Replicas	First Offset	Next Offset	Message Count
0	2, 3	0	0	0
1	3, 1	0	0	0
2	1, 2	0	0	0

Topic "reddit\_data" được triển khai trên 3 partitions, cho phép việc xử lý dữ liệu được thực hiện song song trên nhiều broker. Dữ liệu từ Reddit được phân phối đều trên các broker thông qua cơ chế phân vùng này, giúp cân bằng tải và tối ưu hóa việc sử dụng tài nguyên của cluster. Việc sử dụng multiple partitions không chỉ tăng throughput của hệ thống mà còn cung cấp khả năng mở rộng thông qua việc thêm consumer để xử lý song song các partition khác nhau.

Cấu hình replication factor được set bằng 2, nghĩa là mỗi partition sẽ được sao lưu một bản replica trên broker khác trong cluster. Cấu hình này đảm bảo tính sẵn sàng cao của hệ thống, cho phép dịch vụ tiếp tục hoạt động ngay cả khi một broker gặp sự cố.

### 1.2 Luồng xử lý

Dữ liệu thu thập từ Reddit API đẩy vào Kafka sau đó được Kafka Producer gửi vào topic.

Để thu thập dữ liệu từ Reddit, dự án dùng thư viện Praw để tương tác với Reddit API. Các dữ liệu được thu thập là dữ liệu liên quan đến các chủ đề bóng đá trên Reddit

```
def create_reddit_client() -> praw.Reddit:
    """
    Creates and returns a Reddit client instance
    """
    return praw.Reddit(
        client_id="Db5HxiXppDbz_2yVdQk8Yg",
        client_secret="bUN0iWx8_dparp05-7JQlledUx0wBA",
        user_agent="python:Streaming_Kafka_Project:1.0 (by /u/New-Deer-1312)"
    )
```

Sau khi nhận được dữ liệu này thì kafka producer sẽ gửi vào Kafka Topic để Spark xử lý dữ liệu.

## 2. Spark

Spark sẽ xử lý các message nhận được từ kafka topic để đưa vào database Postgres

### 2.1 Làm sạch và biến đổi dữ liệu

Dưới đây là Schema đầu vào để Spark xử lý. Schema sẽ được định nghĩa rõ ràng để đảm bảo tính nhất quán của dữ liệu trong quá trình xử lý

```
self.reddit_schema = StructType([
    StructField("id", StringType()),
    StructField("title", StringType()),
    StructField("author", StringType()),
    StructField("post_time", FloatType()),
    StructField("upvotes", IntegerType()),
    StructField("downvotes", IntegerType()),
    StructField("num_comments", IntegerType()),
    StructField("score", IntegerType()),
    StructField("selftext", StringType()),
    StructField("first_level_comments_count", IntegerType()),
    StructField("second_level_comments_count", IntegerType()),
    StructField("text", StringType()),
    StructField("subreddit", StringType()),
    StructField("processing_timestamp", StringType())
])
logger.info("Schema initialized")
```

## a) Biến Đổi Bài Viết

```
transformed_df = df.select(
    col("id").alias("post_id"),
    "title",
    "author",
    from_unixtime(col("post_time")).cast("timestamp").alias("post_time"),
    "upvotes",
    "downvotes",
    "num_comments",
    "score",
    "text",
    "first_level_comments_count",
    "second_level_comments_count",
    "subreddit",
    hour(from_unixtime("post_time")).alias("hour_of_day"),
    dayofweek(from_unixtime("post_time")).alias("day_of_week"),
    current_timestamp().alias("created_at")
)
```

Các biến đổi chính:

- Đổi tên cột 'id' thành 'post\_id' cho rõ ràng
- Chuyển đổi unix timestamp thành timestamp chuẩn
- Thêm các trường phân tích thời gian:
  - hour\_of\_day: Giờ trong ngày
  - day\_of\_week: Ngày trong tuần
- Thêm trường created\_at để tracking thời gian xử lý

Các biến đổi này giúp:

- Chuẩn hóa dữ liệu thô từ Reddit để phân tích xu hướng thảo luận về bóng đá
- Xác định thời điểm người dùng tương tác nhiều nhất (theo giờ, theo ngày)
- Đánh giá mức độ quan tâm của cộng đồng thông qua số lượng comments và upvotes
- Theo dõi các chủ đề nóng trong cộng đồng bóng đá theo thời gian thực

## b) Xử Lý các thực thể

Ở bài tập lớn này, bọn em biến đổi các title đầu vào thành các thực thể thuộc loại ORG và PER qua thư viện Spacy.

