|  |  |
| --- | --- |
| HỌC VIỆN KỸ THUẬT QUÂN SỰ  **VIỆN CNTT & TT**  **BỘ MÔN HỆ THỐNG THÔNG TIN** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập – Tự do – Hạnh phúc** |

**NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

Họ và tên: Lê Quang Quốc Khánh, Lớp: Công nghệ thông tin 2, Khóa: 55

Ngành: Công nghệ thông tin, Chuyên ngành: Công nghệ thông tin

1. Tên đề tài: Nghiên cứu xây dựng ứng dụng phân tích dữ liệu phụ đề video trên mạng xã hội.

2.Các số liệu ban đầu:

Tự tìm hiểu.

3.Nội dung bản thuyết minh:

Mở đầu.

Chương 1: Tổng quan về bài toán lưu trữ và tìm kiếm dữ liệu lớn từ dữ liệu phụ đề Youtube.

Chương 2: Kiến thức cơ sở cho bài toán lưu trữ và tìm kiếm dữ liệu lớn từ dữ liệu phụ đề Youtube.

Chương 3: Xây dựng hệ thống lưu trữ và tìm kiếm dữ liệu lớn từ dữ liệu phụ đề Youtube.

Kết luận.

Danh mục tài liệu tham khảo.

4. Số lượng, nội dung các bản vẽ và các sản phẩm cụ thể:

Không có bản vẽ.

5. Cán bộ hướng dẫn.

Họ và tên: Nguyễn Trung Tín.

Cấp bậc: Thượng tá.

Chức vụ: Giảng viên.

Đơn vị: Bộ môn Khoa học máy tính, Viện Công nghệ thông tin và Truyền thông, Học viện Kỹ thuật Quân sự.

Hướng dẫn toàn phần.

Ngày giao: 27/06/2024 Ngày hoàn thành: 23/09/2024

*Hà Nội, ngày 23 tháng 9 năm 2024*

|  |  |
| --- | --- |
| **Chủ nhiệm bộ môn** | **Cán bộ hướng dẫn** |
| **3//, PGS, TS. Nguyễn Văn Giang** | **3//, GV, TS. Nguyễn Trung Tín** |

**Học viên thực hiện**

Đã hoàn thành và nộp đồ án ngày 24 tháng 9 năm 2024

**Lê Quang Quốc Khánh**

**DANH MỤC CÁC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **STT** | **Từ viết tắt** | **Diễn giải** |
|  | API | Application Programming Interface |
|  | DBMS | Database Management System |
|  | HDFS | Hadoop Distributed File System |
|  | HTML | Hypertext Markup Language |
|  | NLP | Natural Language Processing |
|  | RDD | Resilient Distribution Datasets |
|  | SBV | Youtube Subtitles |
|  | SQL | Structured Query Language |
|  | SRT | SubRip Subtitle |
|  | Sub | Subtitles |
|  | TF-IDF | Term Frequency-Inverse Document Frequency |
|  | TTML | Timed Text Markup Language |
|  | WebVTT | Web Video Text Tracks |
|  | XML | Extensible Markup Language |

**DANH MỤC HÌNH VẼ**

[Hình 1.1. Đặc trưng của dữ liệu lớn 10](#_Toc177949695)

[Hình 1.2. Ví dụ minh họa về cây Trie. 17](#_Toc177949696)

[Hình 1.3. Kỹ thuật kết hợp giữa Indexing và cây Trie trong tìm kiếm 18](#_Toc177949697)

[Hình 1.4. Các bước triển khai giải quyết bài toán 19](#_Toc177949698)

**Chương 2**

[Hình 2.1. Ví dụ về phụ đề SRT 20](#_Toc177949700)

[Hình 2.2. Ví dụ về phụ đề WebVTT 21](#_Toc177949701)

[Hình 2.3. Ví dụ về phụ đề TTML 22](#_Toc177949702)

[Hình 2.4. Ví dụ về phụ đề SBV 22](#_Toc177949703)

[Hình 2.5. Ví dụ về phụ đề TXT. 22](#_Toc177949704)

[Hình 2.6. Trang web GetSubs.cc 24](#_Toc177949705)

[Hình 2.7. Tiện ích Transcripts & Summary hiển thị phụ đề. 25](#_Toc177949706)

[Hình 2.8. Tổng quát kiến trúc HDFS. 28](#_Toc177949707)

[Hình 2.9. Quy trình ghi dữ liệu HDFS. 30](#_Toc177949708)

[Hình 2.10. Quy trình đọc dữ liệu HDFS. 31](#_Toc177949709)

[Hình 2.11. Cấu trúc của Apache Spark. 34](#_Toc177949710)

[Hình 2.12. Mô tả chi tiết về kiến trúc của Apache Spark 35](#_Toc177949711)

[Hình 2.13. Các công ty lớn nổi bật sử dụng Apache Spark. 36](#_Toc177949712)

[Hình 2.14. Một số công ty nổi tiếng sử dụng Apache Spark. 37](#_Toc177949713)

**Chương 3**

[Hình 3.1. Mô hình hoạt động của ứng dụng 41](#_Toc177949714)

[Hình 3.2. Mô hình tổng quan của Module thu thập dữ liệu phụ đề 43](#_Toc177949715)

[Hình 3.3. Giao diện Youtube Data API v3 trên Google Clouds 44](#_Toc177949716)

[Hình 3.4. File excel video\_list.xlsx 45](#_Toc177949717)

[Hình 3.5. Box hiển thị phụ đề bằng Glasp trên nền tảng Youtube. 45](#_Toc177949718)

[Hình 3.6. Đoạn kết quả thực thi chạy bot thu thập dữ liệu 46](#_Toc177949719)

[Hình 3.7. Cấu trúc cây Trie lưu trữ chữ cái 47](#_Toc177949720)

[Hình 3.8. Cấu trúc cây Trie lưu trữ âm tiết 48](#_Toc177949721)

[Hình 3.9. Quy trình thêm từ khóa vào cây Trie 50](#_Toc177949722)

[Hình 3.10. Quy trình hoạt động của Module tạo cây Trie 51](#_Toc177949723)

[Hình 3.11. Các bước tiến hành tiền xử lý dữ liệu. 52](#_Toc177949724)

[Hình 3.12. Giao diện quản trị máy chủ VMware ESXi. 54](#_Toc177949725)

[Hình 3.13. Cấu trúc thư mục lưu trữ trên HDFS 55](#_Toc177949726)

[Hình 3.14. Quy trình hoạt động của Module lưu trữ 55](#_Toc177949727)

[Hình 3.15. Danh sách tệp phụ đề .txt được lưu trữ trên HDFS 56](#_Toc177949728)

[Hình 3.16. Giao diện khởi chạy ngrox thành công 57](#_Toc177949729)

[Hình 3.17. Giao diện docs của FastAPI 58](#_Toc177949730)

[Hình 3.18. Quy trình hoạt động của Module xử lý tìm kiếm 58](#_Toc177949731)

[Hình 3.19. Quy trình xử lý tìm kiếm và đưa ra thông tin video 59](#_Toc177949732)

[Hình 3.20. Giao diện đăng nhập. 61](#_Toc177949733)

[Hình 3.21. Giao diện tạo tài khoản. 61](#_Toc177949734)

[Hình 3.22. Giao diện home của ứng dụng. 61](#_Toc177949735)

[Hình 3.23. Giao diện tìm kiếm thành công. 62](#_Toc177949736)

[Hình 3.24. Giao diện hiển thị phụ đề của một video. 62](#_Toc177949737)

[Hình 3.25. Giao diện xem video bằng box của video 63](#_Toc177949738)

[Hình 3.26. Giao diện thống kê truy vấn của Admin. 63](#_Toc177949739)

[Hình 3.27. Kết quả tìm kiếm 20 lần với bộ dữ liệu 5,874 videos. 64](#_Toc177949740)

[Hình 3.28. Kết quả tìm kiếm 20 lần với bộ dữ liệu 59,639 videos. 64](#_Toc177949741)

[Hình 3.29. Kết quả tìm kiếm 20 lần với bộ dữ liệu 105,221 videos. 65](#_Toc177949742)

**DANH MỤC BẢNG BIỂU**

[Bảng 1.1. Bảng so sánh HDFS và In-Memory Databases 13](#_Toc177949746)

**Chương 3**

[Bảng 3.1. Thông số phần cứng máy chủ VMware ESXi. 53](#_Toc177949743)

[Bảng 3.2. Thông tin máy chủ. 54](#_Toc177949744)

[Bảng 3.3. Các công cụ phần mềm hỗ trợ 54](#_Toc177949745)

**MỤC LỤC**

[**MỞ ĐẦU** 1](#_Toc178031513)

[1. Lý do chọn đồ án. 1](#_Toc178031514)

[2. Mục đích nghiên cứu. 1](#_Toc178031515)

[3. Đối tượng nghiên cứu. 2](#_Toc178031516)

[4. Phương pháp nghiên cứu. 2](#_Toc178031517)

[5. Bố cục đồ án tốt nghiệp. 3](#_Toc178031518)

[6. Lời cảm ơn. 3](#_Toc178031519)

[**Chương 1 TỔNG QUAN VỀ BÀI TOÁN LƯU TRỮ VÀ TÌM KIẾM DỮ LIỆU LỚN NỘI DUNG PHỤ ĐỀ TRÊN YOUTUBE.** 5](#_Toc178031520)

[1.1. Giới thiệu bài toán. 5](#_Toc178031521)

[1.2. Lý do cần phải áp dụng kỹ thuật lưu trữ và xử lý dữ liệu lớn vào hệ thống tìm kiếm. 7](#_Toc178031522)

[1.2.1. Ví dụ về so sánh chi phí lưu trữ. 7](#_Toc178031523)

[1.2.2. Ví dụ về việc nên sử dụng hệ thống lưu trữ dữ liệu lớn. 8](#_Toc178031524)

[1.3. Tổng quan về dữ liệu lớn. 9](#_Toc178031525)

[1.4. Tổng quan về kỹ thuật lưu trữ. 12](#_Toc178031526)

[1.5. Tổng quan kỹ thuật tìm kiếm. 13](#_Toc178031527)

[1.6. Hướng giải quyết bài toán. 19](#_Toc178031528)

[1.7. Kết luận Chương 1. 19](#_Toc178031529)

[**Chương 2 KIẾN THỨC CƠ SỞ CHO BÀI TOÁN LƯU TRỮ VÀ TÌM KIẾM DỮ LIỆU LỚN NỘI DUNG PHỤ ĐỀ TRÊN YOUTUBE.** 20](#_Toc178031530)

[2.1. Khai thác phụ đề Youtube. 20](#_Toc178031531)

[2.1.1. Định dạng phụ đề Youtube. 20](#_Toc178031532)

[2.1.2. Phương pháp khai thác phụ đề. 23](#_Toc178031533)

[2.2. Kỹ thuật lưu trữ dữ liệu lớn. 28](#_Toc178031534)

[2.2.1. Hadoop Distributed File System. 28](#_Toc178031535)

[2.2.2. Cơ sở dữ liệu trong bộ nhớ. 32](#_Toc178031536)

[2.3. Tổng quan về Apache Spark. 33](#_Toc178031537)

[2.4. Các công trình nghiên cứu liên quan. 37](#_Toc178031538)

[2.5. Kết luận Chương 2. 39](#_Toc178031539)

[**Chương 3 XÂY DỰNG ỨNG DỤNG LƯU TRỮ VÀ TÌM KIẾM DỮ LIỆU LỚN PHỤ ĐỀ YOUTUBE.** 40](#_Toc178031540)

[3.1. Yêu cầu ứng dụng. 40](#_Toc178031541)

[3.2. Mô hình hoạt động 41](#_Toc178031542)

[3.3. Xây dựng Module thu thập dữ liệu phụ đề. 43](#_Toc178031543)

[3.4. Xây dựng Module tạo cây Trie. 46](#_Toc178031544)

[3.4.1. Lựa chọn công cụ phân tán hiệu quả. 46](#_Toc178031545)

[3.4.2. Cấu trúc cây Trie. 47](#_Toc178031546)

[3.4.3. Mô hình tạo cây 51](#_Toc178031547)

[3.5. Xây dựng Module lưu trữ. 53](#_Toc178031548)

[3.6. Xây dụng Module API. 57](#_Toc178031549)

[3.7. Xây dựng Module xử lý tìm kiếm. 58](#_Toc178031550)

[3.8. Xây dựng Module giao diện. 60](#_Toc178031551)

[3.9. Thực nghiệm với các bộ dữ liệu khác nhau 64](#_Toc178031552)

[3.10. Kết luận Chương 3 65](#_Toc178031553)

[**KẾT LUẬN** 66](#_Toc178031554)

[1. Các kết quả đã thu được. 66](#_Toc178031555)

[2. Hạn chế của đồ án. 66](#_Toc178031556)

[3. Hướng phát triển của đồ án. 67](#_Toc178031557)

[**DANH MỤC TÀI LIỆU THAM KHẢO** 68](#_Toc178031558)

**MỞ ĐẦU**

1. Lý do chọn đồ án.

Trong bối cảnh thời đại số hóa hiện nay, việc mạng xã hội ngày càng phát triển, dữ liệu ngày càng nhiều, đa dạng đã đặt ra những thách thức lớn cho nhà nghiên cứu và chuyên gia công nghệ thông tin về đưa ra các phương pháp lưu trữ và quản lý dữ liệu lớn. Do đó dữ liệu lớn (Big Data) đã trở thành một phần quan trọng và không thể thiếu trong môi trường truyền thông và nội dung số. Tiêu biểu cho các mạng xã hội, Youtube với hàng triệu video được tải lên hàng ngày, đang trở thành một nguồn dữ liệu đa dạng và khổng lồ. Mỗi video thường đi kèm với các phụ đề, cung cấp thông tin về nội dung của video đó. Tuy nhiên, việc quản lý và tìm kiếm trong dữ liệu phụ đề này vẫn còn nhiều thách thức, đặc biệt khi đối mặt với khối lượng lớn và đa dạng của dữ liệu.

Hiện nay, việc tích hợp của dữ liệu lớn vào các nhiệm vụ trong Quân đội là hướng tiếp cận mới được nhiều đơn vị trong Quân đội đặt ra. Trong số các ứng dụng nổi bật, việc phát triển các hệ thống tự động hóa để thu thập, lưu trữ, và tra cứu dữ liệu lớn từ các phụ đề trên Youtube đã và đang được triển khai mạnh mẽ. Mục tiêu của những giải pháp này là cung cấp các công cụ linh hoạt, hiệu quả và phù hợp với các yêu cầu đặc thù của từng đơn vị, nhằm nâng cao khả năng đáp ứng nhanh chóng, chính xác trong quá trình thực hiện nhiệm vụ.

Để thực hiện thành công mục tiêu trên, các công nghệ lưu trữ dữ liệu lớn tiên tiến như Hệ thống Tập tin Phân tán Hadoop, cơ sở dữ liệu NoSQL, cơ sở dữ liệu dạng cột, cùng với các nền tảng lưu trữ đám mây và hệ thống cơ sở dữ liệu phân tán đã được đưa vào nghiên cứu và triển khai. Những phương pháp này không chỉ giúp cải thiện hiệu suất xử lý và truy xuất dữ liệu mà còn đóng vai trò quan trọng trong việc tối ưu hóa chi phí và thời gian, đảm bảo hiệu quả cao trong quản lý khối lượng dữ liệu khổng lồ từ các nguồn phụ đề trên Youtube.

2. Mục đích nghiên cứu.

Đồ án nghiên cứu và phân tích các thách thức liên quan đến việc lưu trữ và tìm kiếm dữ liệu lớn từ nội dung phụ đề trên Youtube. Xây dựng một hệ thống lưu trữ dữ liệu phụ đề Youtube được thu thập mới hàng ngày, bao gồm việc áp dụng các phương pháp phân tán và cơ chế sao lưu để đảm bảo tính an toàn và sẵn sàng của dữ liệu. Phát triển các giải pháp và công nghệ tìm kiếm dữ liệu phụ đề hiệu quả, bao gồm việc sử dụng các kỹ thuật tiền xử lý văn bản cho ngôn ngữ Tiếng Việt. Nghiên cứu và áp dụng hệ sinh thái Hadoop để tối ưu hóa việc lưu trữ và xử lý dữ liệu lớn từ phụ đề trên Youtube. Sử dụng Apache Spark để thực hiện các tác vụ tìm kiếm dữ liệu phụ đề một cách hiệu quả và nhanh chóng. Đánh giá và thử nghiệm hiệu suất của hệ thống lưu trữ và tìm kiếm phụ đề Youtube được thu thập. Đề xuất các hướng phát triển và cải tiến trong tương lai để nâng cao khả năng lưu trữ và tìm kiếm dữ liệu phụ đề từ Youtube, phục vụ cho cả nghiên cứu và ứng dụng thực tiễn.

3. Đối tượng nghiên cứu.

- Các phương pháp về thu thập dữ liệu phụ đề video.

- Các phương pháp lưu trữ dữ liệu trên hệ thống cơ sở dữ liệu phân tán.

- Các phương pháp xây dựng cây Trie trên Spark để phục vụ cho lưu trữ và tìm kiếm phụ đề.

- Một số cách sử dụng xử lý ngôn ngữ tự nhiên trong xử lý phụ đề thu thập được.

- Cách để cấu hình hệ thống phân tán từ cụm máy tính.

4. Phương pháp nghiên cứu.

**- Về mặt lý thuyết:**

+ Trang bị lại các kiến thức cơ bản về crawl dữ liệu, cụ thể là dự liệu phụ đề Youtube.

+ Trang bị kiến thức về hệ thống phân tán, Hadoop, HDFS, Spark.

+ Trang bị lý thuyết cơ bản về phương pháp xây dựng cây Trie, Xử lý ngôn ngữ tự nhiên (NLP).

**- Phương pháp thí nghiệm:**

+ Cài đặt hệ thống crawl dữ liệu.

+ Nghiên cứu và cài đặt hệ thống lưu trữ HDFS và xử lý Spark, xây dựng cây Trie trên bộ dữ liệu phụ đề.

+ Thử nghiệm và đánh giá kết quả trên kích thước dữ liệu khác nhau.

5. Bố cục đồ án tốt nghiệp.

Dựa vào các nội dung đã nêu ở trên, đồ án được tổ chức thành ba chương với các nội dung cụ thể như sau:

*Phần mở đầu:* Nêu lên tính cấp thiết, mục tiêu, nhiệm vụ của đề tài, phương pháp nghiên cứu, đối tượng và phạm vi nghiên cứu.

*Phần nội dung:* Gồm 3 chương:

- *Chương 1*. Trình bày tổng quan về bài toán lưu trữ và tìm kiếm dữ liệu lớn nội dung phụ đề trên Youtube. Nêu ra các vấn đề cần phải xử lý trong bài toán.

- *Chương 2.* Chương này trình bày các kiến thức cơ bản và nguyên lý quan trọng liên quan đến việc xử lý dữ liệu lớn và tìm kiếm thông tin. Các khái niệm về hệ thống lưu trữ phân tán, cơ chế sao lưu, quản lý dữ liệu và các phương pháp tìm kiếm dữ liệu hiệu quả. Từ đó hiểu rõ các lý thuật để giải quyết bài toán.

- *Chương 3.* Chương này tập trung vào việc thiết kế và triển khai các giải pháp cụ thể để xây dựng hệ thống lưu trữ và tìm kiếm dữ liệu lớn từ phụ đề trên Youtube. Việc áp dụng các kiến thức và công nghệ đã được trình bày trong Chương 2 để giải quyết các thách thức liên quan đến việc lưu trữ, tìm kiếm dữ liệu phụ đề.

*Phần kết luận:* Nêu lên những kết quả đạt được, chỉ ra một số hạn chế và hướng phát triển của đề tài trong thời gian tới.

6. Lời cảm ơn.

Để có thể hoàn thành đồ án một cách hoàn chỉnh, bên cạnh sự nỗ lực cố gắng của bản thân còn có sự giúp đỡ nhiệt tình của quý thầy cô trong Viện Công nghệ thông tin và Truyền thông đã truyền dạy kiến thức quý báu cho tôi về các môn đại cương cũng như các môn chuyên ngành, giúp tôi có nền tảng cơ sở lý luận vững vàng. Đặc biệt, xin chân thành bày tỏ lời cảm ơn sâu sắc đến giảng viên hướng dẫn Tiến sỹ Nguyễn Trung Tín luôn quan tâm sát sao, giúp đỡ và tạo mọi điều kiện thuận lợi nhất cho tôi để có thể hoàn thành đồ án này. Bên cạnh đó còn có sự ủng hộ động viên và tạo điều kiện của đơn vị, gia đình và bạn bè trong suốt thời gian học tập, nghiên cứu và thực hiện đồ án tốt nghiệp.

Mặc dù có nhiều cố gắng nhưng do thời gian và kiến thức còn hạn chế nên không tránh khỏi những sai sót. Vì vậy, tôi mong rằng sẽ nhận được sự góp, hướng dẫn của các thầy cô giảng viên để đồ án của tôi được hoàn thiện hơn.

Tôi xin chân thành cảm ơn!

**Chương 1  
TỔNG QUAN VỀ BÀI TOÁN LƯU TRỮ VÀ TÌM KIẾM DỮ LIỆU LỚN NỘI DUNG PHỤ ĐỀ TRÊN YOUTUBE.**

**1.1. Giới thiệu bài toán.**

Sự phát triển nhanh chóng của internet và mạng xã hội như Facebook, Twitter và Youtube đã mang lại những lợi ích to lớn và khẳng định vai trò quan trọng của họ trong đời sống xã hội. Đặc biệt, Youtube đã trở thành một nền tảng quan trọng cho việc chia sẻ thông tin và giải trí trên toàn thế giới. Hàng triệu video được tải lên mỗi giây, tạo ra một lượng dữ liệu lớn đáng kể.

Tuy nhiên, sự gia tăng về cấp độ, mật độ, tần suất và lưu lượng đăng video trên Youtube cũng mở ra những thách thức mới. Một số bộ phận và các thế lực thù địch đã tận dụng, áp dụng các chiêu thức và thủ đoạn tinh vi để thu hút sự quan tâm và theo dõi từ cộng đồng mạng, thậm chí thực hiện các biện pháp tuyên truyền, kích động và xuyên tạc thông tin. Họ cũng thường xuyên làm mới thông tin cũ, bịa đặt thông tin mới nhằm chống phá Quân đội và gây nhiễu loạn trong cộng đồng mạng. Những hành động này khiến một phần cư dân mạng mất phương hướng, lầm tưởng rằng đó là sự thật, dẫn đến hoài nghi và thiếu niềm tin vào Đảng và chế độ.

Với mục tiêu tìm kiếm nhanh chóng các video mang thông tin sai sự thật việc lưu trữ dữ liệu phụ đề (nội dung chính của video Youtube) đã được quan tâm, chú trọng. Từ đó bài toán bài toán lưu trữ và tìm kiếm dữ liệu lớn trong nội dung phụ đề trên Youtube đã được đặt ra .Với mục tiêu của bài toán là từ các chuỗi kí tự, từ khóa đưa vào thì cho ra được kết quả là các video Youtube có nội dung liên quan đến các từ khóa đưa vào trong thời gian ngăn nhất. Bài toán đã đưa ra nhiều thách thức đáng kể. Trước tiên, thu thập phụ đề từ video trên Youtube đặt ra một thách thức lớn do hiện chưa có công cụ tự động hoàn chỉnh hỗ trợ quá trình này. Bên cạnh đó, việc thu thập phụ đề cũng phải tuân thủ các chính sách của Youtube về bản quyền và quy định về việc sử dụng dữ liệu. Với lượng video khổng lồ trên Youtube, quá trình thu thập phụ đề yêu cầu đầu tư đáng kể về thời gian và công sức để thu được một lượng phụ đề đủ lớn và đa dạng để phục vụ cho mục đích nghiên cứu và phát triển hệ thống. Việc lưu trữ hàng triệu video và phụ đề tương ứng đòi hỏi một cơ sở hạ tầng máy chủ mạnh mẽ và hiệu quả. Cần phải có hệ thống lưu trữ phân tán và cơ chế sao lưu để đảm bảo tính an toàn và sẵn sàng của dữ liệu.

Tuy nhiên, thách thức lớn nhất không chỉ là lưu trữ mà còn là việc tìm kiếm thông tin trong lượng dữ liệu khổng lồ này. Việc trích xuất thông tin từ các phụ đề không chỉ yêu cầu hiệu suất cao mà còn đòi hỏi các giải thuật xử lý ngôn ngữ tự nhiên (NLP) tiên tiến. Không chỉ đơn thuần là tìm kiếm từ khóa, mà còn là khả năng hiểu ý nghĩa ngữ cảnh, ngữ pháp và ngữ nghĩa trong các câu văn.

Đối với người dùng, việc tìm kiếm thông tin từ dữ liệu phụ đề thu thập từ Youtube phải cung cấp kết quả chính xác và phản hồi nhanh chóng, đồng thời hỗ trợ các tính năng tìm kiếm nâng cao như tìm kiếm theo ngôn ngữ, thời lượng video, hoặc ngày tải lên.

Trong tương lai, việc nghiên cứu và phát triển các giải pháp lưu trữ và tìm kiếm dữ liệu lớn trong nội dung phụ đề trên Youtube sẽ tiếp tục là một lĩnh vực nổi bật trong ngành khoa học máy tính và công nghệ thông tin. Sự tiến bộ trong lĩnh vực này không chỉ mang lại lợi ích cho các nhà nghiên cứu và nhà phát triển, mà còn làm tăng trải nghiệm người dùng và mở ra nhiều cơ hội mới trong việc khai thác thông tin từ dữ liệu video trên mạng.

Tóm lại, một là việc xây dựng một hệ thống lưu trữ dữ liệu lớn yêu cầu sự kết hợp giữa cơ sở hạ tầng máy chủ mạnh mẽ, cơ chế sao lưu an toàn và các phương pháp quản lý dữ liệu hiệu quả. Điều này giúp đảm bảo tính sẵn sàng, an toàn và khả năng mở rộng của hệ thống, đồng thời cung cấp một môi trường lưu trữ đáng tin cậy cho dữ liệu lớn từ các phụ đề video trên Youtube. Hai là việc tìm kiếm dữ liệu từ hàng triệu phụ đề video Youtube cũng đặt ra nhiều thách thức và yêu cầu phức tạp. Để đảm bảo khả năng tìm kiếm chính xác và hiệu quả, cần phải phát triển các giải thuật và công nghệ tiên tiến trong việc xử lý, truy xuất dữ liệu.

**1.2. Lý do cần phải áp dụng kỹ thuật lưu trữ và xử lý dữ liệu lớn vào hệ thống tìm kiếm.**

Trong bối cảnh khối lượng dữ liệu ngày càng tăng theo cấp số nhân, các hệ thống tìm kiếm truyền thống không còn đáp ứng được yêu cầu về tốc độ và hiệu quả. Việc áp dụng kỹ thuật lưu trữ và xử lý dữ liệu lớn trở nên cần thiết để quản lý, lưu trữ và truy xuất thông tin một cách nhanh chóng. Những kỹ thuật này không chỉ tối ưu hóa quá trình xử lý dữ liệu mà còn đảm bảo khả năng mở rộng, giúp các tổ chức có thể xử lý khối lượng dữ liệu khổng lồ trong thời gian thực, từ đó nâng cao hiệu quả tìm kiếm và ra quyết định. Một số ví dụ về việc so sánh khi không sử dụng kỹ thuật lưu trữ và xử lý dữ liệu lớn vào hệ thống tìm kiếm:

1.2.1. Ví dụ về so sánh chi phí lưu trữ.

***Chi phí lưu trữ dữ liệu trên ổ cứng (HDD):***

Giá trung bình của ổ cứng HDD 1TB dành cho hệ thống doanh nghiệp: 40-80 USD.

Để lưu trữ 1 petabyte (PB) dữ liệu (1PB = 1024TB) , chi phí cho ổ cứng sẽ như sau:

40 USD x 1024 = 40,960 USD (ở mức giá thấp 40 USD/TB).

80 USD x 1024 = 81,920 USD (ở mức giá cao 80 USD/TB).

***Chi phí lưu trữ dữ liệu trên RAM:***

Giá trung bình của 1TB RAM DDR4 cho hệ thống doanh nghiệp vào khoảng 4,000 - 6,000 USD.

Để lưu trữ 1 petabyte (1PB = 1024TB) dữ liệu trong RAM, chi phí phần cứng sẽ như sau:

5,000 USD x 1024 = 5,120,000 USD (ở mức giá thấp 5,000 USD/TB).

6,000 USD x 1024 = 6,144,000 USD (ở mức giá cao 6,000 USD/TB).

***So sánh:***

***Lưu trữ 1PB dữ liệu trên ổ cứng HDD:*** 40,960 - 81,920 USD.

***Lưu trữ 1PB dữ liệu trên RAM:*** 5,120,000 - 6,144,000 USD.

Như vậy, chi phí lưu trữ dữ liệu trong RAM đắt hơn rất nhiều so với lưu trữ trên ổ cứng HDD, đặc biệt khi quy mô dữ liệu lớn như 1 petabyte, do đó việc áp dụng các hệ thống lưu trữ và xử lý dữ liệu lớn vào hệ thống tìm kiếm không chỉ giúp tối ưu hiệu suất mà còn đảm bảo tính kinh tế trong quản lý và vận hành.

1.2.2. Ví dụ về việc nên sử dụng hệ thống lưu trữ dữ liệu lớn.

Giả sử ta sử dụng list trong Python để lưu trữ danh sách phụ đề của các video. Cách tính kích thước tổng thể của 1 list có thể được tính bằng:

*Kích thước tổng = Kích thước cố định của list + Kích thước của từng phần tử*

*Kích thước cố định của list*: Đây là kích thước cơ bản của list, bao gồm con trỏ tới các phần tử. Kích thước này không phụ thuộc vào nội dung cụ thể của các phần tử trong list.

*Kích thước của từng phần tử*: Kích thước của từng phần tử trong list phụ thuộc vào kiểu dữ liệu và cấu trúc dữ liệu của phần tử. Nếu phần tử là đối tượng phức tạp (như list hoặc dictionary) , kích thước này sẽ lớn hơn do nó chứa thêm nhiều tham chiếu và các đối tượng con.

Nếu bạn có một list chứa phụ đề từ 1 triệu video Youtube, mỗi phụ đề chứa trung bình 500 từ, thì tổng số từ cần lưu trữ có thể là 500 triệu từ. Một từ chiếm khoảng 4 bytes, thì tổng dung lượng list này sẽ cần khoảng 2 GB RAM. Tuy nhiên, tổng dung lượng list trong Python không chỉ bao gồm dung lượng của các từ, mà còn bao gồm cả kích thước cố định của list. Kích thước cố định này bao gồm các con trỏ tham chiếu đến từng phần tử, và nó tăng dần theo số lượng phần tử. Điều này có thể dẫn đến việc sử dụng thêm một lượng bộ nhớ đáng kể.

Nếu tiếp tục tăng lượng dữ liệu (ví dụ: lên tới hàng trăm triệu phụ đề) , một máy tính cá nhân với 8GB hoặc 16GB RAM sẽ không đủ khả năng lưu trữ toàn bộ dữ liệu này trong bộ nhớ. Khi đó, hệ thống sẽ bắt đầu sử dụng bộ nhớ ảo trên ổ cứng, dẫn đến hiệu suất giảm mạnh.

Trong khi đó, nếu dữ liệu được phân tán trên nhiều máy, mỗi máy chỉ cần lưu trữ một phần nhỏ của dữ liệu và việc truy cập sẽ diễn ra nhanh hơn nhờ vào khả năng xử lý song song.

Từ các ví dụ ta có thể thấy được việc áp dụng hệ thống lưu trữ và xử lý dữ liệu lớn vào hệ thống tìm kiếm là điều tất yếu trong thời đại dữ liệu bùng nổ. Những hệ thống này không chỉ giúp xử lý hiệu quả lượng dữ liệu khổng lồ mà còn đảm bảo tốc độ truy vấn nhanh chóng và khả năng mở rộng linh hoạt, góp phần cải thiện chất lượng và hiệu quả của các dịch vụ tìm kiếm.

**1.3. Tổng quan về dữ liệu lớn.**

Trước đây, chúng ta mới chỉ biết đến dữ liệu có cấu trúc (structure data) , ngày nay, với sự kết hợp của dữ liệu và internet, đã xuất hiện một dạng khác của dữ liệu - Big data (dịch là “dữ liệu lớn”). Dữ liệu này có thể từ các nguồn như: hồ sơ hành chính, giao dịch điện tử, dòng trạng thái (status) , chia sẻ hình ảnh, bình luận, tin nhắn... của chính chúng ta, nói cách khác chúng là dữ liệu được sản sinh qua quá trình chia sẻ thông tin trực tuyến liên tục của người sử dụng. Trong thời gian gần đây, sự quan tâm đến tiềm năng lớn của dữ liệu lớn (Big Data) đã trở thành một xu hướng rõ ràng trong nhiều ngành công nghiệp. Nhiều cơ quan chính phủ đã công bố các kế hoạch quan trọng nhằm thúc đẩy nghiên cứu và ứng dụng của Big Data [10]. Bên cạnh đó, các vấn đề liên quan đến Big Data cũng thường được bàn luận và nhấn mạnh trên các phương tiện truyền thông hàng ngày như Economist [6], [10], New York Times [9], [20] và Nation Public Radio [5], [20]. Các tạp chí khoa học hàng đầu như Nature và Science cũng đã tập trung vào việc thảo luận về những thách thức và tác động của Big Data [21], [5]. Tất cả những dấu hiệu này cho thấy chúng ta đang bước vào một thời đại của Big Data [29].

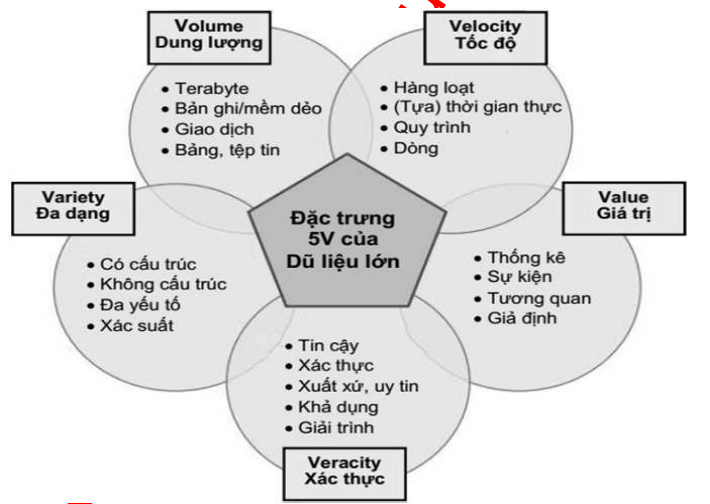
***Định nghĩa dữ liệu lớn:***

*- Theo wikipedia*: Dữ liệu lớn (Big data) là một thuật ngữ chỉ bộ dữ liệu lớn hoặc phức tạp mà các phương pháp truyền thống không đủ các ứng dụng để xử lý dữ liệu này.

*- Theo Gartner:* Dữ liệu lớn là những nguồn thông tin có đặc điểm chung khối lượng lớn, tốc độ nhanh và dữ liệu định dạng dưới nhiều hình thức khác nhau, do đó muốn khai thác được đòi hỏi phải có hình thức xử lý mới để đưa ra quyết định, khám phá và tối ưu hóa quy trình.

***Các đặc trưng của dữ liệu lớn:***

Năm 2001, công ty phân tích Metagroup (nay là Gartner) đã đưa ra khái niệm về các đặc trưng chính của Big Data, được biểu diễn thông qua ba yếu tố quan trọng, được gọi là 3V: Volume (Khối lượng) , Velocity (Tốc độ) , và Variety (Đa dạng). Nhưng theo thời gian, dữ liệu ngày càng phát triển mạnh mẽ, đến mức mà chúng ta bắt đầu gọi nó là dữ liệu lớn. Đáp ứng với sự phức tạp và quy mô ngày càng lớn của dữ liệu, hai yếu tố khác đã được thêm vào để mô tả đầy đủ hơn về Big Data, đó là Value (Giá trị) và Veracity (Chính xác). Do đó, khái niệm 2V mới này đã bổ sung vào 3V gốc để tạo ra một hình ảnh toàn diện về Big Data.



Hình 1.1. Đặc trưng của dữ liệu lớn

***Mô hình 5Vs mô tả các đặc trưng chính của Big Data như sau:***

***(1) Khối lượng dữ liệu (Volume).***

Đây là đặc điểm tiêu biểu nhất của dữ liệu lớn, khối lượng dữ liệu rất lớn. Kích cỡ của Big Data đang từng ngày tăng lên, và tính đến năm 2012 thì nó có thể nằm trong khoảng vài chục terabyte cho đến nhiều petabyte (1 petabyte = 1024 terabyte) chỉ cho một tập hợp dữ liệu. Dữ liệu truyền thống có thể lưu trữ trên các thiết bị đĩa mềm, đĩa cứng. Nhưng với dữ liệu lớn, chúng ta sẽ sử dụng công nghệ "đám mây" mới đáp ứng khả năng lưu trữ được dữ liệu lớn.

***(2) Tốc độ (Velocity).***

Tốc độ có thể hiểu theo 2 khía cạnh: (a) Khối lượng dữ liệu gia tăng rất nhanh (mỗi giây có tới 72, 9 triệu các yêu cầu truy cập tìm kiếm trên web bán hàng của Amazon) ; (b) Xử lý dữ liệu nhanh ở mức thời gian thực (real-time) , có nghĩa là dữ liệu được xử lý ngay tức thời ngay sau khi chúng phát sinh (tính đến bằng mili giây). Các ứng dụng phổ biến trong lĩnh vực Internet, Tài chính, Ngân hàng, Hàng không, Quân sự, Y tế – Sức khỏe như hiện nay phần lớn dữ liệu lớn được xử lý theo thời gian thực. Công nghệ xử lý dữ liệu lớn ngày nay đã cho phép chúng ta xử lý tức thì trước khi chúng được lưu trữ vào cơ sở dữ liệu.

***(3) Đa dạng (Variety).***

Đối với dữ liệu truyền thống, chúng ta hay nói đến dữ liệu có cấu trúc, nhưng ngày nay hơn 80% dữ liệu được sinh ra là phi cấu trúc (tài liệu, blog, hình ảnh, video, bài hát, dữ liệu từ thiết bị cảm biến vật lý, thiết bị chăm sóc sức khỏe…). Big Data cho phép liên kết và phân tích nhiều dạng dữ liệu khác nhau. Ví dụ, với các bình luận của một nhóm người dùng nào đó trên Facebook với thông tin video được chia sẻ từ Youtube và Twitter.

***(4) Độ tin cậy/chính xác (Veracity).***

Một trong những tính chất phức tạp nhất của Dữ liệu lớn là độ tin cậy/chính xác của dữ liệu. Với xu hướng phát triển của phương tiện truyền thông xã hội (Social Media) và mạng xã hội (Social Network) ngày nay, cùng sự gia tăng mạnh mẽ của tính tương tác và chia sẻ từ người dùng di động, việc xác định độ tin cậy và chính xác của dữ liệu ngày càng trở nên khó khăn. Bài toán phân tích và loại bỏ dữ liệu thiếu chính xác và nhiễu đang là một tính chất quan trọng của Big Data.

***(5) Giá trị (Value).***

Giá trị là đặc điểm quan trọng nhất của dữ liệu lớn. Khi bắt đầu triển khai xây dựng dữ liệu lớn, việc đầu tiên cần làm là xác định giá trị mà thông tin mang lại, từ đó quyết định có nên triển khai dữ liệu lớn hay không. Nếu có dữ liệu lớn mà chỉ nhận được 1% lợi ích từ nó, thì không nên đầu tư phát triển dữ liệu lớn. Kết quả dự báo chính xác thể hiện rõ nét giá trị mà dữ liệu lớn mang lại. Ví dụ, từ khối dữ liệu phát sinh trong quá trình khám, chữa bệnh, việc dự báo về sức khỏe sẽ chính xác hơn, từ đó giảm chi phí điều trị, các chi phí liên quan đến y tế. ***Ví dụ về dữ liệu lớn:***

Một công ty thương mại điện tử lớn có hàng triệu khách hàng và hàng ngàn sản phẩm. Họ thu thập thông tin về hành vi mua hàng, lịch sử giao dịch, phản hồi khách hàng, đánh giá sản phẩm và dữ liệu từ mạng xã hội. Tổng hợp các dữ liệu này sẽ tạo thành một nguồn dữ liệu khổng lồ, phức tạp và đa dạng.

Sử dụng Big Data, công ty có thể phân tích mẫu thị trường, dự đoán xu hướng mua hàng, cá nhân hóa trải nghiệm người dùng, tối ưu hóa quy trình chuỗi cung ứng và quảng cáo, cung cấp khuyến nghị sản phẩm cá nhân hóa, phát hiện gian lận và đánh giá hiệu suất sản phẩm.