Việc xử lý và phân tích thực thể trong dữ liệu Reddit đóng vai trò quan trọng trong việc theo dõi và đánh giá xu hướng thảo luận trong cộng đồng bóng đá. Thông qua việc trích xuất và phân tích các entity, hệ thống có thể tự động phát hiện và theo dõi các cầu thủ và câu lạc bộ được nhắc đến trong các bài viết, từ đó đánh giá mức độ quan tâm của cộng đồng đối với từng đối tượng. Về mặt phân tích thị trường, việc này giúp xác định những cầu thủ đang nhận được nhiều sự chú ý nhất, đồng thời theo dõi phản ứng của cộng đồng đối với các tin đồn chuyển nhượng và đánh giá danh tiếng của cầu thủ/CLB thông qua số lượng và chất lượng các cuộc thảo luận. Hệ thống còn có tiềm năng mở rộng để thực hiện phân tích cảm xúc, cho phép đánh giá phản ứng của người hâm mộ đối với các sự kiện và quyết định khác nhau, cũng như theo dõi sự thay đổi trong quan điểm của cộng đồng người hâm mộ theo thời gian.

```
for post_id, title in titles_list:
    doc = self.nlp(title)
    for ent in doc.ents:
        if ent.label_ in ['PERSON', 'ORG']:
            entities_list.append((
                post_id,
                ent.text,
                ent.label_,
                1, # mention_count
            ))
```

## c) Tính Toán Player Mentions

Các biến đổi chính được sử dụng ở đây là

- total\_mentions: Tổng số lần được nhắc đến
- total\_posts: Số bài viết có nhắc đến
- total\_upvotes: Tổng số upvote từ các bài viết

- avg\_post\_score: Điểm trung bình của các bài viết
- highest\_upvoted\_post\_id: Bài viết có điểm cao nhất
- lowest\_upvoted\_post\_id: Bài viết có điểm thấp nhất

Việc tính toán và phân tích lượng đề cập đến cầu thủ (player mentions) phục vụ nhiều mục đích quan trọng. Về phân tích độ nổi tiếng, hệ thống giúp xác định những cầu thủ được thảo luận nhiều nhất, đo lường mức độ tương tác với các bài viết về từng cầu thủ và theo dõi xu hướng thay đổi trong sự quan tâm của người hâm mộ. Đối với thị trường chuyển nhượng, việc theo dõi số lần nhắc đến giúp phát hiện sớm các tin đồn thông qua các đột biến trong dữ liệu, đồng thời đánh giá phản ứng của cộng đồng và tác động truyền thông xã hội của các thương vụ.

```
# Calculate aggregate metrics
metrics_df = joined_df.groupby("entity_name") \
    .agg(
        sum("mention_count").alias("total_mentions"),
        count("post_id").alias("total_posts"),
        sum("upvotes").alias("total_upvotes"),
        avg("score").alias("avg_post_score"),
        current_timestamp().alias("last_updated")
    )
```

#### d) Phân Tích Thời Gian

Các biến đổi chính được sử dụng ở đây là:

- avg\_upvotes: Trung bình upvotes
- avg\_comments: Trung bình số comments
- total\_posts: Tổng số bài viết

Việc phân tích thời gian (time engagement) giúp tối ưu hóa hoạt động của cộng đồng bằng cách xác định "golden hours" - những thời điểm có lượng tương tác cao nhất, phân tích patterns tương tác theo giờ và ngày trong tuần, từ đó tối ưu thời điểm đăng tin. Thông tin này được sử dụng để xây dựng chiến lược nội dung bằng cách điều chỉnh lịch đăng bài, phân bổ nguồn lực moderation phù hợp với thời gian hoạt động của cộng đồng và lập kế hoạch nội dung hiệu quả. Ngoài ra, việc phân tích hành vi người dùng giúp hiểu rõ thói quen theo múi giờ, phát hiện các sự

kiện ảnh hưởng đến patterns tương tác, từ đó có thể dự đoán và chuẩn bị tốt cho các thời điểm cao điểm.

```
result_df = df.groupby("hour_of_day", "day_of_week") \
    .agg(
        avg("upvotes").alias("avg_upvotes"),
        avg("num_comments").alias("avg_comments"),
        count("*").alias("total_posts"),
        current_timestamp().alias("last_updated")
    )
```