Bằng cách khai thác Big-data này, công ty có thể đưa ra quyết định chiến lược và cung cấp dịch vụ tốt hơn cho khách hàng.

Ví dụ trên chỉ là một trong số vô số các trường hợp sử dụng Big-data. Các ngành công nghiệp khác như y tế, tài chính, giao thông, giáo dục cũng đang tận dụng Big-data để tạo ra giá trị và thúc đẩy sự phát triển.

1.4. Tổng quan về kỹ thuật lưu trữ.

Trong nghiên cứu về kỹ thuật lưu trữ dữ liệu lớn, nhiều hệ thống và cơ sở dữ liệu đặc biệt được đánh giá với mong muốn đáp ứng nhu cầu ngày càng tăng về lưu trữ và xử lý dữ liệu có khối lượng lớn. Trong phạm vi nghiên cứu này, chúng ta sẽ tập trung vào hai kỹ thuật lưu trữ chính, mỗi kỹ thuật đều có đặc điểm và ưu điểm riêng biệt. Dưới đây là sự giới thiệu ngắn gọn về mỗi kỹ thuật:

**Hadoop Distributed File System:** HDFS là hệ thống tệp phân tán được thiết kế để xử lý dữ liệu lớn trên các cụm máy tính. Với khả năng chia nhỏ dữ liệu thành các khối và lưu trữ chúng trên nhiều máy chủ, HDFS tối ưu hóa tính mở rộng và đảm bảo khả năng chịu lỗi.

**In-Memory Databases:** Cơ sở dữ liệu lưu trữ toàn bộ hoặc một phần dữ liệu trong bộ nhớ truy cập ngẫu nhiên để cung cấp truy cập nhanh chóng. Ví dụ như Apache Ignite và Redis, chúng thích hợp cho các ứng dụng đòi hỏi thời gian phản ứng nhanh.

Bảng 1.1. Bảng so sánh HDFS và In-Memory Databases

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Hadoop Distributed File System (HDFS)** | **Cơ sở dữ liệu trong bộ nhớ (In-Memory Databases)** |
| **Tốc độ xử lý** | Chậm hơn do sử dụng đĩa cứng. | Nhanh do lưu trữ dữ liệu trong bộ nhớ truy cập ngẫu nhiên (RAM). |
| **Khả năng mở rộng** | Cao, dễ dàng mở rộng bằng cách thêm máy chủ. | Hạn chế bởi dung lượng RAM. |
| **Khả năng chịu lỗi** | Cao, dữ liệu được sao chép trên nhiều máy chủ. | Dữ liệu dễ mất nếu không có cơ chế sao lưu. |
| **Chi phí phần cứng** | Thấp hơn, sử dụng phần cứng giá rẻ. | Cao hơn do cần RAM dung lượng lớn. |
| **Ứng dụng phù hợp** | Xử lý dữ liệu lớn theo lô (batch processing). | Ứng dụng yêu cầu thời gian thực, phản hồi nhanh. |
| **Giới hạn dung lượng** | Không giới hạn, có thể mở rộng với nhiều máy chủ. | Bị giới hạn bởi dung lượng RAM. |
| **Chi phí lưu trữ** | Thấp, sử dụng đĩa cứng. | Cao, sử dụng RAM. |
| **Quản lý và bảo trì** | Phức tạp, yêu cầu kiến thức kỹ thuật chuyên sâu. | Đơn giản hơn, nhưng cần quản lý bộ nhớ cẩn thận. |
| **Phù hợp với dữ liệu thời gian thực** | Không, chủ yếu dùng cho xử lý theo lô. | Có, phù hợp với các ứng dụng yêu cầu phản hồi tức thì. |
| **Ví dụ ứng dụng** | Phân tích dữ liệu lớn, xử lý batch. | Giao dịch tài chính, quảng cáo trực tuyến, giám sát. |

**1.5. Tổng quan kỹ thuật tìm kiếm.**

Hiện nay, trong lĩnh vực dữ liệu lớn, có nhiều kỹ thuật tìm kiếm khác nhau được sử dụng để tối ưu hóa quá trình truy xuất thông tin. Mỗi phương pháp có đặc điểm riêng, phù hợp với từng loại dữ liệu và hệ thống cụ thể. Các kỹ thuật tìm kiếm phổ biến hiện tại như:

***Kỹ thuật tìm kiếm hashing:***

Kỹ thuật tìm kiếm Hashing đóng vai trò quan trọng trong việc tìm kiếm và truy xuất dữ liệu. Hashing là quá trình chuyển đổi một dữ liệu đầu vào (gọi là khóa) thành một giá trị số (gọi là giá trị hash) thông qua một hàm băm. Quá trình này thường sử dụng một hàm băm determinist để đảm bảo rằng cùng một đầu vào luôn tạo ra cùng một giá trị hash. Khi xảy ra đụng độ (collision) , có hai phương pháp chính để xử lý: Open Addressing và Separate Chaining.

Ưu điểm của Hashing bao gồm hiệu suất tìm kiếm cao với độ phức tạp thời gian trung bình là O (1) , sự phù hợp cho cơ sở dữ liệu lớn và khả năng tìm kiếm đối tượng trong các cấu trúc dữ liệu khác nhau. Tuy nhiên, Hashing cũng có nhược điểm như xảy ra đụng độ, không phản ánh thứ tự của dữ liệu đầu vào và không đảm bảo thứ tự tìm kiếm.

***Kỹ thuật tìm kiếm bằng tóm tắt văn bản:***

Kỹ thuật tìm kiếm bằng tóm tắt văn bản [11], [16] là một phương pháp quan trọng trong lĩnh vực tìm kiếm dữ liệu lớn. Được xây dựng dựa trên các nguyên tắc của xử lý ngôn ngữ tự nhiên, phương pháp này tập trung vào việc tạo ra các tóm tắt ngắn gọn và có ý nghĩa từ các văn bản dài và phức tạp. Mục tiêu của kỹ thuật này là tạo ra các đoạn tóm tắt mô tả nội dung cốt lõi của văn bản một cách chính xác và ngắn gọn nhất có thể. Để thực hiện việc này, kỹ thuật tìm kiếm bằng tóm tắt văn bản thường sử dụng các thuật toán và kỹ thuật trong lĩnh vực xử lý ngôn ngữ tự nhiên, bao gồm cả phân tích cú pháp, nhận dạng thực thể, phân đoạn văn bản và rút trích thông tin. Sau đó, thông qua việc sử dụng các tiêu chí đánh giá như độ chính xác và tính tổng quát của tóm tắt, các đoạn văn ngắn được tạo ra để mô tả nội dung của văn bản một cách hiệu quả. Kỹ thuật tìm kiếm bằng tóm tắt văn bản đóng vai trò quan trọng trong việc giảm thiểu thời gian và công sức khi đọc và hiểu các văn bản dài, đặc biệt là trong môi trường tìm kiếm dữ liệu lớn, nơi lượng thông tin có thể là rất lớn và phức tạp. Bằng cách tạo ra các tóm tắt ngắn gọn và thông tin, kỹ thuật này giúp người dùng nhanh chóng tiếp cận và hiểu được nội dung cốt lõi của các văn bản, từ đó tăng hiệu quả và hiệu suất trong quá trình tìm kiếm và phân tích dữ liệu.

***Kỹ thuật tìm kiếm bằng trích xuất từ khóa:***

Kỹ thuật tìm kiếm bằng trích xuất từ khóa là một phương pháp quan trọng trong lĩnh vực tìm kiếm dữ liệu lớn. Phương pháp này tập trung vào việc xác định và trích xuất các từ khóa quan trọng từ các tài liệu văn bản để sử dụng như là các chỉ mục hoặc nhãn mô tả cho văn bản đó. Để thực hiện việc này, kỹ thuật tìm kiếm bằng trích xuất từ khóa thường sử dụng các thuật toán và kỹ thuật trong lĩnh vực xử lý ngôn ngữ tự nhiên để phân tích cú pháp và ngữ cảnh của văn bản. Các từ khóa được chọn ra dựa trên các tiêu chí như tần suất xuất hiện, tính phổ biến, độ quan trọng và mối liên kết với nội dung của văn bản.

Các từ khóa này thường được chọn ra một cách tự động và đại diện cho các khái niệm hoặc ý chính trong văn bản. Kỹ thuật tìm kiếm bằng trích xuất từ khóa đóng vai trò quan trọng trong việc tổ chức và quản lý thông tin trong các hệ thống tìm kiếm và phân tích dữ liệu lớn. Bằng cách trích xuất các từ khóa quan trọng, người dùng có thể nhanh chóng nhận biết và hiểu được nội dung của các văn bản một cách tổng quan. Đồng thời, việc sử dụng từ khóa cũng giúp cải thiện hiệu suất của các hệ thống tìm kiếm bằng cách tăng cường khả năng tìm kiếm và phân loại dữ liệu.

***Kỹ thuật tìm kiếm bằng trích xuất đặc trưng:***

Kỹ thuật tìm kiếm bằng trích xuất đặc trưng là một phương pháp quan trọng trong lĩnh vực tìm kiếm dữ liệu lớn. Phương pháp này tập trung vào việc xác định và trích xuất các đặc trưng quan trọng từ dữ liệu để sử dụng như là các chỉ mục hoặc nhãn mô tả cho dữ liệu đó. Để thực hiện việc này, kỹ thuật tìm kiếm bằng trích xuất đặc trưng thường sử dụng các thuật toán và kỹ thuật trong lĩnh vực học máy và xử lý dữ liệu lớn để phân tích và xử lý dữ liệu. Các đặc trưng được chọn ra dựa trên các tiêu chí như tính phổ biến, độ quan trọng và mối liên kết với nội dung của dữ liệu.

Các đặc trưng này thường được chọn ra một cách tự động và đại diện cho các thuộc tính hoặc đặc điểm quan trọng trong dữ liệu. Kỹ thuật tìm kiếm bằng trích xuất đặc trưng đóng vai trò quan trọng trong việc tổ chức và quản lý thông tin trong các hệ thống tìm kiếm và phân tích dữ liệu lớn. Bằng cách trích xuất các đặc trưng quan trọng, người dùng có thể nhanh chóng nhận biết và hiểu được các đặc điểm quan trọng của dữ liệu. Đồng thời, việc sử dụng các đặc trưng cũng giúp cải thiện hiệu suất của các hệ thống tìm kiếm bằng cách tăng cường khả năng tìm kiếm và phân loại dữ liệu.

***Kỹ thuật tìm kiếm Indexing (Chỉ mục):***

Trong nghiên cứu về kỹ thuật tìm kiếm dữ liệu lớn, Indexing (Chỉ mục) là một phương pháp quan trọng nhằm tối ưu hóa quá trình truy vấn dữ liệu trong các hệ thống quy mô lớn. Mô hình tìm kiếm này bao gồm hai thành phần chính:

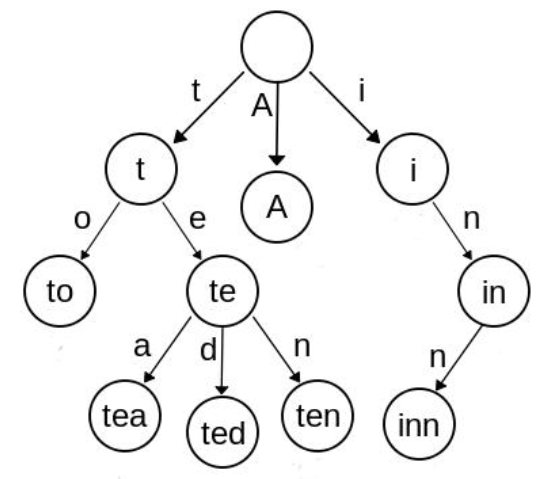
Dictionary (Từ điển) chứa các từ hoặc cụm từ duy nhất xuất hiện trong các tài liệu, đóng vai trò như khóa để tra cứu thông tin.

Posting List (Danh sách vị trí) là danh sách các tài liệu mà từ hoặc cụm từ trong Dictionary xuất hiện, kèm theo vị trí xuất hiện của chúng trong mỗi tài liệu. Posting List giúp nhanh chóng xác định được tài liệu liên quan khi có truy vấn tìm kiếm.

Phương pháp Indexing thường sử dụng cấu trúc Inverted Index (Chỉ mục ngược) , cho phép hệ thống dễ dàng tra cứu vị trí của các từ hoặc cụm từ trong cơ sở dữ liệu. Điều này giúp tăng tốc độ truy vấn, tối ưu hóa hiệu suất hệ thống và hỗ trợ tìm kiếm văn bản, cũng như các truy vấn phức tạp. Tuy nhiên, Indexing cũng có những hạn chế, bao gồm yêu cầu bộ nhớ lớn, chi phí cao cho việc cập nhật chỉ mục khi dữ liệu thay đổi, và không phù hợp cho các hệ thống mà dữ liệu thay đổi quá nhanh. Mặc dù vậy, với khả năng tối ưu hóa quá trình truy xuất thông tin, Indexing vẫn là một kỹ thuật quan trọng trong việc xử lý và tìm kiếm dữ liệu lớn trong các hệ thống tìm kiếm và cơ sở dữ liệu.

***Kỹ thuật tìm kiếm bằng cách xây dựng cây Trie:***

Kỹ thuật tìm kiếm bằng cây Trie là một phương pháp hiệu quả trong việc tìm kiếm và quản lý dữ liệu chuỗi, đặc biệt là trong các ứng dụng liên quan đến từ điển và tự động hoàn thành. Phương pháp này dựa trên việc tổ chức các chuỗi ký tự thành một cấu trúc cây, trong đó mỗi nút đại diện cho một ký tự và đường đi từ gốc đến một nút lá đại diện cho một chuỗi hoàn chỉnh. Cấu trúc cây Trie cho phép việc tìm kiếm và truy vấn dữ liệu trở nên nhanh chóng và hiệu quả, đặc biệt khi cần tìm các chuỗi có chung tiền tố.

****

Hình 1.2. Ví dụ minh họa về cây Trie.

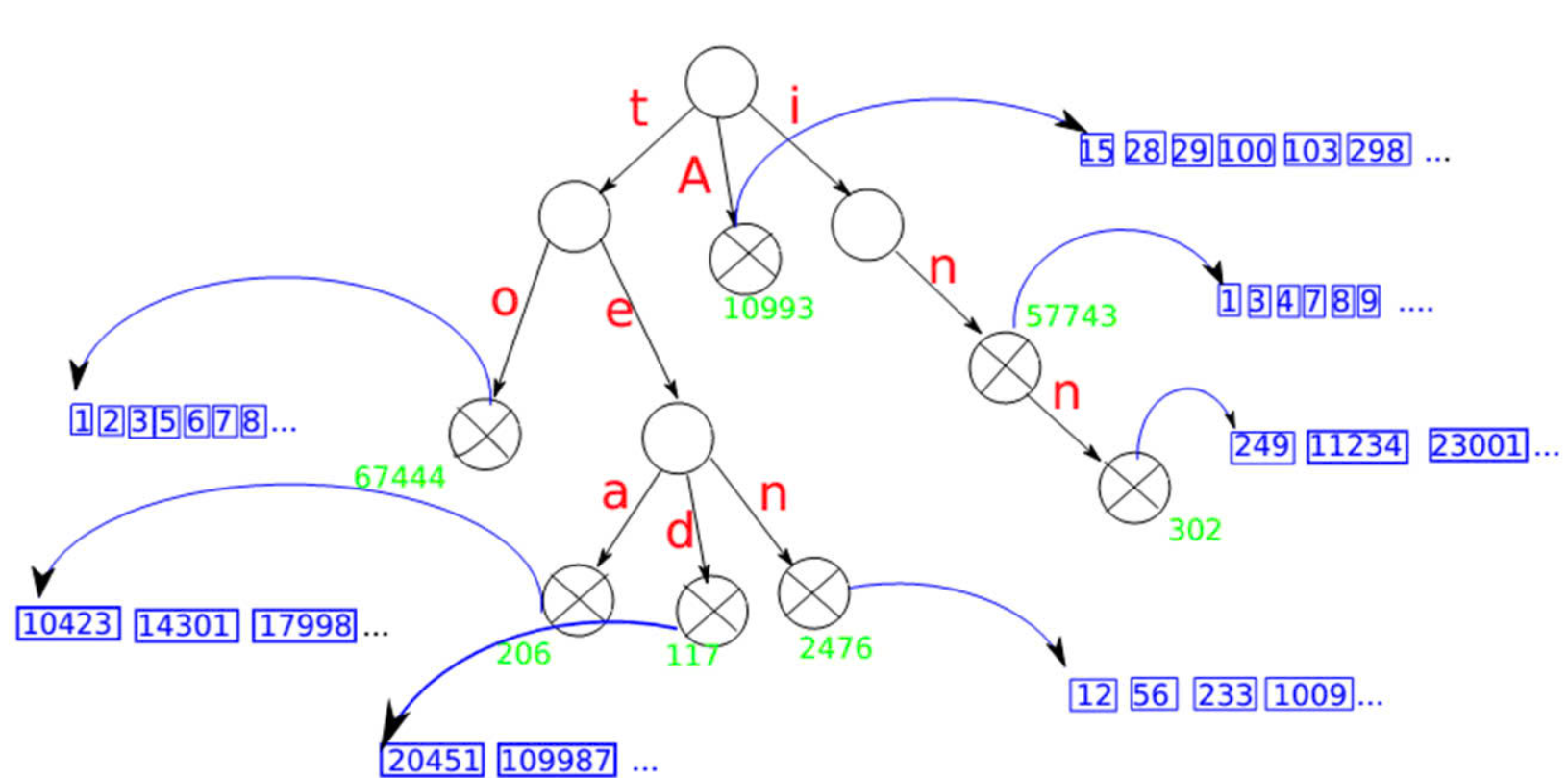
Trong quá trình xây dựng cây Trie, mỗi ký tự trong chuỗi được thêm vào một cách tuần tự, tạo ra các nhánh khác nhau của cây. Điều này giúp tổ chức dữ liệu một cách có hệ thống, đồng thời giảm thiểu sự trùng lặp của các ký tự chung trong các chuỗi. Khi tìm kiếm, cây Trie cho phép truy cập nhanh chóng vào các chuỗi bằng cách lần lượt duyệt qua các ký tự theo thứ tự, từ đó tìm ra các kết quả phù hợp chỉ trong thời gian ngắn.

Kỹ thuật tìm kiếm bằng cây Trie không chỉ giúp tối ưu hóa quá trình tìm kiếm mà còn hỗ trợ mạnh mẽ trong việc xử lý các bài toán như tự động hoàn thành, gợi ý từ, và kiểm tra chính tả. Nhờ vào cấu trúc dữ liệu đặc biệt của mình, cây Trie giúp người dùng nhanh chóng xác định và truy vấn các chuỗi ký tự cần thiết, đồng thời cải thiện hiệu suất của các hệ thống tìm kiếm và xử lý dữ liệu.

Tuy nhiên, để đáp ứng được nhu cầu của bài toán dữ liệu lớn hiện tại, việc nắm chắc và sử dụng kết hợp các kỹ thuật tìm kiếm là không thể thiếu. Nổi bật trong việc kết kết hợp các kỹ thuật tìm kiếm có thể nhắc đến:

***Kỹ thuật kết hợp giữa*** ***Indexing (Chỉ mục) và cây Trie trong tìm kiếm dữ liệu lớn:***

Trong phương pháp này, cây Trie được sử dụng để tìm kiếm từ khóa nhanh chóng trong Posting List của chỉ mục ngược. Cụ thể, cây Trie giúp tổ chức và tìm kiếm các từ khóa theo cách hiệu quả bằng cách duyệt qua từng ký tự của từ khóa, từ đó xác định nhanh vị trí của từ trong Dictionary. Sau khi tìm được từ khóa trong Dictionary, hệ thống sẽ sử dụng Posting List tương ứng để xác định các tài liệu chứa từ khóa đó.



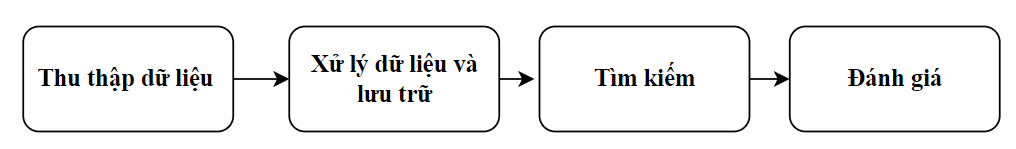
Hình 1.3. Kỹ thuật kết hợp giữa Indexing và cây Trie trong tìm kiếm

Khi có nhiều từ khóa trong truy vấn, hệ thống sẽ lần lượt tìm từng từ khóa trong cây Trie và truy xuất danh sách các tài liệu chứa các từ đó từ Posting List. Cuối cùng, quá trình so khớp các Posting List sẽ giúp tìm ra những tài liệu chứa tất cả các từ khóa tìm kiếm, đảm bảo tính chính xác và hiệu quả trong việc truy xuất thông tin.

Phương pháp kết hợp này không chỉ tận dụng được ưu điểm của cây Trie trong việc tìm kiếm từ khóa nhanh chóng mà còn khai thác tối đa sức mạnh của chỉ mục ngược trong việc tối ưu hóa việc truy xuất tài liệu liên quan. Điều này giúp cải thiện tốc độ và hiệu suất của hệ thống tìm kiếm, đặc biệt trong các hệ thống có khối lượng dữ liệu lớn.

**1.6. Hướng giải quyết bài toán.**

Từ cơ sở các kỹ thuật lưu trữ và tìm kiếm dữ liệu lớn, để giải quyết bài toán lưu trữ và tìm kiếm dữ liệu lớn nội dung phụ đề trên Youtube được triển khai theo các bước sau:



Hình 1.4. Các bước triển khai giải quyết bài toán

Thu thập dữ liệu: Dữ liệu phụ đề trên YouTube thu thập tự động, có khả năng tải và đồng bộ dữ liệu theo thời gian thực để đảm bảo tính liên tục và đầy đủ của dữ liệu.

Lưu trữ dữ liệu và đánh chỉ mục: Sau khi thu thập, dữ liệu sẽ được xử lý bằng cách để đánh chỉ mục dữ liệu và xây dựng cây Trie, sau đó lưu trữ trong HDFS, giúp tra cứu nhanh chóng khi cần thiết.

Tìm kiếm: Khi có truy vấn tìm kiếm từ người dùng, ứng dụng sẽ áp dụng kỹ thuật kết hợp giữa Indexing (Chỉ mục) và cây Trie trong tìm kiếm dữ liệu lớn đưa ra kết quả cho người dùng

Đánh giá: Sau khi thực hiện tìm kiếm, kết quả trả về cần được đánh giá để đảm bảo độ chính xác và phù hợp với truy vấn của người dùng.

**1.7. Kết luận Chương 1.**

Chương 1 giới thiệu tổng quan về bài toán thu thập, lưu trữ và tìm kiếm dữ liệu lớn từ nội dung phụ đề trên Youtube, đồng thời giải thích sự quan trọng của dữ liệu lớn trong các lĩnh vực cuộc sống hiện đại. Giới thiệu các kỹ thuật áp dụng cho việc lưu trữ và tìm kiếm dữ liệu lớn. Từ đó giải thích rõ hướng giải quyết bài toán hiện tại. Với những hiểu biết cơ bản về bài toán lưu trữ và tìm kiếm dữ liệu lớn từ nội dung phụ đề trên YouTube, cũng như các kỹ thuật áp dụng để giải quyết vấn đề này, chương 2 sẽ đi sâu vào các kiến thức cơ sở cần thiết, cung cấp nền tảng vững chắc cho việc phát triển và triển khai ứng dụng tìm kiếm hiệu quả trong lĩnh vực dữ liệu lớn.

**Chương 2  
KIẾN THỨC CƠ SỞ CHO BÀI TOÁN LƯU TRỮ VÀ TÌM KIẾM DỮ LIỆU LỚN NỘI DUNG PHỤ ĐỀ TRÊN YOUTUBE.**

**2.1. Khai thác phụ đề Youtube.**

Phụ đề (Subtitles) Youtube là dòng văn bản được đặt ở cuối màn hình. Phần phụ đề giúp người xem nắm được nội dung đoạn hội thoại đang diễn ra trong mỗi phân cảnh của video. Phụ đề có thể được viết dưới ngôn ngữ gốc của clip hoặc được dịch sang một ngôn ngữ khác tùy thuộc vào người dùng.

Người tạo nội dung trên Youtube có thể thêm phụ đề vào video của họ bằng cách sử dụng các công cụ chỉnh sửa phụ đề của Youtube hoặc nhập trực tiếp từ các tệp định dạng phụ đề khác nhau.

**2.1.1. Định dạng phụ đề Youtube.**

Youtube hỗ trợ nhiều định dạng phụ đề khác nhau bao gồm:

**SRT (SubRip Subtitle):** Đây là một trong những định dạng phụ đề phổ biến nhất. Mỗi tệp SRT (SubRip Subtitle) bao gồm các đoạn văn bản hiển thị trên màn hình kèm theo các dấu thời gian chính xác để xác định khi nào phụ đề xuất hiện và biến mất. Các tệp SRT có định dạng đơn giản và dễ hiểu, bao gồm số thứ tự của các dòng, thời gian bắt đầu và kết thúc, và nội dung phụ đề. Thời gian được biểu diễn bằng “giờ: phút: giây, miligiây”.

|  |
| --- |
| 1  00: 00: 01, 000 --> 00: 00: 04, 000  Đây là một ví dụ phụ đề SRT.  2  00: 00: 05, 000 --> 00: 00: 07, 000  Định dạng này rất phổ biến. |

Hình 2.1. Ví dụ về phụ đề SRT

**WebVTT (Web Video Text Tracks):** WebVTT (Web Video Text Tracks) là định dạng phụ đề thường được sử dụng cho HTML5 video. Nó tương tự như SRT nhưng có thêm khả năng hỗ trợ các tính năng nâng cao như định dạng văn bản, hỗ trợ hình ảnh, chú thích, và điều chỉnh vị trí của phụ đề trên video. WebVTT được thiết kế đặc biệt để hoạt động với các trình duyệt web hiện đại. Thời gian được biểu diễn bằng “giờ: phút: giây, miligiây”.

**VTT (****Video Text Tracks):** VTT (Video Text Tracks) là một phiên bản đơn giản của WebVTT.

|  |
| --- |
| **WEBVTT**  1  00: 00: 01.000 --> 00: 00: 05.000  <b>Phụ đề này được in đậm</b>  2  00: 00: 06.000 --> 00: 00: 10.000  <i>Phụ đề này được in nghiêng</i>  3  00: 00: 11.000 --> 00: 00: 15.000  <u>Phụ đề này có gạch chân</u>  4  00: 00: 16.000 --> 00: 00: 20.000  Phụ đề này nằm ở vị trí bên dưới cùng màn hình.  position: 90%  5  00: 00: 21.000 --> 00: 00: 25.000  Phụ đề này nằm ở bên phải màn hình.  align: end position: 10%  6  00: 00: 26.000 --> 00: 00: 30.000  Phụ đề này bao gồm một hình ảnh:  <img src="https: //www.example.com/image.png" alt="Hình ảnh" />  7  00: 00: 31.000 --> 00: 00: 35.000  Phụ đề này chứa <c.highlight>một chú thích quan trọng</c.highlight>. |

Hình 2.2. Ví dụ về phụ đề WebVTT

**TTML (Timed Text Markup Language):** TTML (Timed Text Markup Language) là một định dạng phụ đề XML-based (dựa trên XML) hỗ trợ nội dung có thời gian hiển thị. TTML cung cấp khả năng điều chỉnh định dạng văn bản như màu sắc, phông chữ và kích thước. Nó thường được sử dụng cho các nội dung có yêu cầu cao về định dạng phụ đề phức tạp, đặc biệt là trong phát sóng truyền hình hoặc các ứng dụng đa phương tiện lớn. Thời gian được biểu diễn bằng “giờ: phút: giây, miligiây”.

|  |
| --- |
| <tt xmlns="http: //www.w3.org/ns/ttml">  <body>  <div>  <p begin="00: 00: 01.000" end="00: 00: 04.000">Đây là một ví dụ phụ đề TTML.</p>  <p begin="00: 00: 05.000" end="00: 00: 07.000">TTML hỗ trợ định dạng văn bản phức tạp.</p>  </div>  </body>  </tt> |

Hình 2.3. Ví dụ về phụ đề TTML

**SBV (Youtube Subtitles):** SBV (Youtube Subtitles) là một định dạng phụ đề đơn giản thường được sử dụng bởi Youtube. Mỗi đoạn phụ đề trong SBV bao gồm một khoảng thời gian bắt đầu và kết thúc, theo sau là văn bản phụ đề. Các mốc thời gian sử dụng dấu phẩy làm dấu phân cách giữa phút, giây và phần nghìn giây.

|  |
| --- |
| 0: 00: 05.420, 0: 00: 12.059  [Music]  0: 00: 07.940, 0: 00: 15.059  Đây là một ví dụ phụ đề SBV. |

Hình 2.4. Ví dụ về phụ đề SBV

**TXT (Text):** Định dạng TXT rất đơn giản, chỉ bao gồm nội dung văn bản phụ đề mà không có thông tin thời gian. Định dạng này thường không được sử dụng trực tiếp để hiển thị phụ đề nhưng có thể được sử dụng để cung cấp nội dung văn bản của video. Phụ đề TXT không đi kèm với mốc thời gian hoặc định dạng đặc biệt.

|  |
| --- |
| Đây là đoạn phụ đề đầu tiên.  Đây là đoạn phụ đề thứ hai.  Phụ đề này có thể dài hơn. |

Hình 2.5. Ví dụ về phụ đề TXT.

**YCF (Youtube Caption File):** Đây là định dạng phụ đề độc quyền của Youtube, được sử dụng để chứa phụ đề tự động hoặc thủ công cho video Youtube. Youtube cung cấp khả năng tạo phụ đề trực tiếp thông qua công cụ Youtube Studio và xuất ra tệp YCF. Loại định dạng này chủ yếu được Youtube quản lý và sử dụng cho các mục đích nội bộ trên nền tảng của họ.

Các định dạng này đều chứa thông tin về thời gian và nội dung của phụ đề, nhưng có định dạng cú pháp và biểu diễn thời gian khác nhau. Sự chọn lựa giữa chúng thường phụ thuộc vào nền tảng hoặc ứng dụng cụ thể mà bạn đang sử dụng.

**2.1.2. Phương pháp khai thác phụ đề.**

Một trong những phương pháp chính thức và an toàn nhất để khai thác dữ liệu video từ các nền tảng video lớn như Youtube là sử dụng Application Programming Interface. API (Application Programming Interface) cho phép người dùng giao tiếp với các dịch vụ của Youtube thông qua các yêu cầu HTTP để lấy thông tin về video, bao gồm tiêu đề, mô tả, thời lượng, số lượt xem. Tuy nhiên phương pháp này không thể lấy phụ đề trực tiếp từ Youtube nên ta phải sử dụng các phương pháp khác để khai thác phụ đề từ Youtube. Mỗi phương pháp khai thác phụ đề đều có những đặc điểm riêng biệt về kỹ thuật, tính hiệu quả, cũng như các thách thức gặp phải trong quá trình thực hiện. Trong phần này, chúng ta sẽ xem xét các phương pháp khai thác phụ đề phổ biến:

***Khai thác phụ đề bằng công cụ yt-dlp/Youtube-dl:***

Đây là các công cụ mã nguồn mở mạnh mẽ, cho phép tải xuống nội dung từ Youtube và các nền tảng video khác. Đặc biệt, yt-dlp có thể hỗ trợ tải phụ đề từ video nếu video đó có phụ đề được cung cấp bởi người đăng tải hoặc do Youtube tự động tạo ra. Công cụ này có thể xuất phụ đề dưới nhiều định dạng như .srt, .vtt, hoặc thậm chí là .txt, tùy theo yêu cầu của người dùng.

|  |
| --- |
| Bash |
| yt-dlp --write-subs --skip-download "https: //www.Youtube.com/watch?v=example" |

Phương pháp này có ưu điểm là dễ sử dụng, không yêu cầu cấu hình phức tạp và có thể thu thập phụ đề với số lượng lớn video. Tuy nhiên, điểm yếu của phương pháp này là nó không chính thức và có thể bị thay đổi hoặc ngừng hoạt động nếu Youtube điều chỉnh lại cấu trúc trang web hoặc cơ chế phân phối nội dung. Dữ liệu phụ đề sẽ phụ thuộc vào Youtube cung cấp.

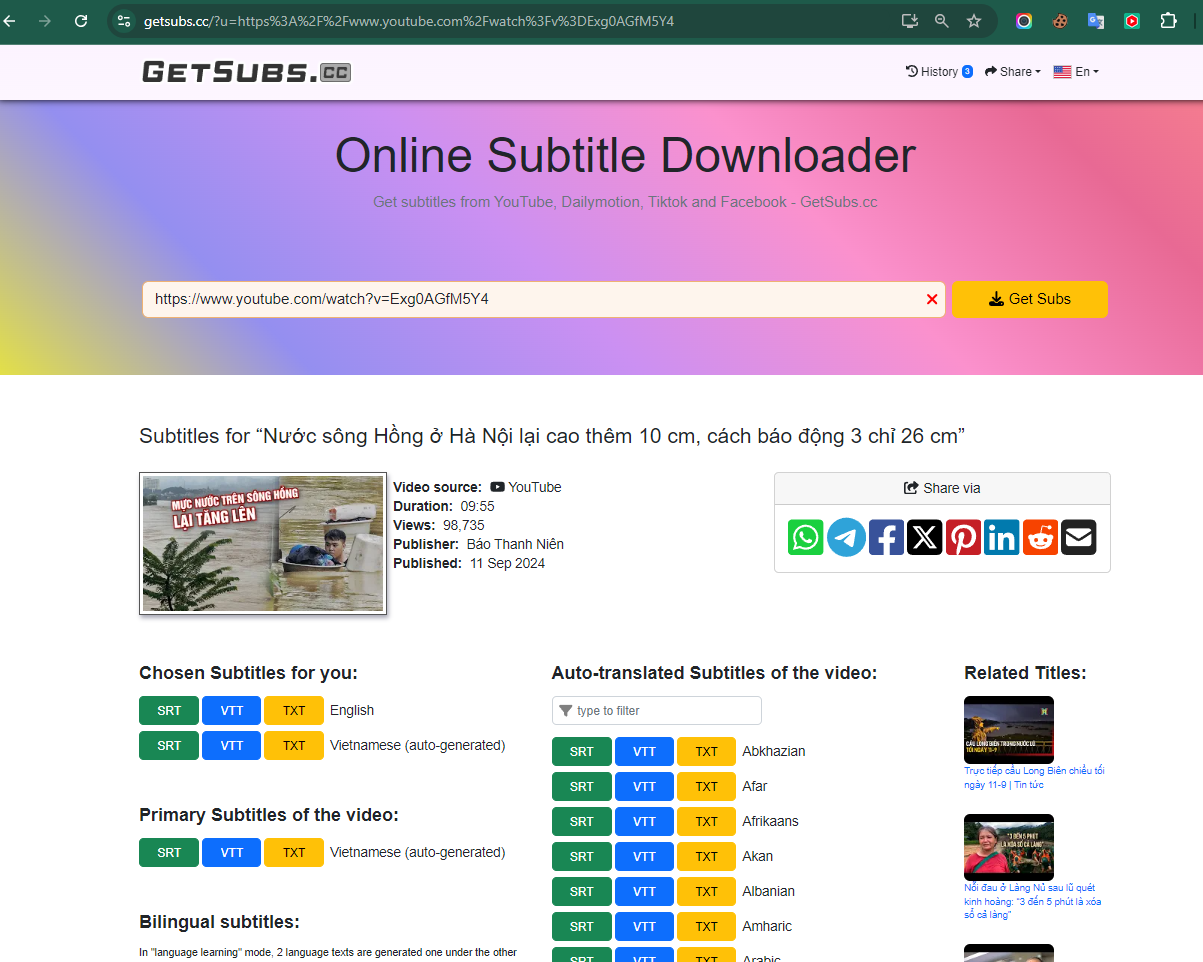
***Khai thác phụ đề bằng phương pháp Web Scraping kết hợp với nguồn trang bên thứ 3:***

Hiện nay có rất nhiều trang web trực tuyến giúp người dùng dễ dàng tải phụ đề từ các video trên Youtube và nhiều nền tảng khác. Một số trang web có thể tải xuống phụ đề miễn phí:

- Downsub là dịch vụ hỗ trợ người dùng tải phụ đề trên video Youtube hỗ trợ tiếng Việt hoàn toàn miễn phí, cung cấp nhiều ngôn ngữ phụ đề khác nhau để người dùng chọn như tiếng Việt, tiếng Pháp, tiếng Nhật, ...

- Getting Youtube Subtitles là dịch vụ hỗ trợ tải phụ đề Youtube với giao diện đơn giản, dễ sử dụng, hỗ trợ nhiều ngôn ngữ tải về giúp bạn linh động trong việc lựa chọn.

- Getsubs cũng tương tự, nhưng trang này tập trung vào việc tìm kiếm phụ đề từ nhiều nguồn khác nhau, hỗ trợ người dùng tìm kiếm phụ đề cho nhiều ngôn ngữ và định dạng khác nhau.

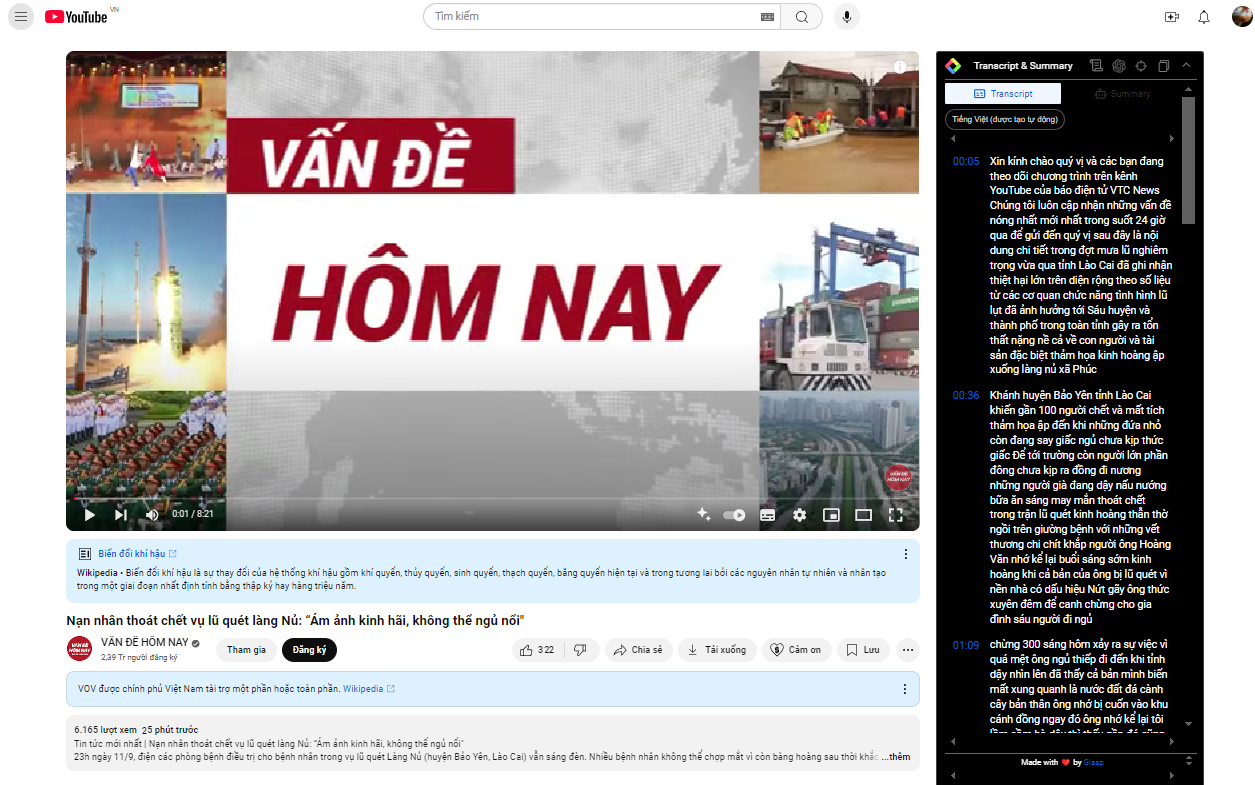


Hình 2.6. Trang web GetSubs.cc

Những công cụ này đặc biệt hữu ích cho người dùng không muốn cài đặt phần mềm hay sử dụng các dòng lệnh, mà chỉ cần thực hiện thao tác đơn giản trên trình duyệt để thu thập phụ đề một cách nhanh chóng. Tuy nhiên các công cụ này đều có chung hạn chế là trong thời gian ngắn không thể thực hiện truy vấn nhiều và tốc độ truy vấn của các trang này chậm hơn nhiều so với các cách khác. Dữ liệu phụ đề sẽ phụ thuộc vào Youtube cung cấp.