## 2.2 Xử lý và lưu trữ dữ liệu

Dữ liệu được xử lý theo batch tuần tự theo thứ tự sau:

- Lọc bỏ records đã tồn tại trong database
- Loại bỏ dữ liệu trùng lặp trong batch
- Xử lý và lưu trữ dữ liệu theo thứ tự:
  - Bài viết (posts)
  - Entities
  - Player mentions
  - Time engagement

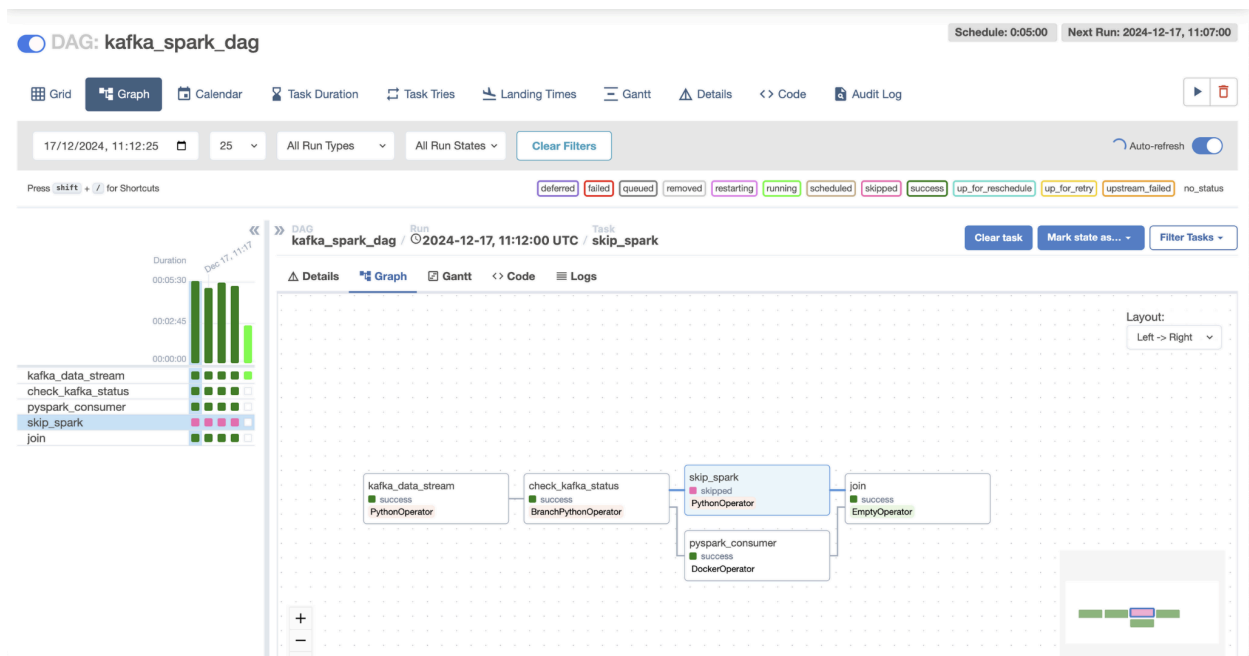
Các bảng phải được xử lý và lưu trữ theo thứ tự trên do ràng buộc khóa chính được thiết kế.

Trong quá trình xử lý, bạn có sử dụng caching thông qua Persist() cho các dữ liệu được sử dụng thường xuyên và Unpersist() để giải phóng sau khi không sử dụng nữa. Cuối cùng sau các phép biến đổi dữ liệu sẽ được ghi vào PostgreSQL thông qua JDBC connector.