***Khai thác phụ đề bằng phương pháp Web Scraping kết hợp với tiện ích của Google:***

Khá tương đồng với phương pháp sử dụng các trang web trực tuyến là đều khai thác phụ đề video Youtube đơn giản thì đối với phương pháp này người dùng sẽ phải cài đặt các tiện ích của Google hỗ trợ cho việc xem video Youtube. Thay vì không cần phải vào trực tiếp video Youtube như phương pháp pháp sử dụng các trang web trực tuyến thì phương pháp này yêu cầu ta phải vào trực tiếp video Youtube và bắt đầu sử dụng các tiện ích của Google này. Một số tiện ích của Google có thế sử dụng để khai thác phụ đề Youtube như: *Youtube Subtiles Speaker and Translator, Find Subtitles for Youtube, Transcripts & Summary.*



Hình 2.7. Tiện ích Transcripts & Summary hiển thị phụ đề.

Các tiện ích này đề sẽ cho ta có thể trích xuất phụ đề của video Youtube qua 1 khung văn bản ở cạnh video. Từ đó ta có thể sao chép, tải về và sử dụng. Phương pháp này đảm bảo được tốc độ và đơn giản khi sử dụng. Tuy nhiên điểm yêu của phương pháp này là phải truy cập trực tiếp vào các video Youtube nên yêu cầu tốc độ mạng phải đảm bảo mới có thể đảm bảo thời gian khai thác có thể nhanh. Dữ liệu phụ đề sẽ phụ thuộc vào Youtube cung cấp.

***Khai thác phụ đề bằng tải video Youtube về kết hợp với speech-to-text:***

Khai thác phụ đề bằng phương pháp tải video Youtube về và sử dụng công nghệ chuyển đổi giọng nói thành văn bản (speech-to-text) là một phương pháp sáng tạo, mang lại hiệu quả cao cho phép người dùng có thể lấy phụ đề của bất cứ video nào.

Hiện nay có rất nhiều phương pháp nhận dạng giọng nói, tiêu biểu như sử dụng các thư viện có sẵn. VD: thư viện Whisper của OpenAI là một trong những giải pháp mạnh mẽ và phổ biến nhất hiện nay. Whisper hỗ trợ nhận dạng giọng nói và dịch thuật ở nhiều ngôn ngữ khác nhau, có khả năng xử lý những video có âm thanh không hoàn hảo hoặc nhiều giọng nói chồng chéo. Cụ thể, các phiên bản nhẹ hơn như Whisper Fast hoặc Whisper Small (như FastWhisper) có thể được sử dụng để tối ưu tốc độ chuyển đổi trên những hệ thống cấu hình thấp.

Hoặc người dùng có thể sử dụng các thuật toán, mô hình mà bản thân tự cấu hình để phù hợp với từng yêu cầu bài toán.

Điểm mạnh của phương pháp này có thể thấy rõ rệt vượt trội hơn với các phương pháp trên là dữ liệu phụ đề sẽ không bị lệ thuộc vào Youtube cung cấp mà sẽ chủ động hơn trong việc trích xuất phụ đề. Tuy nhiên phương pháp này đòi hỏi người dùng phải có kiến thức chuyên sâu về speech-to-text, và cần thời gian để cấu hình 1 mô hình đảm bảo độ chính xác tốt nhất và phù hợp với nhiều loại ngôn ngữ của từng video Youtube.

**2.1.3.Tiền xử lý dữ liệu phụ đề Tiếng Việt.**

Đối với dữ liệu phụ đề thu thập được sẽ tập trung chủ yếu vào dữ liệu phụ đề bằng tiếng việt. Đối với các phụ đề tiếng nước ngoài thì sẽ sử dụng **Translate** để phiên dịch lại thành Tiếng Việt.

Các phương pháp tiền xử lý dữ liệu đóng vai trò quan trọng trong quá trình phân tích, trích xuất thông tin và giúp cải thiện chất lượng của dữ liệu phụ đề Tiếng Việt, một số bước trong quá trình tiền xử lý bao gồm:

**Chuyển đổi thành chữ thường:** Toàn bộ văn bản trong bình luận được chuyển đổi thành chữ thường để tránh nhầm lẫn giữa các từ có cùng chữ viết nhưng khác biệt về ký tự hoa thường.

**Loại bỏ khoảng trắng dư thừa:** Các khoảng trắng không cần thiết trong phụ đề được loại bỏ để làm cho dữ liệu trở nên gọn gàng hơn.

**Loại bỏ liên kết:** Các liên kết không có ý nghĩa trong phụ đề được loại bỏ để tập trung vào nội dung chính.

**Chuẩn hóa Unicode:** Đảm bảo rằng các từ tiếng Việt được chuẩn hóa thành một định dạng Unicode chung như UTF-8 để tránh sự nhầm lẫn do sự không đồng nhất trong biểu diễn của các ký tự.

**Loại bỏ ký tự dư thừa:** Các ký tự không cần thiết mà có thể được người dùng nhập vào như dấu cách dư thừa được loại bỏ.

**Chuẩn hóa ký tự có dấu:** Các quy tắc được áp dụng để chuẩn hóa dấu thanh trong tiếng Việt, giúp đồng nhất các từ và dễ dàng xử lý.

**Tách từ:** Sử dụng bộ tách từ của thư viện NLP underthesea để chia câu thành các đơn vị từ hoặc cụm từ có ý nghĩa.

**Loại bỏ từ dừng (stopwords):** Các từ phổ biến không mang lại nhiều ý nghĩa trong phân tích được loại bỏ để tập trung vào các từ quan trọng hơn.

Việc thực hiện các bước tiền xử lý này không chỉ giúp cải thiện chất lượng của dữ liệu mà còn tạo ra các dữ liệu chuẩn để sử dụng trong việc huấn luyện mô hình phân loại, từ đó nâng cao hiệu suất và độ chính xác của hệ thống phân loại.

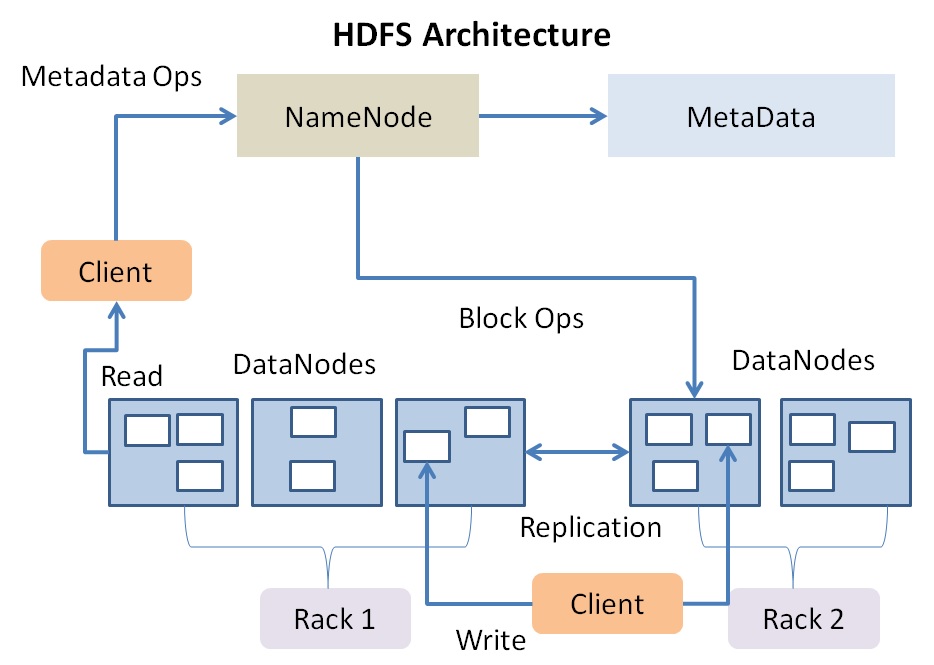
**2.2. Kỹ thuật lưu trữ dữ liệu lớn.**

**2.2.1. Hadoop Distributed File System.**

Khi dữ liệu người dùng ngày càng lớn và kích thước của các file dữ liệu ngày càng vượt quá giới hạn lưu trữ. Lúc này, nhu cầu phân chia dữ liệu ra các ổ cứng trên nhiều máy tính sẽ rất cần thiết. Và HDFS đã ra đời để hỗ trợ giải quyết triệt để được vấn đề này. HDFS (tên viết tắt của từ Hadoop Distributed File System) là một hệ thống lưu dữ dữ dữ liệu được sử dụng bởi Hadoop. Chức năng của hệ thống này là cung cấp khả năng truy cập với hiệu suất cao đến với các dữ liệu nằm trên các cụm của Hadoop.

***Kiến Trúc HDFS:***

Là một hệ thống lưu trữ được thiết kế để hỗ trợ việc ghi dữ liệu một lần trên một máy chủ và cho phép đọc nhiều lần sau đó từ bất kỳ máy chủ nào khác trong cụm. Cấu trúc của HDFS gồm một Namenode chính, đóng vai trò quản lý không gian lưu trữ và điều khiển truy cập dữ liệu, cùng với nhiều Datanode kết nối, làm nhiệm vụ lưu trữ dữ liệu thực tế. Sự phối hợp giữa Namenode và Datanodes tạo thành một cụm HDFS, nơi dữ liệu được lưu trữ một cách an toàn và có sẵn cho các hoạt động đọc tập trung hoặc phân tán.



Hình 2.8. Tổng quát kiến trúc HDFS.

***Master Node (NameNode):***

Master Node quản lý siêu dữ liệu (metadata) của hệ thống, bao gồm thông tin về vị trí, kích thước và trạng thái của từng khối dữ liệu. NameNode duy trì namespace, bảng chứa tất cả tên file và thư mục trong HDFS, và xử lý các hoạt động như tạo, xóa và di chuyển file và thư mục.

***Slave Nodes (DataNode):***

Slave Nodes (DataNode) lưu trữ các khối dữ liệu và chịu trách nhiệm báo cáo về trạng thái của dữ liệu được lưu trữ trên nó đến NameNode. Khi có yêu cầu ghi hoặc đọc dữ liệu, DataNode thực hiện các thao tác này dựa trên hướng dẫn từ NameNode. Quy trình lưu trữ dữ liệu trên HDFS bao gồm việc chia nhỏ dữ liệu thành các khối có kích thước mặc định (ví dụ, 128 MB hoặc 256 MB) , phân tán các khối dữ liệu trên nhiều DataNode trong hệ thống và nhân bản mỗi khối dữ liệu để lưu trữ trên nhiều DataNode khác nhau nhằm đảm bảo độ tin cậy và khả năng chịu lỗi. Thông tin về vị trí và sao chép của từng khối dữ liệu được ghi vào file edits và duyệt qua bởi NameNode.

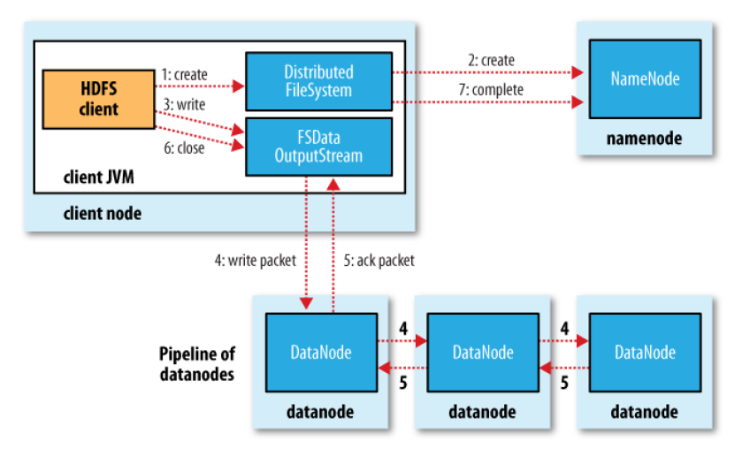
***Điểm đặc biệt của HDFS:***

HDFS có khả năng mở rộng linh hoạt bằng cách thêm máy chủ vào cụm mà không làm ảnh hưởng đến hiệu suất và khả năng chịu lỗi. Dữ liệu trên HDFS được nhân bản trên ít nhất hai DataNode khác nhau để đảm bảo độ tin cậy. Trong trường hợp một DataNode hoặc một số DataNode gặp sự cố, HDFS tự động phục hồi bằng cách sử dụng các bản sao dự trữ. Hơn nữa, HDFS được thiết kế để hoạt động tốt với framework xử lý dữ liệu phân tán MapReduce.

***Ghi dữ liệu HDFS:***

Quy trình ghi dữ liệu trong HDFS bắt đầu khi client gửi yêu cầu tạo file qua DistributedFileSystem APIs. DistributedFileSystem sau đó yêu cầu tạo file từ NameNode, nơi kiểm tra quyền truy cập của client và xác định xem file mới này có tồn tại hay không.

Khi yêu cầu được chấp thuận, DistributedFileSystem trả lại cho client một đối tượng FSDataOutputStream. FSDataOutputStream chứa DFSOutputStream, một thành phần quan trọng giúp xử lý giao tiếp giữa NameNode và DataNode. Khi client bắt đầu ghi dữ liệu, DFSOutputStream chia dữ liệu thành nhiều gói nhỏ và đưa chúng vào hàng đợi DataQueue.



Hình 2.9. Quy trình ghi dữ liệu HDFS.

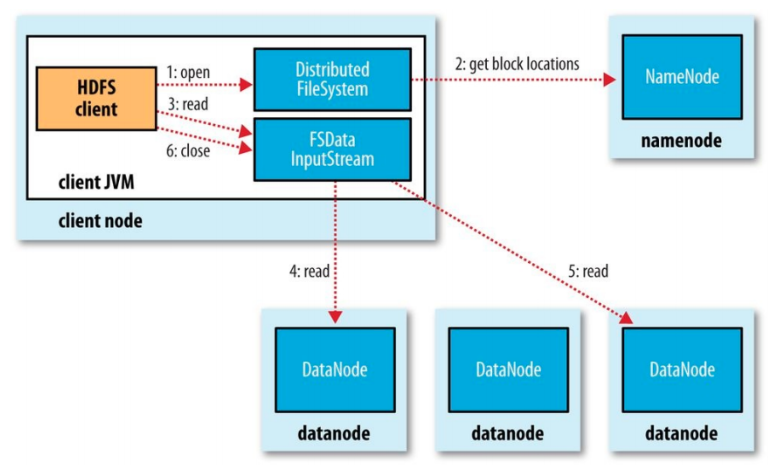
DataStreamer, một quá trình nền khác, sẽ liên lạc với NameNode để phân bổ các khối dữ liệu vào các DataNode nhằm lưu trữ bản sao. Các DataNode này tạo thành một chuỗi pipeline, mỗi DataNode trong chuỗi tương ứng với một bản sao của file. DataStreamer gửi các gói dữ liệu tới DataNode đầu tiên trong pipeline, từ đó các gói được chuyển tiếp đến các DataNode tiếp theo.

Trong khi dữ liệu đang được gửi, DFSOutputStream quản lý một hàng đợi gọi là Ack Queue để theo dõi các gói chưa được xác nhận bởi các DataNode. Khi tất cả các DataNode trong pipeline xác nhận đã nhận được gói dữ liệu, gói đó mới được loại bỏ khỏi Ack Queue.

Cuối cùng, khi quá trình ghi kết thúc, client sẽ gọi hàm close () để đóng luồng dữ liệu, đẩy những gói còn lại vào pipeline. Sau khi tất cả các gói đã được ghi vào các DataNode và đã nhận được xác nhận từ tất cả, một thông báo hoàn thành ghi file được gửi lên để kết thúc quá trình.

***Đọc dữ liệu HDFS:***

Quá trình mở và đọc file trong HDFS bắt đầu khi client sử dụng phương thức open từ đối tượng FileSystemObject. DistributedFileSystem sau đó yêu cầu NameNode cung cấp vị trí các block của file cần mở. NameNode trả lời bằng cách cung cấp địa chỉ của các DataNode nơi lưu trữ các bản sao của từng block.



Hình 2.10. Quy trình đọc dữ liệu HDFS.

Một khi đã có các địa chỉ này, một đối tượng FSDataInputStream được cấp cho client. FSDataInputStream này bao gồm DFSInputStream, quản lý toàn bộ quá trình I/O giữa client và các DataNode cũng như NameNode.

Client tiến hành đọc dữ liệu bằng cách gọi phương thức “read () ” trên FSDataInputStream. DFSInputStream tìm DataNode gần nhất chứa block đầu tiên của file và thiết lập kết nối để bắt đầu quá trình đọc. Để đảm bảo liên tục và hiệu quả, phương thức “read () ” sẽ được thực hiện lặp đi lặp lại cho đến khi đọc hết block.

Khi block hiện tại được đọc xong, DFSInputStream sẽ ngắt kết nối với DataNode đó và tìm DataNode tiếp theo chứa block kế tiếp của file. Nếu trong quá trình đọc xảy ra lỗi, DFSInputStream sẽ tự động chuyển sang DataNode khác gần nhất có chứa block đó để tiếp tục đọc.

Cuối cùng, sau khi đọc toàn bộ file, client sẽ gọi phương thức “close () ” để kết thúc phiên đọc, giải phóng mọi tài nguyên và ngắt kết nối đã thiết lập. Quy trình này đảm bảo rằng dữ liệu được đọc một cách hiệu quả, với tính sẵn sàng cao và khả năng chịu lỗi mạnh mẽ của HDFS.

***Ưu điểm và nhược điểm của HDFS:***

HDFS có nhiều ưu điểm nổi bật, bao gồm khả năng mở rộng dễ dàng, cho phép xử lý lượng dữ liệu lớn mà không gặp vấn đề về mở rộng bằng cách thêm vào các máy chủ mới. Tính chịu lỗi cao của HDFS được đảm bảo nhờ sao lưu dữ liệu thành nhiều bản sao trên nhiều máy chủ, giúp hệ thống luôn sẵn sàng. HDFS cũng có hiệu suất cao cho việc đọc và ghi dữ liệu, đặc biệt là với các tệp tin lớn. Dữ liệu trong HDFS được phân tán trên nhiều máy chủ, tận dụng tối đa tài nguyên và tăng tốc độ xử lý. Hơn nữa, HDFS tích hợp chặt chẽ với các thành phần khác của hệ sinh thái Hadoop, như MapReduce và HBase, tạo điều kiện thuận lợi cho việc xử lý và phân tích dữ liệu lớn.

HDFS có một số nhược điểm, bao gồm độ trễ khi truy cập dữ liệu ngẫu nhiên do dữ liệu được phân tán trên nhiều máy chủ. HDFS không thích hợp cho việc xử lý và lưu trữ các tệp tin nhỏ vì nó được tối ưu hóa cho dữ liệu lớn. HDFS cũng gặp khó khăn trong việc xử lý dữ liệu real-time do có thể gây ra độ trễ. Dung lượng lưu trữ trong HDFS thường điều chỉnh theo quy mô của hệ thống, điều này có thể làm tăng chi phí nếu cần lưu trữ dữ liệu lớn. Cuối cùng, triển khai HDFS yêu cầu một hạ tầng lớn với nhiều máy chủ để đảm bảo tính mở rộng và khả năng chịu lỗi.

**2.2.2. Cơ sở dữ liệu trong bộ nhớ.**

Cơ sở dữ liệu trong bộ nhớ (In-Memory Database, viết tắt là **IMDB**) là một loại cơ sở dữ liệu trong đó dữ liệu được lưu trữ và truy cập trực tiếp từ RAM (bộ nhớ truy cập ngẫu nhiên) thay vì từ đĩa cứng. Điều này giúp tăng tốc độ truy xuất dữ liệu một cách đáng kể so với cơ sở dữ liệu truyền thống, vốn lưu trữ dữ liệu trên đĩa cứng. Lý do cơ sở dữ liệu trong bộ nhớ nhanh hơn:

**Truy cập RAM nhanh hơn nhiều so với truy cập đĩa cứng**: RAM cho phép truy cập ngẫu nhiên đến dữ liệu, trong khi đĩa cứng thường phải mất thời gian tìm kiếm (seek time) để đọc hoặc ghi dữ liệu.

**Tránh việc I/O đĩa (Input/Output):** Việc đọc/ghi dữ liệu từ đĩa cần phải thông qua quá trình I/O đĩa, làm tăng độ trễ. Trong khi đó, với IMDB, tất cả thao tác đều diễn ra trong RAM, giúp tốc độ truy cập và xử lý nhanh hơn rất nhiều.

**Dữ liệu được cập nhật tức thời:** Khi có thay đổi dữ liệu, cơ sở dữ liệu trong bộ nhớ có thể cập nhật ngay lập tức mà không phải đợi lưu xuống đĩa cứng, giúp dữ liệu luôn đồng bộ và nhất quán.

**Ưu điểm và nhược điểm của IMDB:**

Cơ sở dữ liệu trong bộ nhớ có nhiều ưu điểm. Hiệu suất cao là một trong số đó, vì dữ liệu được lưu trữ trực tiếp trong bộ nhớ RAM, cung cấp tốc độ truy cập rất nhanh. Thời gian đáp ứng ngắn do không có độ trễ khi truy cập dữ liệu từ đĩa cứng, giúp các truy vấn diễn ra nhanh chóng. Các ứng dụng đòi hỏi xử lý nhanh và liên tục của dữ liệu, như hệ thống giao dịch trực tuyến, thường hưởng lợi từ cơ sở dữ liệu trong bộ nhớ. Cơ sở dữ liệu này cũng phù hợp cho việc xử lý và truy vấn dữ liệu phức tạp mà không làm giảm hiệu suất. Ngoài ra, với việc giảm chi phí bộ nhớ, một số cơ sở dữ liệu trong bộ nhớ có khả năng mở rộng tốt.

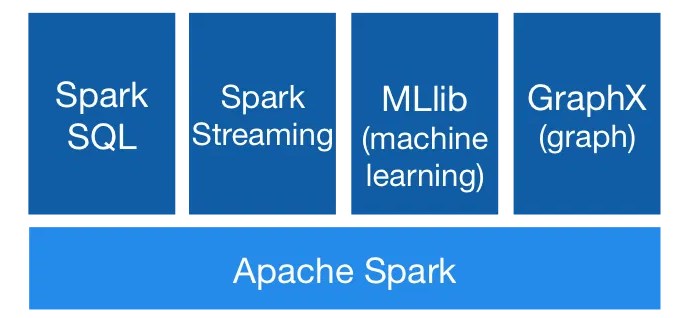
Tuy nhiên, cơ sở dữ liệu trong bộ nhớ cũng có nhược điểm. Có giới hạn về khả năng lưu trữ so với cơ sở dữ liệu truyền thống do dữ liệu được lưu trữ trực tiếp trong bộ nhớ. Bộ nhớ RAM thường đắt đỏ, do đó triển khai và duy trì cơ sở dữ liệu này có thể gặp chi phí cao. Khả năng mở rộng cũng có thể gặp hạn chế khi dữ liệu tăng kích thước. Dữ liệu cần được tải vào bộ nhớ trước khi xử lý, điều này có thể tạo ra thời gian đầu tư khi khởi động hệ thống. Cuối cùng, đồng bộ hóa dữ liệu giữa các bộ nhớ và lưu trữ có thể là một thách thức, đặc biệt khi có các cập nhật liên tục.

**2.3. Tổng quan về Apache Spark.**

Apache Spark, một framework mã nguồn mở tiên tiến, đã nhanh chóng trở thành điểm đến đắc lực cho việc xử lý và phân tích dữ liệu lớn trong thời đại big data hiện nay. Với sự thiết kế chủ đạo là tốc độ tính toán và khả năng mở rộng linh hoạt, Spark đã chứng minh vai trò quan trọng trong các tổ chức và công ty trên khắp thế giới.

Apache Spark nổi bật với khả năng cung cấp tốc độ tính toán ấn tượng cho dữ liệu lớn [7] và khả năng mở rộng linh hoạt. Nhờ vào tính toán phân tán, Spark giúp giảm thời gian xử lý, mang lại hiệu suất cao và khả năng xử lý dữ liệu linh hoạt trên cụm máy tính.

Trong một thời đại mà dữ liệu lớn trở thành nguồn tài nguyên quan trọng, Apache Spark đã chứng minh sự đa năng của mình. Hiện nay, Spark không chỉ được sử dụng mạnh mẽ trong lĩnh vực kinh doanh, mà còn mở rộng ra các lĩnh vực như y tế, tài chính, giáo dục, khoa học, và công nghệ. Một số thành phần trong Apache Spark như sau.



Hình 2.11. *Cấu trúc của Apache Spark.*

**Spark Core:** Spark Core là thành phần cốt lõi của Apache Spark, chịu trách nhiệm quản lý cụm máy tính và cung cấp các chức năng cơ bản.

Spark cung cấp khả năng xử lý dữ liệu phân tán (DDP) , quản lý bộ nhớ, thiết lập lịch tác vụ và khôi phục lỗi.

**Spark SQL:** Spark SQL cho phép truy vấn dữ liệu bằng ngôn ngữ SQL trên các tập dữ liệu lớn. Tích hợp tốt với cơ sở dữ liệu quan hệ như MySQL, Oracle và PostgreSQL. Ngoài ra còn hỗ trợ tích hợp với công cụ Extract, Transform, Load và Spark Streaming.

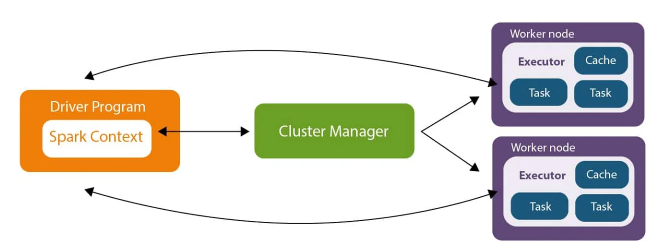
**Spark Streaming:** Spark Streaming là module xử lý dữ liệu trực tiếp trong thời gian thực (real-time). Sử dụng cơ chế xử lý dữ liệu liên tục (stream processing) thông qua microbatch. Tích hợp với Spark SQL và MLlib, phù hợp cho phân tích dữ liệu trực tuyến và giám sát hệ thống.

**Spark MLlib (Machine Learning Library):** Spark MLlib là thư viện máy học tích hợp trong Apache Spark. Cung cấp nhiều thuật toán máy học như Regression, Classification, Clustering, Collaborative Filtering, Dimensionality Reduction, Feature Extraction, và Transformation. Tích hợp tốt với các module khác của Apache Spark.

**GraphX:** GraphX là thành phần xử lý cấu trúc đồ thị, sử dụng RDD, VertexRDD, và EdgeRDD. Cung cấp các thuật toán phân tán cho xử lý đồ thị, thích hợp cho các ứng dụng liên quan đến mạng xã hội và quan hệ đối tượng. Những thành phần này cùng nhau tạo nên một hệ sinh thái mạnh mẽ, giúp Apache Spark trở thành một công cụ đa nhiệm và hiệu quả trong việc xử lý dữ liệu lớn.

***Kiến trúc của Apache Spark:***

Kiến trúc của Apache Spark được tổ chức một cách phân tán và có các thành phần chính đóng vai trò quan trọng trong quá trình xử lý dữ liệu lớn.



Hình 2.12. Mô tả chi tiết về kiến trúc của Apache Spark

**Driver Program:**  Là chương trình chính của ứng dụng Spark chạy trên một node trong cluster. Quản lý và điều khiển quá trình xử lý trên toàn bộ cluster. Tạo và duy trì Spark Context, bao gồm tất cả các chức năng cơ bản.

**Spark Context:** Được tạo và duy trì bởi Driver Program. Bao gồm các chức năng cơ bản như quản lý tài nguyên, thiết lập kết nối với cluster, và tạo Resilient Distributed Datasets (RDDs).

**Cluster Manager:** Hỗ trợ quản lý và phân phối tài nguyên trên các node của cluster. Phân phối và giám sát tiến trình xử lý trên các node để đảm bảo hiệu suất và hiệu quả.

**Resilient Distributed Datasets (RDDs):** Cấu trúc dữ liệu phân tán và bất biến được tạo trong Spark Context. Phân phối trên nhiều worker nodes và có thể được cached ở đó để tối ưu hiệu suất.

**Worker Nodes:** Thực thi nhiệm vụ được giao bởi Cluster Manager. Mỗi worker node có thể chứa nhiều Executor để xử lý các công việc đồng thời.

**Executors:** Tiến trình chạy trên các worker nodes, được quản lý bởi Driver Program. Thực hiện các nhiệm vụ xử lý dữ liệu được giao. Mỗi Executor có thể chứa nhiều task (công việc) để tận dụng tối đa tài nguyên. Kiến trúc này cho phép Apache Spark xử lý dữ liệu phân tán một cách hiệu quả, tận dụng khả năng tính toán đồng thời trên nhiều node trong cluster, giúp tăng cường khả năng mở rộng và hiệu suất của hệ thống.

***Các công ty lớn sử dụng Apache Spark:***

Apache Spark đã trở thành một framework xử lý dữ liệu phân tán phổ biến và được sử dụng rộng rãi trong nhiều lĩnh vực.



Hình 2.13. Các công ty lớn nổi bật sử dụng Apache Spark.

|  |  |
| --- | --- |
|  | **Yelp:** Sử dụng Apache Spark để tạo mô hình dự đoán và xác định tương tác của người dùng với quảng cáo. Xử lý lượng lớn dữ liệu trên Amazon EMR, giúp tăng doanh thu và tỷ lệ nhấp vào quảng cáo.  **Zillow:** Sở hữu và điều hành một trong những trang web bất động sản trực tuyến lớn nhất thế giới. Sử dụng Apache Spark trên Amazon EMR để xử lý dữ liệu lớn trong thời gian thực, đặc biệt trong việc tính toán Zestimates - công cụ ước tính định giá nhà.  **CrowdStrike:** Cung cấp khả năng bảo vệ điểm cuối để ngăn chặn vi phạm. Sử dụng Apache Spark trên Amazon EMR để xử lý hàng trăm terabyte dữ liệu sự kiện và phân tích hành vi cấp cao trên máy chủ để phát hiện sự hiện diện của hoạt động độc hại.  **Hearst Corporation:** Công ty đa ngành với truyền thông là mảng lớn nhất. Sử dụng Apache Spark Streaming trên Amazon EMR để theo dõi hiệu suất bài viết và chủ đề thịnh hành theo thời gian thực trên web.  **Intent Media:** Sử dụng Apache Spark và MLlib để đào tạo và triển khai các mô hình máy học trên quy mô lớn. Hỗ trợ các công ty du lịch tối ưu hóa doanh thu trên trang web và ứng dụng của họ thông qua việc áp dụng các mô hình máy học. Những công ty này là ví dụ minh họa cho sự linh hoạt và khả năng ứng dụng của Apache Spark trong nhiều lĩnh vực khác nhau, từ dự đoán người dùng đến phân tích dữ liệu thời gian thực và tối ưu hóa doanh thu. |

Hình 2.14. Một số công ty nổi tiếng sử dụng Apache Spark.

2.4. Các công trình nghiên cứu liên quan.

Trong nghiên cứu của Aslam [2], [3] đã khám phá vai trò của điện toán đám mây trong xử lý [4], [8], lưu trữ và quản lý dữ liệu lớn [19]. Bài báo bắt đầu bằng việc định nghĩa cloud computing và mối quan hệ với big data. Các lợi ích của việc sử dụng cloud computing cho xử lý và lưu trữ big data cũng được phân tích, cùng với những thách thức đi kèm khi triển khai điện toán đám mây [15], [17], [27] cho dữ liệu lớn.

Nghiên cứu “Efficient Query Analysis and Performance Evaluation of the Nosql Data Store for BigData” của S. Gupta, et al. [14] đã trình bày một cách hiệu quả để lưu trữ và phân tích dữ liệu lớn [1], [30] bằng cách sử dụng hệ thống lưu trữ dữ liệu Nosql. Dữ liệu lớn từ các ứng dụng web và mạng xã hội [22] có cấu trúc không đồng nhất, không phù hợp với cách sử dụng cơ sở dữ liệu quan hệ truyền thống. Nghiên cứu này giới thiệu một cách tiếp cận hiệu quả cho việc lưu trữ và phân tích dữ liệu lớn bằng cách sử dụng hệ thống lưu trữ dữ liệu Nosql và chứng minh tính hiệu quả của nó thông qua phân tích kết quả về việc truy vấn hiệu quả bằng cách đánh giá ước lượng hiệu suất trên các thao tác đọc và ghi trên các truy vấn đơn giản và phức tạp, cũng như về việc lưu trữ và truy xuất số lượng bản ghi ngày càng tăng. Kết quả cho thấy rằng hệ thống lưu trữ dữ liệu Nosql được chọn - Cassandra, hiệu quả hơn so với cơ sở dữ liệu quan hệ - mysql và các hệ thống lưu trữ dữ liệu Nosql khác - HBase và MongoDB.

Nghiên cứu của Kyrylo Kleshch [18] có tên “Development of fuzzy search method for creating an efficient information search system in text data” mục tiêu của nghiên cứu là quá trình tìm kiếm thông tin hiệu quả trong một tập hợp dữ liệu văn bản. Đối tượng của nghiên cứu là phương pháp tìm kiếm mờ, giúp giải quyết vấn đề tìm kiếm thông tin trong một tập hợp dữ liệu văn bản một cách hiệu quả. Dựa trên phương pháp này, đề xuất tạo ra một hệ thống tìm kiếm mờ sẽ giải quyết vấn đề tìm kiếm các tài liệu có liên quan nhất từ một tập hợp các tài liệu như vậy.

MapReduce là mô hình lập trình phân tán dành cho việc xử lý khối lượng lớn dữ liệu [19], [7]. MapReduce là phương pháp hiệu quả để phân cụm các tập truy vấn lớn trong một khoảng thời gian hợp lý. Apache Hadoop và Apache Spark [7] là hai trong số các khung công tác phổ biến nhất được sử dụng để thực thi các chương trình MapReduce. Apache Hadoop là một khung công tác chủ yếu được sử dụng cho việc xử lý phân tán và mở rộng trong môi trường đám mây [12], [26]. Hadoop xử lý khối lượng dữ liệu lớn bằng cách chia các nhiệm vụ thành các nhiệm vụ nhỏ hơn [19], [23]. Nó thường xuyên ghi và đọc dữ liệu từ đĩa, điều này có thể làm giảm hiệu suất của hệ thống. Tuy nhiên, Apache Spark cung cấp các tính toán trong bộ nhớ để giảm số lần đọc/ghi trên đĩa [13].

Phương pháp tần suất thuật ngữ (TF) [28] và đo lường cosine với biểu diễn đặc điểm của ngôn ngữ truy vấn SQL được sử dụng trong phương pháp đề xuất kế hoạch truy cập được trình bày. Trong bài viết hiện tại, một mô hình MapReduce song song được áp dụng để tăng tốc hoạt động phân cụm truy vấn trong Apache Hadoop [25]. Hơn nữa, hiệu suất của phương pháp đề xuất kế hoạch truy cập được trình bày [24] được cải thiện bằng cách thực hiện trong Apache Spark, đây là một trình xử lý dữ liệu phân tán trong bộ nhớ.

**2.5. Kết luận Chương 2.**

Chương 2 đã trình bày một nền tảng toàn diện về các khái niệm và công nghệ then chốt trong việc lưu trữ, phân loại, cũng như tìm kiếm dữ liệu lớn từ phụ đề Youtube. Chương bắt đầu với việc phân tích cấu trúc phụ đề và các phương pháp khai thác hiệu quả, đồng thời làm nổi bật quy trình tiền xử lý dữ liệu nhằm cải thiện độ chính xác và hiệu suất xử lý. Các công nghệ tiên tiến như HDFS và Apache Spark được giới thiệu như những công cụ mạnh mẽ để quản lý, xử lý khối lượng lớn dữ liệu một cách nhanh chóng và hiệu quả. Những kiến thức này không chỉ tạo nền tảng lý thuyết mà còn mở đường cho việc triển khai ứng dụng tìm kiếm toàn diện và tối ưu trong chương 3 để giải quyết bài toán.

**Chương 3  
XÂY DỰNG ỨNG DỤNG LƯU TRỮ VÀ TÌM KIẾM DỮ LIỆU LỚN   
PHỤ ĐỀ** **YOUTUBE.**

**3.1. Yêu cầu ứng dụng.**

Để xây dựng và phát triển ứng dạng tự động thu thập, lưu trữ, và tìm kiếm dữ liệu phụ đề từ Youtube, cần có sự đầu tư kỹ lưỡng vào công nghệ, nhân lực và tài nguyên. Đồng thời, để đảm bảo tính chính xác và hiệu quả của ứng dụng cần đáp ứng một số yêu cầu sau:

**Thu thập dữ liệu đa nguồn và liên tục:** Hệ thống phải có khả năng thu thập dữ liệu phụ đề từ nhiều nguồn trên Youtube, dựa trên các kênh, từ khóa được chỉ định. Quá trình này phải diễn ra một cách liên tục và tự động để cập nhật thông tin mới nhất, đảm bảo tính đa dạng và phong phú của dữ liệu.

**Khả năng tự động cập nhật:** Hệ thống cần tự động cập nhật các dữ liệu phụ đề mới nhất từ các video được tải lên trên Youtube, đảm bảo rằng thông tin được cung cấp cho người dùng luôn là những tin tức mới nhất và chính xác nhất.

**Mở rộng dễ dàng:** Hệ thống cần có khả năng mở rộng linh hoạt để đáp ứng được sự tăng trưởng của dữ liệu mà không làm gián đoạn hoạt động. Việc mở rộng này phải được thực hiện một cách dễ dàng và hiệu quả.

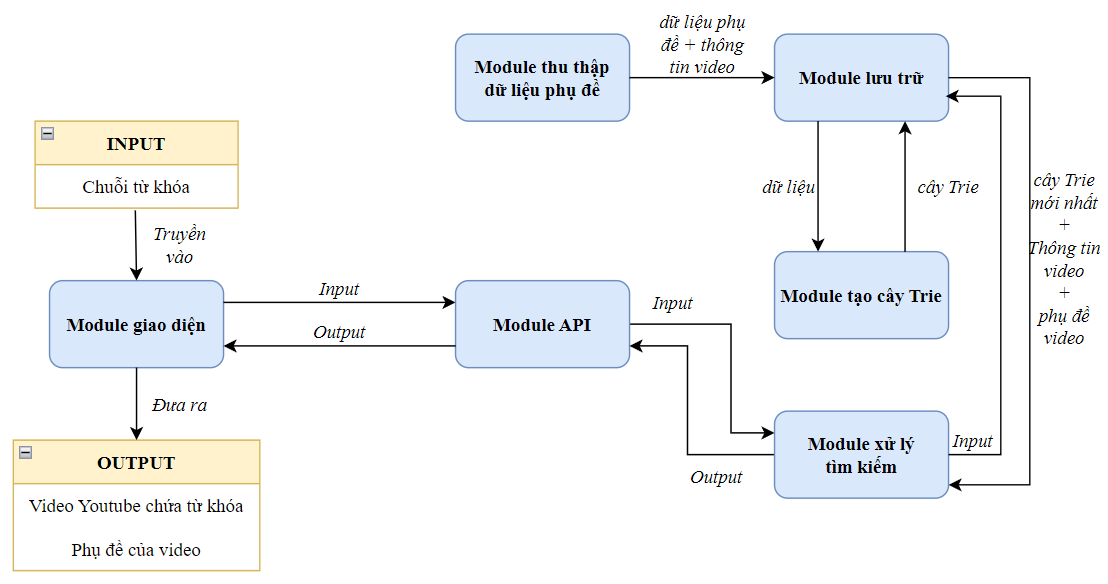
**Chi phí hiệu quả:** Thiết kế của hệ thống cần đảm bảo tính khả thi và hiệu quả kinh tế, đảm bảo rằng việc triển khai và duy trì hệ thống không gây quá nhiều chi phí không cần thiết.

**Hoạt động liên tục:** Hệ thống phải hoạt động liên tục mà không gặp sự gián đoạn, đảm bảo rằng người dùng có thể truy cập vào dữ liệu phụ đề bất kỳ lúc nào mà họ cần.

**Đảm bảo tính an toàn và bảo mật:** Hệ thống cần có các biện pháp bảo mật mạnh mẽ để bảo vệ dữ liệu phụ đề khỏi bị truy cập trái phép hoặc thất thoát. Đồng thời, cần có các biện pháp sao lưu và phục hồi dữ liệu để đảm bảo tính an toàn và khả năng phục hồi khi có sự cố xảy ra.

**Giao diện người dùng thân thiện:** Hệ thống cần có một giao diện dễ sử dụng, trực quan, giúp người dùng dễ dàng tìm kiếm và truy cập vào dữ liệu phụ đề mà không cần kiến thức kỹ thuật chuyên sâu.