### 3. Airflow

Airflow được sử dụng để thiết lập lịch cho các tác vụ:

- Tác vụ đầu tiên là Kafka Stream Task. Tác vụ này được triển khai bằng PythonOperator để chạy hàm streaming Kafka. Nhiệm vụ của nó là truyền dữ liệu từ API Reddit vào Kafka topic, khởi động quy trình xử lý dữ liệu.
- Tác vụ tiếp theo là Spark Stream Task. Tác vụ này sử dụng DockerOperator để thực thi. Nó chạy một Docker container sử dụng image Spark, có nhiệm vụ xử lý dữ liệu nhận được từ Kafka.
- Các tác vụ được sắp xếp theo thứ tự tuần tự, trong đó tác vụ streaming Kafka được thực thi trước tác vụ xử lý Spark. Thứ tự này rất quan trọng để đảm bảo rằng dữ liệu được truyền vào Kafka trước khi được Spark xử lý.



Sau đây là chi tiết các task vụ và nhiệm vụ của từng task:

Task đầu tiên trong pipeline là Kafka Stream Task, được triển khai thông qua PythonOperator với task\_id "kafka\_data\_stream". Task này đảm nhiệm việc thu thập dữ liệu từ Reddit API và đẩy vào Kafka topic. Task được cấu hình với do\_xcom\_push=True để có thể chia sẻ trạng thái thực thi với các task tiếp theo thông qua XCom của Airflow. Nếu quá trình stream dữ liệu thành công, task sẽ trả



về True, ngược lại sẽ trả về False để các task sau có thể quyết định luồng xử lý phù hợp.

```
kafka_stream_task = PythonOperator(  
    task_id="kafka_data_stream",  
    python_callable=stream_to_kafka,  
    do_xcom_push=True,  
    dag=dag,  
)
```

Tiếp theo là task kiểm tra trạng thái Kafka, được triển khai qua BranchPythonOperator với task\_id "check\_kafka\_status". Task này đóng vai trò then chốt trong việc điều hướng luồng xử lý dựa trên kết quả của Kafka stream task. Thông qua context của Airflow, task đọc giá trị XCom từ task Kafka stream trước đó và quyết định nhánh xử lý tiếp theo: nếu Kafka stream thành công, luồng sẽ được điều hướng tới task Spark processing; nếu thất bại, luồng sẽ chuyển sang nhánh skip\_spark để xử lý ngoại lệ.

```
check_kafka_status = BranchPythonOperator(  
    task_id='check_kafka_status',  
    python_callable=check_kafka_success,  
    provide_context=True,  
    dag=dag,  
)
```

Task xử lý chính của pipeline là Spark Processing Task, được triển khai thông qua DockerOperator với task\_id "pyspark\_consumer". Task này chạy trong một Docker container riêng biệt sử dụng image "reddit-consumer/spark:latest". Container được cấu hình để chạy Spark submit command với các packages cần thiết cho PostgreSQL và Kafka integration. Task được kết nối với mạng "airflow-kafka" để có thể giao tiếp với Kafka cluster. Task sử dụng docker\_url='tcp://docker-proxy:2375' để kết nối với Docker daemon thông qua proxy service, đảm bảo vấn đề permission được xử lý an toàn.

```

spark_stream_task = DockerOperator(
    task_id="pyspark_consumer",
    image="reddit-consumer/spark:latest",
    api_version="auto",
    auto_remove=True,
    # mount_tmp_dir=False,
    command="./bin/spark-submit --master local[*] --packages org.postgresql:postgresql:42.5.4,org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.0 ./sp
    docker_url='tcp://docker-proxy:2375',
    environment={
        'SPARK_LOCAL_HOSTNAME': 'localhost',
        'KAFKA_BOOTSTRAP_SERVERS': 'kafka1:9092',
        'KAFKA_GROUP_ID': 'spark-streaming-group',
        'KAFKA_AUTO_OFFSET_RESET': 'earliest'
    },
    network_mode="airflow-kafka",
    dag=dag,
)