3.2. Mô hình hoạt động

****

Hình 3.1. Mô hình hoạt động của ứng dụng

**Input:** Đầu vào của ứng dụng sẽ là chuỗi các từ khóa. Các từ khóa sẽ cách nhau bằng dấu phẩy ‘, ’. Ví dụ như“Đảng, bạo loạn, chiến tranh, Trung Quốc, Tô Lâm”. Input sẽ được nhập từ Module giao diện.

**Output:** Đầu ra của ứng dụng sẽ là các video Youtube chứa các từ khóa đưa vào và phụ đề của các video đó. Đầu ra sẽ được gửi lại cho giao diện người dùng thông qua API.

**Module giao diện** (Client Interface): Đây là nơi người dùng nhập chuỗi từ khóa để thực hiện tìm kiếm. Chuỗi từ khóa này sẽ được truyền đến Module API để xử lý tiếp theo. Sau khi quá trình xử lý hoàn thành, kết quả bao gồm video Youtube chứa từ khóa và phụ đề của video sẽ được trả về và hiển thị cho người dùng qua giao diện.

**Module API:** Đây là trung gian giữa Module giao diện và các module xử lý trên server. API sẽ nhận input từ khóa từ người dùng và truyền đến các module xử lý bên dưới. Sau khi tìm kiếm hoàn tất, output kết quả sẽ được truyền ngược lại từ các module xử lý đến Module API, rồi từ đó trả về giao diện cho người dùng.

**Module thu thập dữ liệu phụ đề (Subtitle Data Collection):** Module này chịu trách nhiệm thu thập và cập nhật dữ liệu phụ đề từ Youtube hoặc các nguồn dữ liệu khác. Dữ liệu phụ đề sẽ được cung cấp liên tục và đưa vào Module lưu trữ để lưu trữ và xử lý.

**Module lưu trữ (Storage Module):** Sau khi nhận được dữ liệu từ Module thu thập, module này sẽ lưu trữ phụ đề vào hệ thống lưu trữ (HDFS). Đồng thời, dữ liệu từ module này cũng được chuyển đến Module tạo cây Trie để cập nhật cây tìm kiếm.

**Module tạo cây Trie (Trie Creation Module):** Module này chịu trách nhiệm tạo và cập nhật cấu trúc cây Trie dựa trên dữ liệu phụ đề đã được lưu trữ.Cây Trie này sẽ giúp quá trình tìm kiếm diễn ra nhanh hơn và hiệu quả hơn bằng cách tổ chức các từ khóa theo cấu trúc phân cấp. Cây Trie mới nhất sẽ liên tục được cập nhật và sử dụng trong quá trình xử lý tìm kiếm.

**Module xử lý tìm kiếm (Search Processing Module):** Khi từ khóa được truyền từ API, module này sẽ sử dụng cây Trie để thực hiện tìm kiếm nhanh chóng và chính xác. Kết quả tìm kiếm sẽ bao gồm các video Youtube chứa từ khóa cùng với phụ đề tương ứng. Sau khi hoàn tất, kết quả này sẽ được trả lại Module API để gửi về giao diện cho người dùng.

***Mô tả quy trình hoạt động***

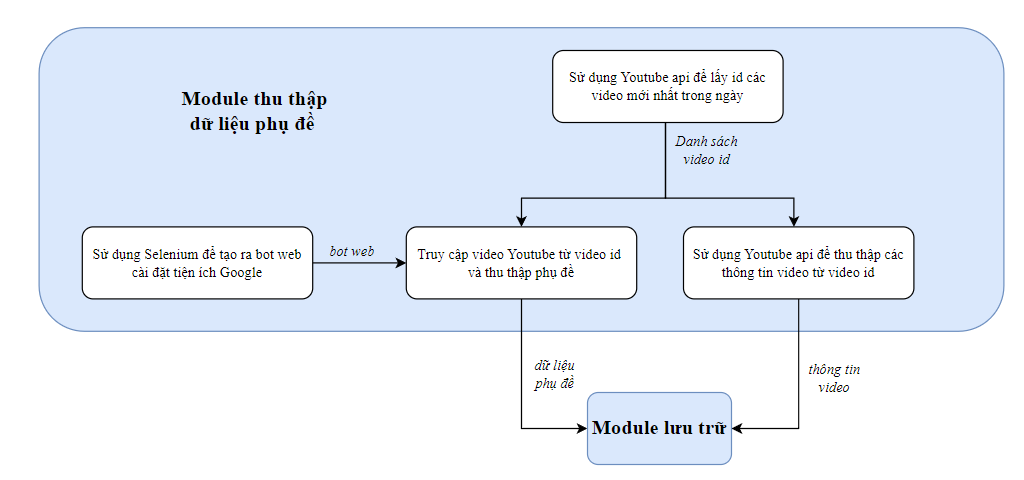
* Người dùng nhập từ khóa tìm kiếm vào giao diện.
* Từ khóa được truyền qua API tới module xử lý.
* Hệ thống thu thập và lưu trữ dữ liệu phụ đề, đồng thời liên tục cập nhật cây Trie để tối ưu hóa tìm kiếm.
* Khi có yêu cầu tìm kiếm, cây Trie đã được cập nhật sẽ giúp module xử lý tìm kiếm nhanh chóng, trả lại kết quả video và phụ đề cho người dùng.

**3.3. Xây dựng Module thu thập dữ liệu phụ đề.**

Thu thập dữ liệu phụ đề trên Youtube là một quá trình quan trọng trong lĩnh vực nghiên cứu và phát triển các ứng dụng liên quan đến ngôn ngữ tự nhiên và trí tuệ nhân tạo. Youtube, với hàng tỷ giờ video được tải lên mỗi ngày, là một nguồn tài nguyên khổng lồ về dữ liệu văn bản phong phú. Dữ liệu phụ đề, được tạo ra tự động hoặc bởi người dùng, cung cấp thông tin về nội dung của video, từ các lời thoại cho đến mô tả nội dung.

***Mô hình tổng quan thu thập phụ đề:***

Việc thu thập phụ đề của video Youtube ta sẽ sử dụng cách Khai thác phụ đề bằng phương pháp Web Scraping kết hợp với tiện ích của Google kết hợp với sử dụng Youtube Data API v3 để lấy thêm các thông tin của video.



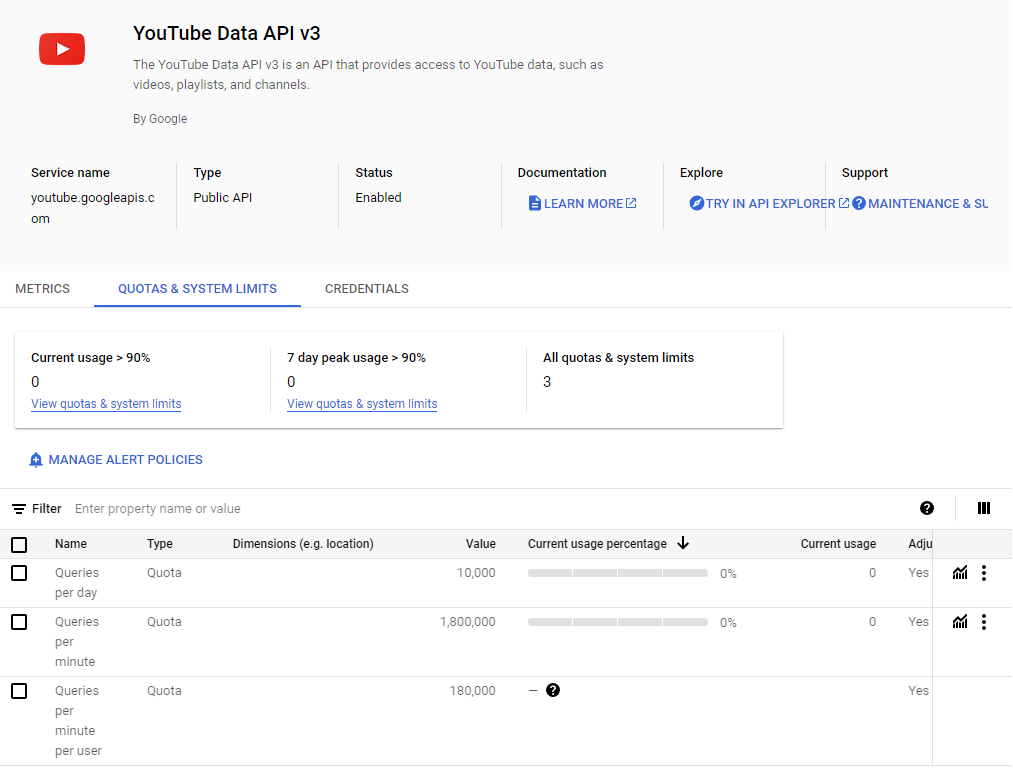
Hình 3.2. Mô hình tổng quan của Module thu thập dữ liệu phụ đề

***Mô tả quy trình hoạt động***

**Bước 1.** Đầu tiên ta cần kết nối với Youtube Data API v3 để lấy API KEY từ Google console.

\*\*Đối với mỗi tài khoản thì 1 ngày, ta chỉ có thể sử dụng tối đa là 10.000 điểm truy vấn. Mỗi loại yêu cầu truy vấn sẻ sử dụng điểm này. Mỗi loại yêu cầu khác nhau thì sẽ tiêu tốn một số điểm truy vấn khác nhau. VD: **List Videos** (Liệt kê video): Mất **1 điểm** mỗi yêu cầu. **Search API** (Tìm kiếm video): Mất **100 điểm** mỗi yêu cầu. Nên ta sẽ sử dụng nhiều tài khoản để lập nhiều API KEY để sử dụng.

Sau khi có được API KEY thì ta có thể sử dụng nó để gọi đến Youtube api lấy dữ liệu về. Đầu tiên ta sẽ sử dụng **Search API** để đưa ra các video mới nhất trong ngày của các channel tin tức nổi tiếng (Báo Tuổi trẻ, BBC news Tiếng Việt, 24h bản tin, v.v..). Bước này sẽ đưa ra cho 1 list danh sách video\_id của các video Youtube mới nhất.

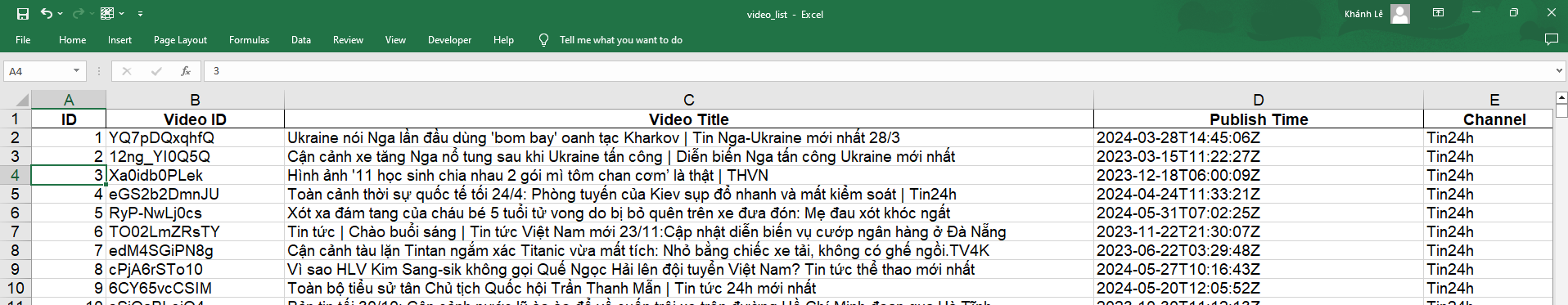


Hình 3.3. Giao diện Youtube Data API v3 trên Google Clouds

**Bước 2.**Sau khi đã có 1 list danh sách video\_id thì ta sẽ tiến hành thu thập các thông tin của video và phụ đề của video từ từng video\_id. Hai quá trình này có thể thực hiện song song chứ không cần phải luân phiên hay đợi lần lượt. Với mỗi video\_id thì ta sẽ gắn với thuộc tính id để tiện xử lý.

***Thu thập thông tin video:***

Dựa vào video\_id ta sẽ sử dụng **List Videos** để thu thập các thông tin video: Tiêu đề video, ngày xuất bản, kênh xuất bản. Sau đó ta sẽ lưu các thông tin này thành dạng bảng gồm các thông tin: ID, Video ID, Video Title, Publish Time, Channel. Sau đó lưu bảng này vào 1 file excel *video\_list.xlsx .* Tiếp đó ta sẽ lưu file *video\_list.xlsx* bằng **Module lưu trữ.**

****

Hình 3.4. File excel video\_list.xlsx

***Thu thập phụ đề video:***

Ta sẽ sử dụng phương pháp **Web Scraping kết hợp với tiện ích của Google** để thu thập dữ liệu phụ đề. Cụ thể:

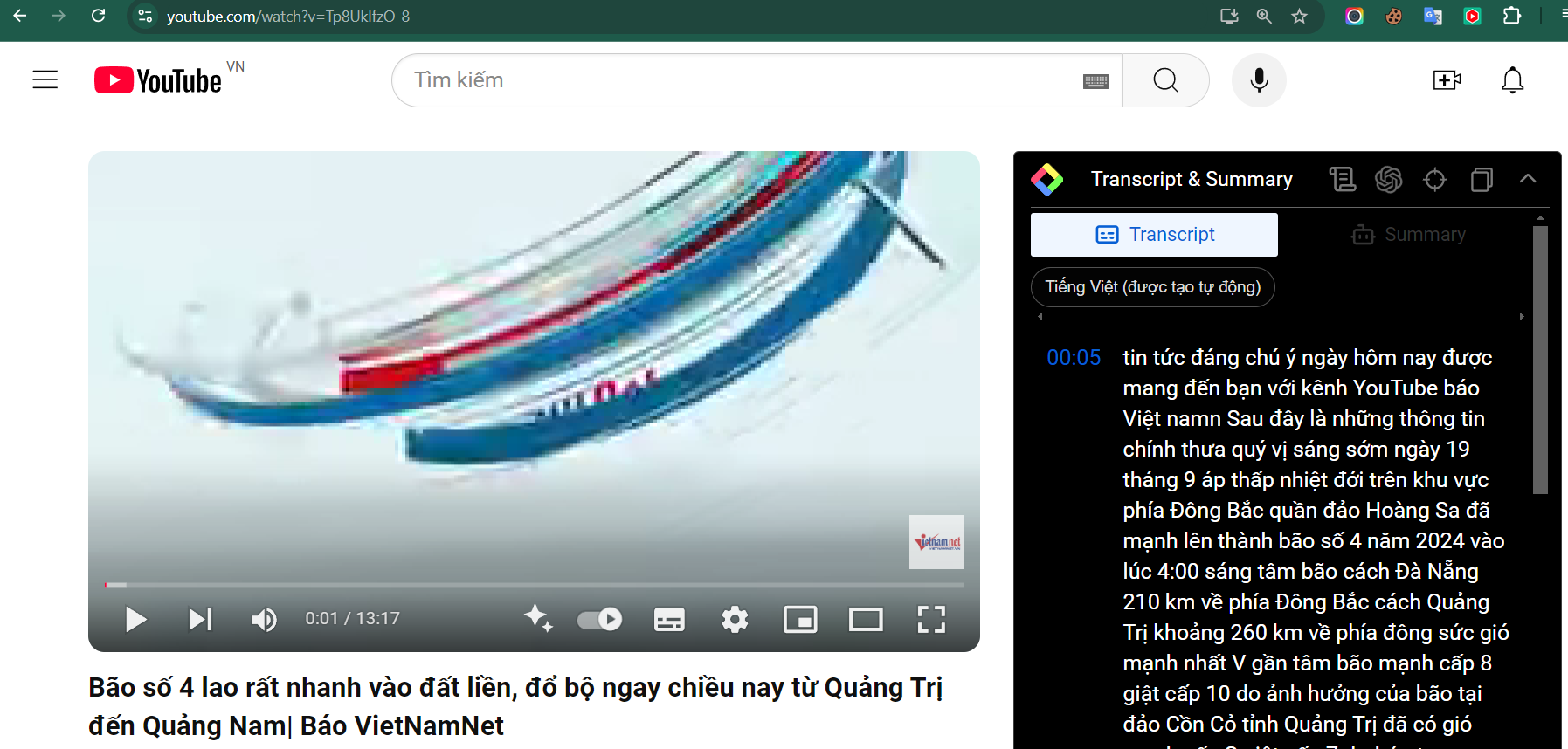
Web Scraping: **Selenium.**

 Tiện ích của Google Extensions:  **Glasp Web Highlighter: PDF & Web Highlight.**

Cách thức thu thập: Trước tiên ta sử dụng **Selenium** để tạo ra một bot web sau đó cài đặt tiện ích **Glasp Web Highlighter: PDF & Web Highlight.** Sau khi cài đặt xong thì ta cài đặt cho bot tự động thực thi vòng lặp:

1. Truy cập vào video Youtube từ video\_id.

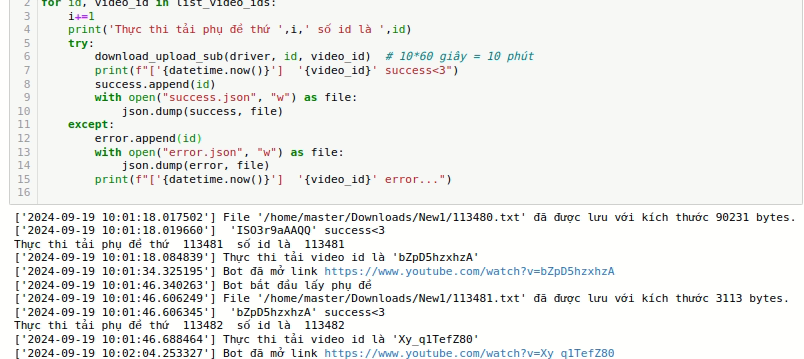
2. Nhấn vào tiện ích **Transcript & Summary (**Đây là 1 tiện ích nhỏ của tiện ích Glasp).



Hình 3.5. Box hiển thị phụ đề bằng Glasp trên nền tảng Youtube.

3. Thu thập phụ đề từ box Transcript & Summary sau đó lưu vào file .txt với tên file là id của video đó. VD: id = 11348 thì tên file là 11348.txt. Đối với các file chứa phụ đề thì chỉ lấy những phụ đề nào có kích thước lớn hơn 200 bytes, còn lại thì sẽ xóa đi.

Sau đó đẩy file này qua cho **Module Lưu trữ.**



Hình 3.6. Đoạn kết quả thực thi chạy bot thu thập dữ liệu

3.4. Xây dựng Module tạo cây Trie.

Quá trình xây dựng cây Trie sẽ đóng vai trò chính và phức tạp nhất trong bài toán này. Ta cần tạo cây phù hợp với yêu cầu bài toán nhanh và gọn gàng.

3.4.1. Lựa chọn công cụ phân tán hiệu quả.

GraphX là một công cụ mạnh mẽ trong việc xử lý các đồ thị phân tán trên nền tảng Apache Spark, nhưng nó được thiết kế chủ yếu để xử lý các đồ thị có cấu trúc phức tạp, như mạng xã hội, các hệ thống giao thông, hoặc các quan hệ giữa các nút và cạnh. Trie, ngược lại, là một cấu trúc cây đơn giản được sử dụng để quản lý các chuỗi ký tự hoặc từ điển từ ngữ, do đó, trong thực tế, việc sử dụng GraphX để tạo cây Trie không phổ biến.

Hiện nay, các công cụ và thư viện khác như RDD (Resilient Distributed Dataset) trong Spark hoặc các cấu trúc dữ liệu khác như B-tree, Radix Tree, hoặc HashMap thường được ưa chuộng hơn khi làm việc với Trie. Những công cụ này được tối ưu cho việc quản lý dữ liệu đơn giản, có thể xử lý hàng triệu từ khóa hoặc chuỗi ký tự một cách hiệu quả mà không cần đến việc sử dụng các mô hình đồ thị phức tạp. Đối với bài toán hiện tại ta sẽ sử dụng thư viện RDD trong Spark để tạo cây.