```

Hai task cuối cùng của pipeline là Skip Task và Join Task. Skip Task được triển khai qua PythonOperator với task\_id "skip\_spark", thực hiện việc log lỗi và xử lý trường hợp Kafka stream thất bại. Join Task là một DummyOperator với task\_id "join" và trigger\_rule='none\_failed', đóng vai trò điểm hội tụ của các nhánh xử lý, đảm bảo pipeline kết thúc một cách graceful bất kể nhánh nào được thực thi. Cả hai task này tạo thành một cơ chế error handling hoàn chỉnh cho pipeline, giúp đảm bảo tính ổn định và khả năng theo dõi của hệ thống.

```

skip_spark_task = PythonOperator(
    task_id='skip_spark',
    python_callable=handle_skip,
    provide_context=True,
    dag=dag,
)

join_task = DummyOperator(
    task_id='join',
    trigger_rule='none_failed',
    dag=dag,
)

```

Tất cả các task được liên kết với nhau theo thứ tự:

```

kafka_stream_task >> check_kafka_status >> [spark_stream_task, skip_spark_task] >> join_task

```

## 4. Streamlit

Frontend được xây dựng bằng Streamlit để trực quan hóa dữ liệu từ PostgreSQL. Hệ thống cho phép người dùng tương tác với dữ liệu Reddit đã được xử lý theo thời gian thực, cung cấp insights về hoạt động của cộng đồng bóng đá.

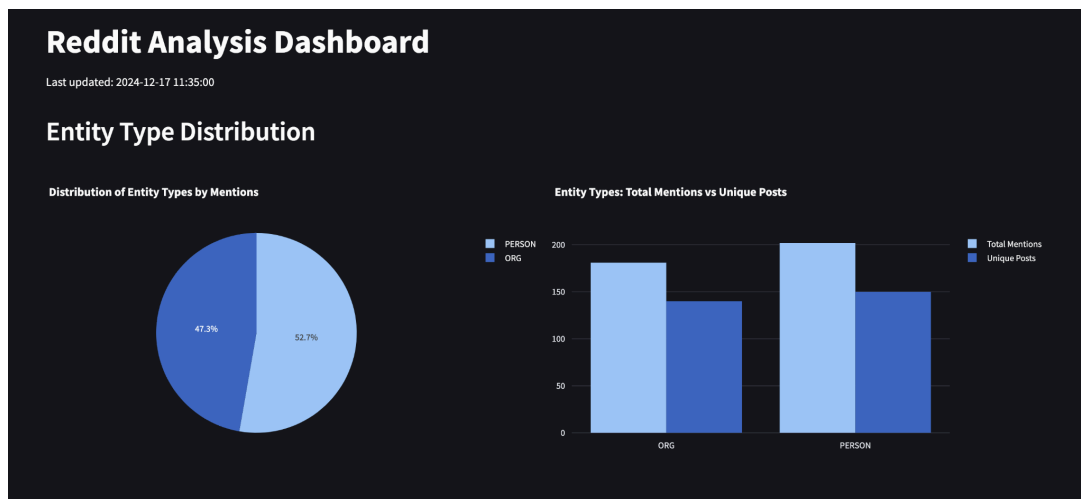
Chương trình kết nối với database thông qua các biến environment

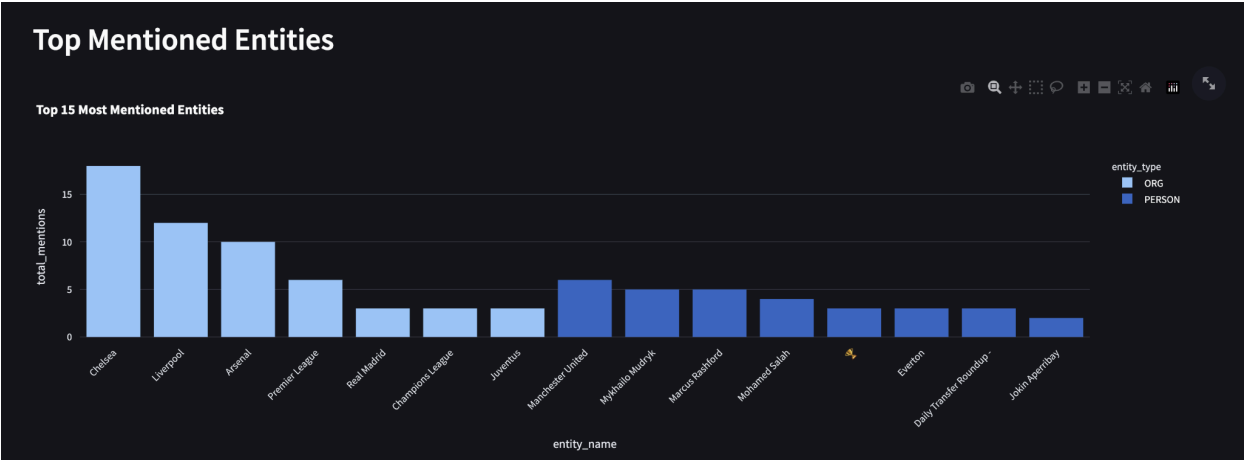
```
def get_connection():
    max_retries = 5
    retry_delay = 5
    retry_count = 0

    while retry_count < max_retries:
        try:
            dbname = os.getenv('POSTGRES_DB', 'reddit-data')
            user = os.getenv('POSTGRES_USER', 'postgres')
            password = os.getenv('POSTGRES_PASSWORD', 'postgres')
            host = os.getenv('POSTGRES_HOST', 'host.docker.internal')
            port = os.getenv('POSTGRES_PORT', '5432')

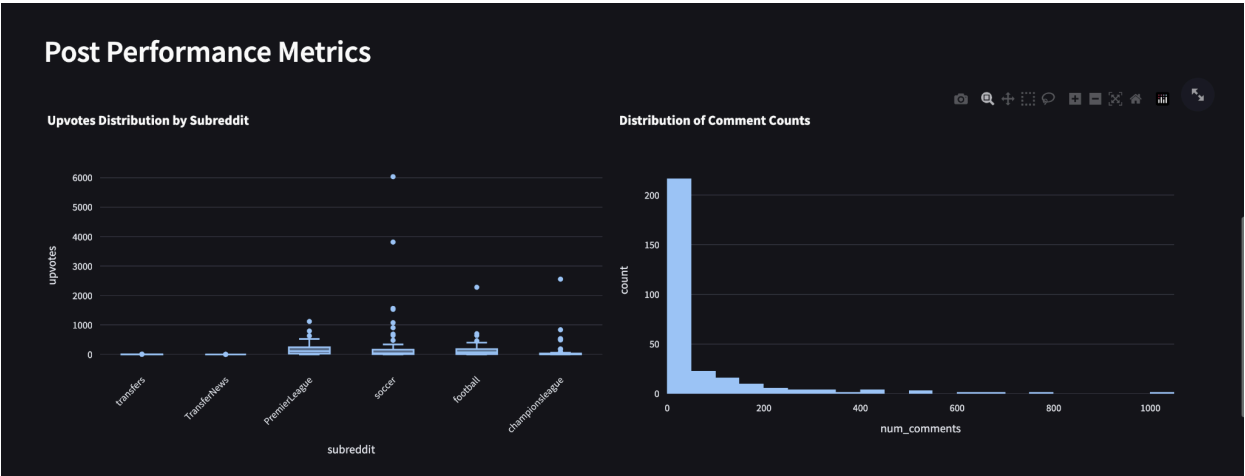
            connection_string = f"postgresql://{user}:{password}@{host}:{port}/{dbname}"
            engine = create_engine(connection_string)
```

Sau đó, chương trình sẽ thực hiện truy vấn sql và đưa ra một số các biểu đồ như dưới đây:

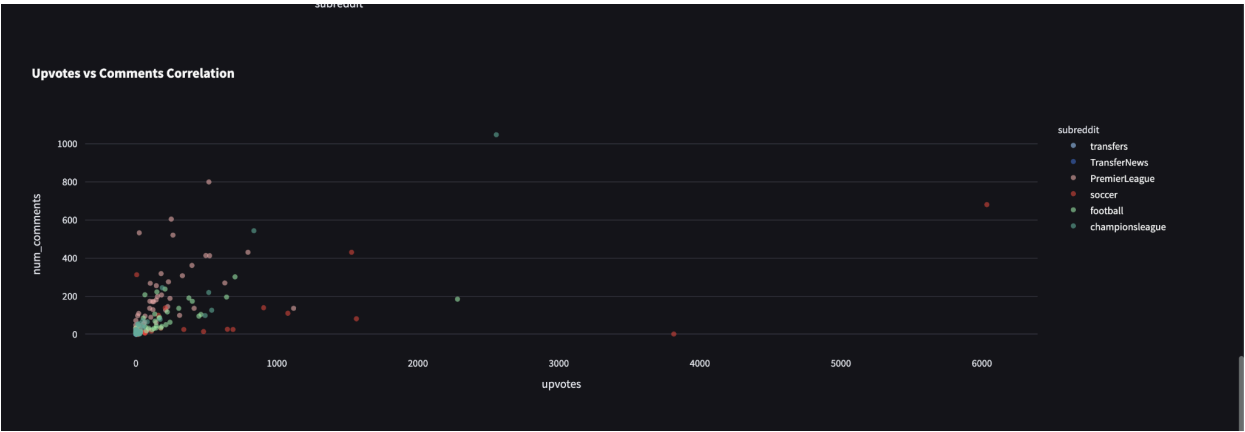




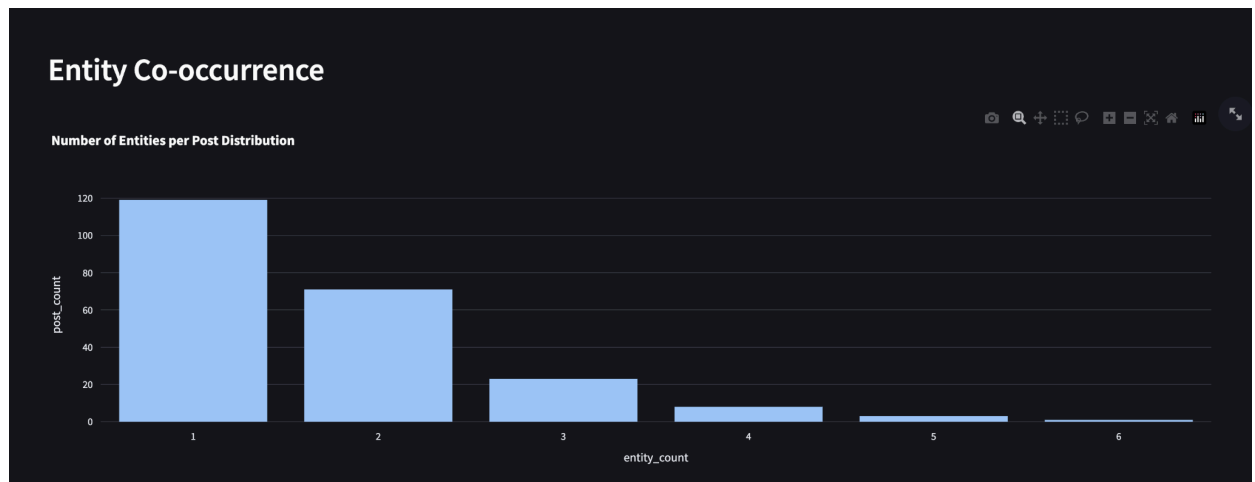
*Bảng: Các câu thủ và câu lạc bộ được quan tâm nhiều nhất*



*Bảng phân phối Upvotes và Comments ở các post*



*Bảng mối tương quan giữa upvote và comments*



*Bảng phân phối các thực thể trung bình trên một post*

## 5. Các bước chạy hệ thống

Bước 1: Khởi Tạo Network

```
docker network create airflow-kafka
```

Bước 2: Khởi Động Kafka Cluster

```
docker-compose up -d
```

Bước 3: Tạo Kafka Topic

```
docker exec -it kafka1 kafka-topics.sh \
    --create \
    --bootstrap-server kafka1:9092 \
    --topic reddit_data \
    --partitions 3 \
    --replication-factor 2
```

Bước 4: Khởi Tạo Database Schema

```
python scripts/create_table.py
```

## Bước 5: Build Spark Image

```
docker build -f spark/Dockerfile -t reddit-consumer/spark:latest .
```

## Bước 6: Khởi Động Airflow

```
docker compose -f docker-compose-airflow.yaml up -d
```

# IV. Bài học kinh nghiệm

## Kinh nghiệm 1: Xử lý Docker Socket Permission trong Airflow DockerOperator

- Context và background:
  - DockerOperator trong Airflow cần kết nối với Docker daemon để chạy container
  - Cách tiếp cận thông thường là mount `/var/run/docker.sock` vào Airflow container
  - Vấn đề permission khi truy cập docker.sock từ container Airflow
- Thách thức gặp phải:
  - Permission denied khi truy cập docker.sock trong Airflow container
  - Giải pháp chmod 777 không an toàn về mặt bảo mật
  - Cần một phương án vừa hoạt động vừa đảm bảo security
- Impact với hệ thống:
  - Ảnh hưởng đến khả năng chạy Docker tasks trong Airflow
  - Vấn đề về security nếu xử lý không đúng cách
  - Cần đảm bảo communication giữa các containers

## Các giải pháp đã thử

- Approach 1: Mount trực tiếp docker.sock
  - Mount `/var/run/docker.sock` vào Airflow container
  - Gặp vấn đề permission denied
- Approach 2: Chmod 777 docker.sock
  - Cấp full permission cho docker.sock
  - Giải quyết được vấn đề technical
  - Tạo ra lỗ hổng bảo mật nghiêm trọng

## Giải pháp cuối cùng

- Chi tiết giải pháp:
  - Sử dụng alpine/socat làm docker-proxy
  - Proxy lắng nghe trên port TCP 2375
  - Forward requests tới Docker socket
  - DockerOperator kết nối qua tcp://docker-proxy:2375

## Bài học rút ra

- Technical insights:
  - Socket permissions là vấn đề phức tạp trong container
  - Proxy pattern là giải pháp elegant cho vấn đề này
- Best Practices:
  - Không nên sử dụng chmod 777 cho docker.sock
  - Nên sử dụng proxy pattern thay vì truy cập trực tiếp

## Kinh nghiệm 2: Kết nối Database từ Container tới Host Machine

### Mô tả vấn đề

- Context và background:
  - Cần kết nối từ Spark, Streamlit container đến PostgreSQL database chạy trên host machine
  - Thông thường sẽ sử dụng localhost hoặc postgres làm hostname
  - Các hostname thông thường không hoạt động trong môi trường container
- Thách thức gặp phải:
  - Localhost trỏ đến container's localhost, không phải host machine
  - Container không thể resolve hostname 'postgres' hoặc 'localhost'
  - Network isolation của Docker gây khó khăn trong việc kết nối cross-container
- Impact với hệ thống:
  - Không thể kết nối được database
  - Dashboard không thể load dữ liệu

- Hệ thống không hoạt động như mong muốn

### **Giải pháp:**

Trong env:

- POSTGRES\_HOST=host.docker.internal

host.docker.internal là special DNS name Docker cung cấp tự động resolve thành IP của host machine.

## **V. Kết luận**

Nhóm đã xây dựng được một hệ thống phân tích dữ liệu Reddit với bao gồm Kafka cluster 3 nodes cho việc streaming dữ liệu, Spark cho xử lý và phân tích, được điều phối bởi Airflow và trực quan hóa qua Streamlit dashboard. Hệ thống mang lại giá trị về việc theo dõi xu hướng thảo luận, phát hiện sớm các tin đồn chuyển nhượng và phân tích mức độ quan tâm của cộng đồng đối với các cầu thủ và câu lạc bộ. Mặc dù gặp một số thách thức trong quá trình triển khai, đặc biệt là vấn đề về container networking và database connection và chưa thành công tích hợp Spark NLP vào, nhưng dự án đã tìm ra các giải pháp phù hợp và tối ưu. Những kinh nghiệm và bài học rút ra từ dự án sẽ là nền tảng quý giá cho việc phát triển và mở rộng hệ thống trong tương lai, với các định hướng như tăng cường khả năng phân tích, cải thiện user experience và triển khai hệ thống monitoring toàn diện.

Link mã nguồn: <https://github.com/hieuhb2003/BigData>