3.4.2. Cấu trúc cây Trie.

Trong khoa học máy tính, trie, hay cây tiền tố, là một cấu trúc dữ liệu sử dụng cây có thứ tự, dùng để lưu trữ một mảng liên kết của các xâu ký tự. Không như cây nhị phân tìm kiếm, mỗi nút trong cây không liên kết với một khóa trong mảng. Thay vào đó, mỗi nút liên kết với một xâu ký tự sao cho các xâu ký tự của tất cả các nút con của một nút đều có chung một tiền tố, chính là xâu ký tự của nút đó. Nút gốc tương ứng với xâu ký tự rỗng. Các bước để chọn cấu trúc cây tiền tố:

***Bước 1. Chọn loại nút cho cây Trie.***

Cây Trie theo yêu cầu bài toán sẽ lưu trữ các từ tiếng việt. Nên 1 số khái niệm cần biết về tiếng việt:

- "k", "h", "á", "n" được gọi là các ký tự lẻ (chữ cái).

- "kh", "khá", "khán", "khánh" là âm tiết.

Có nhiều cách tạo cây Trie, tuy nhiên để phù hợp vơi yêu cầu bài toán thì có 2 cách chính để tạo cây:

|  |  |
| --- | --- |
|  | **Cách 1. Chỉ lưu chữ cái ở các nút.**  Cách này phù hợp với yêu cầu bài toán và tiết kiệm được chi phí. Cây sẽ chỉ tập trung lâu các nút chứ không phải lưu cả cành. Kích thước nút nhỏ. Tuy nhiên khi dính phải lỗi trong quá trình khởi tạo thì rất khó để khắc phục. Ví dụ: *Khi lỗi ở nút c thì không biết rõ là nút c nào lỗi.* |
| Hình 3.7. Cấu trúc cây Trie lưu trữ chữ cái | |
|  | **Cách 2.Lưu trữ âm tiết của từ ở các nút.**  Cách này tương tự cách trên chỉ khác là ở nút chúng ta sẽ lưu trữ 1 phần của từ. Cách này thì khiến cho lượng dữ liệu lưu trữ lớn hơn cách 1, nhưng phải số lượng dữ liệu rất lớn mới có thể thấy rõ được điểm khác biệt. Cách này thì dễ kiểm tra lỗi để khắc phục hơn.  Đối với nghiên cứu và thực nghiệm thì sẽ thường sử dụng cách này hơn.  \* Đối với bài toán lần này, ta sẽ sử dụng cách này để có thể dễ hình dung hơn. |
| Hình 3.8. Cấu trúc cây Trie lưu trữ âm tiết | |

***Bước 2. Lựa chọn lưu trữ từ đơn hay từ ghép vào cây***

Đối với bài toán dữ liệu lớn thì khối lượng các từ đơn là rất lớn. Nếu ta sử dụng từ ghép để lưu trữ thì sẽ khối lượng tăng lên gấp bội lần. Ví dụ:

Giả sử bạn có các từ sau:

* Từ đơn: "người", "yêu", "thương"
* Từ ghép: "người yêu", "thương yêu"

Đây là 1 ví dụ đơn giản về từ đơn và từ ghép. Tuy nhiên từ ghép có rất nhiều từ được ghép lại từ nhiều hơn hai từ đơn. Mặc dù có lợi hơn trong việc ngữ nghĩa của từ nhưng lại khiến cho số lượng âm tiết tăng lên rất nhiều lần. Khiến cho tài nguyên để lưu trữ cây trở nên hạn chế hơn. Cây Trie chủ yếu được sử dụng để lưu trữ và tìm kiếm các từ hoặc chuỗi có cùng tiền tố. Từ ghép có thể được hình thành từ nhiều từ đơn, và việc chỉ lưu từ đơn giúp dễ dàng tra cứu hoặc xây dựng các từ ghép từ những từ đơn đã có trong Trie. Khi cần tìm một từ ghép, chỉ cần tìm các từ đơn cấu thành, thay vì lưu trữ và tìm kiếm riêng các từ ghép, điều này có thể làm giảm thời gian tìm kiếm.

* + Ta sẽ tạo cây Trie từ các từ đơn. Ngoài ra để đảm bảo cho việc tìm kiếm từ ghép thì ta sẽ lưu trữ chính xác (lưu trữ vị trí từ ở trong đoạn văn) để đảm bảo cho quá trình tìm từ được nhanh hơn.

***Bước 3. Quá trình hình thành cấu trúc dữ liệu ở nút của cây***

Để tạo ra 1 cây Trie bằng rdd thì ta sẽ lưu trữ ở nút của cây thì ta sẽ đặt cho mỗi nút 1 biến id. Dữ liệu ở nút sẽ có dạng: *[id, name]* trong đó name là âm tiết cần lưu.

Để các âm tiết của 1 từ có thể liên kết được với nhau thì ta sẽ đặt thêm nút danh sách id của các âm tiết sau. Dữ liệu ở nút sẽ có dạng:

*[id, name, list[dst, name]] Trong đó list[dst, name\_dst]* là danh sách lưu trữ id (dst) và name (name\_dst) của các âm tiết sau âm tiết hiện tại.

Ta cần biết name của nút hiện tại có chứa trong đoạn văn nào không. Ví dụ đang ở nút *“khán”* thì có chứa trong đoạn văn chứa từ *“khán giả”* nên ta sẽ thêm 1 danh sách lưu id của đoạn văn. Ngoài ra với bài toán hiện tại ta sử dụng phương pháp tìm kiếm chính xác nên sẽ yêu cầu thêm việc lưu trữ vị trí của từng từ trong đoạn văn nên ta sẽ thêm vào nút 1 danh sách chứa id của đoạn văn và danh sách các vị trí xuất hiện của từ trong đoạn văn.Dữ liệu ở nút sẽ có dạng:

*[id,*

*name,*

*list[dst, name\_dst],*

*list[id đoạn văn, list[vị trí xuất hiện của từ]]*

VD về 1 nút sẽ có dạng:

[850,

'khán',

[1050, 'khánh'],

[ (50210, [593]) , (1441, [29, 123]) , (58619, [488]) , (51578, [187]) , (50016, [56, 217, 315]) , (55845, [745, 765, 777, 908]) , (6571, [1467]) , (30148, [35]) , (54698, [431]) , (47480, [55, 750]) , (2165, [69]) , (31349, [75]) , (9935, [580]) , (50620, [165]) , (40681, [123]) ]]

Ta có thể hiểu nút có id là 850, chứa âm tiết 'khán', nút sau có id là 1050, chứa âm tiết 'khánh', có ở vị trí 593 của đoạn văn 50210, vị trí 29 và 123 của đoạn văn 1441,.v.v...

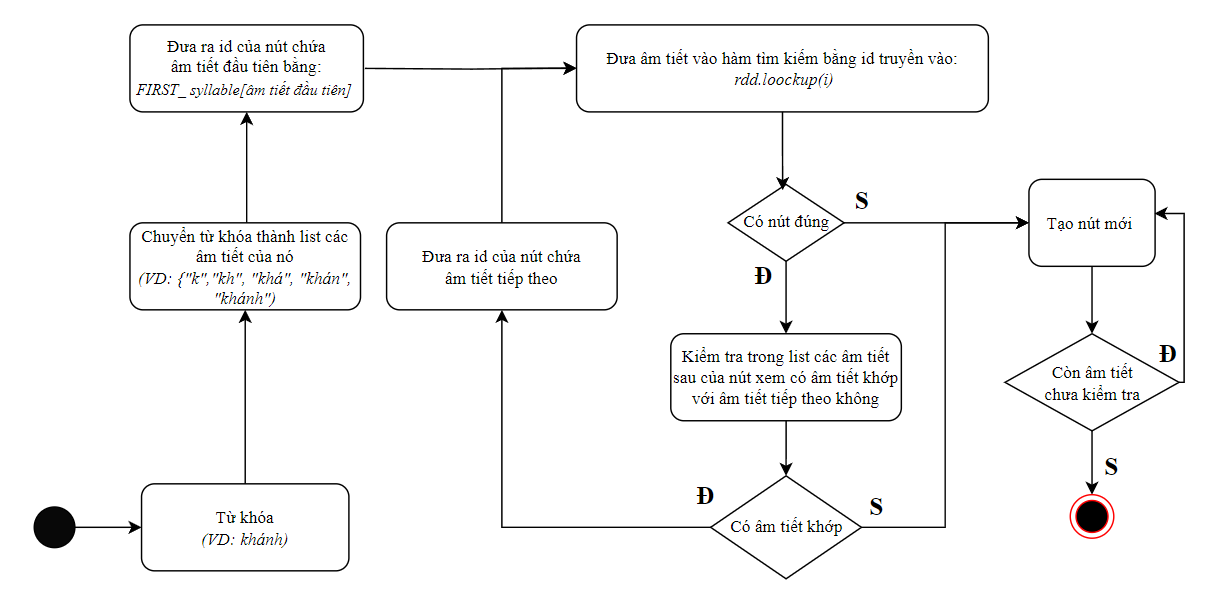
***Bước 4. Chọn cách tạo cây với RDD***

Có nhiều cách tạo ra 1 cây đồ thị từ rdd, tuy nhiên đối với bài toán này ta có thể sử dụng phương pháp tạo cây Trie từ 1 rdd duy nhất. Rdd này sẽ lưu trữ tất cả các nút của cây. Cách tìm kiếm nút chứa âm tiết bất kì thì sử dụng hàm filter của spark: *rdd.filter (lambda x: x == ‘âm tiết cần tìm’).collect ().* Dùng hàm filter thì sẽ cho kết quả đợi rất lâu và để phù hợp cây Trie thì ta phải đưa ra cách để tìm ra vị trí của âm tiết đầu tiên. Nên ta tạo ra list *FIRST\_* *syllable* đểlưu trữ tất cả các chữ cái tiếng việt có dấu, không có dấu, chữ cái tiếng anh và chữ số:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, đ, e, g, h, i, k, l, m, n, o, p, q, r, s, t, u, v, x, y, à, á, ả, ã, ạ, â, ầ, ấ, ẩ, ẫ, ậ, ă, ằ, ắ, ẳ, ẵ, ặ, è, é, ẻ, ẽ, ẹ, ê, ề, ế, ể, ễ, ệ, ì, í, ỉ, ĩ, ị, ò, ó, ỏ, õ, ọ, ô, ồ, ố, ổ, ỗ, ộ, ơ, ờ, ớ, ở, ỡ, ợ, ù, ú, ủ, ũ, ụ, ư, ừ, ứ, ử, ữ, ự, ỳ, ý, ỷ, ỹ, ỵ, f, j, w, z]

Sắp xếp danh sách theo thứ tự trên để đưa vào cây là ta đã có vị trí của chữ cái đầu thay vì sử dụng hàm filter. Giờ tôi muốn đưa ra vị trí của chữ cái k chỉ cần tôi để là *FIRST\_* *syllable[‘k’]*

Quy trình thêm 1 từ vào trong cây cụ thể như sau:



Hình 3.9. Quy trình thêm từ khóa vào cây Trie

Ngoài ra để tối ưu việc tìm kiếm ta có thể sử dụng cách tạo rdd phân tầng. Ta sẽ tạo ra số lượng rdd ứng với số lượng chữ cái của 1 từ. Ta sẽ tạo ra n rdd:

RDD1 chỉ chứa các âm tiết có 1 chữ cái

RDD2 chỉ chứa các âm tiết có 2 chữ cái

..........

..........

RDDn chỉ chứa các âm tiết có n chữ cái

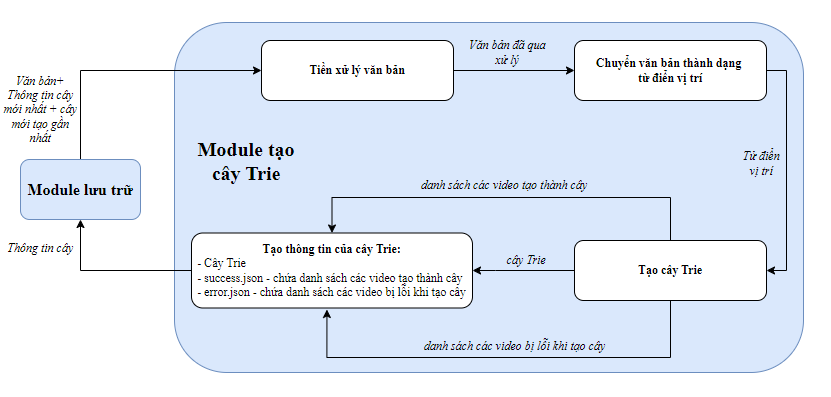
*Điểm mạnh:* Khi sử dụng hàm lookup thì ta sẽ chỉ sử dụng trong các rdd đã chia nhỏ nên tốc độ tìm kiếm sẽ nhanh hơn gấp n lần

*Điểm yếu:* Quá trình khởi tạo và lưu trữ sẽ phức tạp và tốn nhiều bộ nhớ hơn. Đối với Tiếng Việt thì từ đơn dài nhất tối đa là 7 chữ cái. Nhưng nếu với Tiếng Anh thì số lượng chữ cái của một từ sẽ lớn hơn nên khả năng mở rộng của rdd rất hạn chế.

* + Nên ta sẽ chỉ sử dụng một rdd để lưu trữ.

3.4.3. Mô hình tạo cây

Sau khi ta hiểu rõ về cấu trúc cây Trie và cách thêm mới các nút cho cây Trie. Ta tiến hành tạo một cây Trie hoàn chỉnh từ các dữ liệu văn bản được lấy từ **Module lưu trữ** về. Từ **Module lưu trữ** ta cũng sẽ lấy về phiên bản cây Trie mới tạo gần đây nhất để cập nhật cho cây Trie của ta. Trước khi tạo cây Trie ta cần thực hiện một số thao tác tiền xử lý dữ liệu để đưa vào phù hợp với cấu trúc của cây dự định tạo.



Hình 3.10. Quy trình hoạt động của Module tạo cây Trie

***Mô tả quy trình hoạt động:***

**Bước 1.** Tiền xử lý văn bản.

Tiền xử lý nội dung phụ đề là một bước quan trọng trong quá trình xử lý dữ liệu văn bản, nhằm chuẩn bị dữ liệu cho các bước phân tích và ứng dụng tiếp theo. Quá trình này bao gồm các phương pháp và kỹ thuật để làm sạch, chuẩn hóa và trích xuất thông tin từ dữ liệu phụ đề, từ đó tạo ra một tập dữ liệu có cấu trúc và dễ dàng xử lý.

Trong tiền xử lý nội dung phụ đề, các bước chính có thể bao gồm loại bỏ ký tự đặc biệt, chuyển đổi văn bản về dạng chữ thường, loại bỏ từ ngữ không cần thiết như các dấu câu, từ dừng, hay từ ngữ thường gặp.

**Bước 2.** Chuyển văn bản thành dạng tử điển vị trí.

Từ điển vị trí là một cấu trúc dữ liệu lưu trữ các từ (hoặc cụm từ) và vị trí của chúng trong văn bản hoặc đoạn văn. Thay vì chỉ lưu trữ từ khóa, từ điển vị trí còn lưu lại vị trí mà các từ đó xuất hiện trong văn bản. Điều này giúp cải thiện khả năng tìm kiếm chính xác, đặc biệt là khi tìm kiếm theo cụm từ (phrase search).

|  |  |
| --- | --- |
|  | **Phân đoạn văn thành các từ:**  Đầu tiên, bạn cần tách đoạn văn thành các từ riêng lẻ. Trong tiếng Việt, việc tách từ cần đặc biệt chú ý đến từ ghép  **Xác định vị trí từng từ:**  Sau khi tách từ, mỗi từ sẽ được đánh dấu với vị trí của nó trong đoạn văn. Vị trí này có thể là chỉ số thứ tự của từ trong văn bản (dựa trên số lượng từ).  **Tạo từ điển vị trí:**  Mỗi từ sẽ được lưu trữ dưới dạng một khóa trong từ điển, và giá trị là danh sách các vị trí mà từ đó xuất hiện. |

Hình 3.11. Các bước tiến hành tiền xử lý dữ liệu.

**Bước 3.** Tạo cây Trie.

Dựa vào cấu trức cây Trie và phương pháp thêm mới các nút cho cây Trie. Ta lần lượt thêm từng từ khóa trong tử điển vị trí vào cây đến khi hết từ.

**Bước 4.** Tạo thông tin của cây Trie.

Ngoài việc tạo cây Trie, đồng thời trong quá trình thêm nếu bị gặp lỗi giữa chừng thì ta sẽ lập tức dừng lại và lưu lại id của đoạn văn bản vừa thêm lại vào file error.json. Còn nếu thêm hết thành công thì ta lưu id của đoạn văn bản vừa thêm lại vào file sussces.json.

Việc lưu lại 2 file error.json và sussces.json sẽ hỗ trợ ta trong việc kiểm soát thông tin id trong cây và đưa ra các id bị lỗi để từ đó kiểm tra và khắc phục.

**Bước 5**. Upload dữ liệu.

Ta đẩy cây Trie dưới dạng rdd và 2 file error.json và sussces.json vào **Module lưu trữ** để lưu trữ lại. Khi lưu thì ta lưu lại thông tin ngày tạo cây thành công và phiên bản của cây hiện tại.

3.5. Xây dựng Module lưu trữ.

Xây dựng hệ thống lưu trữ dữ liệu sử dụng Hadoop Distributed File System (HDFS) là một quyết định chiến lược trong việc quản lý và xử lý dữ liệu lớn. Với HDFS, dữ liệu được chia nhỏ thành các khối và phân tán trên nhiều nút của một cluster. Mỗi khối dữ liệu được nhân bản trên nhiều nút khác nhau để đảm bảo tính toàn vẹn và sẵn sàng khi có sự cố xảy ra. HDFS cung cấp một giao diện đơn giản và hiệu quả cho việc lưu trữ và truy xuất dữ liệu từ các ứng dụng.

Để hỗ trợ quá trình thử nghiệm, nghiên cứu này sử dụng cấu hình máy tính, máy chủ và phần mềm hỗ trợ được mô tả chi tiết trong Bảng 3.2, Bảng 3.3 và Bảng 3.4.

Bảng 3.1. Thông số phần cứng máy chủ VMware ESXi.

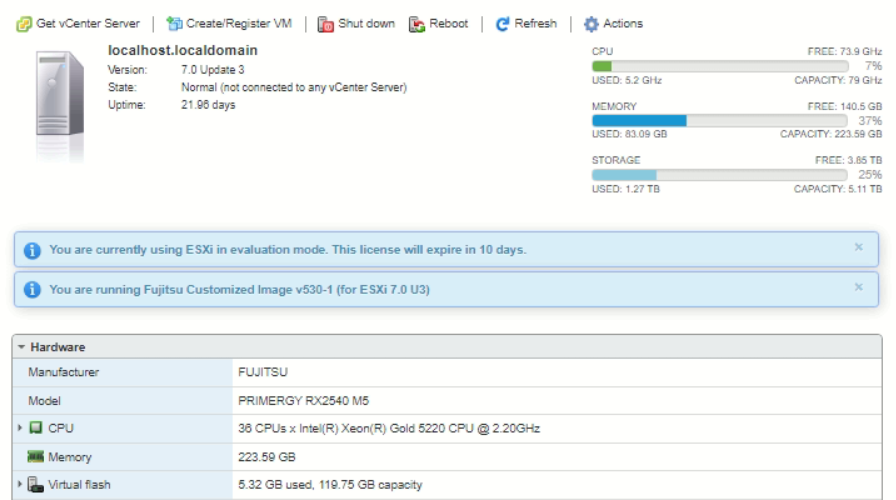
|  |  |  |
| --- | --- | --- |
| **STT** | **Thành phần** | **Thông số kỹ thuật** |
|  | Máy chủ ảo hóa | VMware ESXi 7.0 |
|  | CPU | 36 CPUs x Intel (R) Xeon (R) Gold 5220 CPU @ 2.20GHz |
|  | RAM | 223.59 GB |
|  | HDD | 5.11 TB |

Bảng 3.2. Thông tin máy chủ.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên máy** | **Địa chỉ IP** | **Cấu hình** | **Chức năng** |
|  | master | 10.32.112.39 | CPU: 3 vCPUs  RAM: 20GB HDD: 100GB | Máy chủ chính chạy Hadoop |
|  | slave01 | 10.32.112.40 | CPU: 1 CPUs  RAM: 6GB HDD: 100GB | Máy chủ thứ cấp 1 chạy Hadoop |
|  | slave02 | 10.32.112.41 | CPU: 1 CPUs  RAM: 6GB HDD: 100GB | Máy chủ thứ cấp 2 chạy Hadoop |
|  | slave03 | 10.32.112.86 | CPU: 1 CPUs  RAM: 6GB HDD: 100GB | Máy chủ thứ cấp 3 chạy Hadoop |
|  | slave04 | 10.32.112.87 | CPU: 1 CPUs  RAM: 6GB HDD: 100GB | Máy chủ thứ cấp 4 chạy Hadoop |

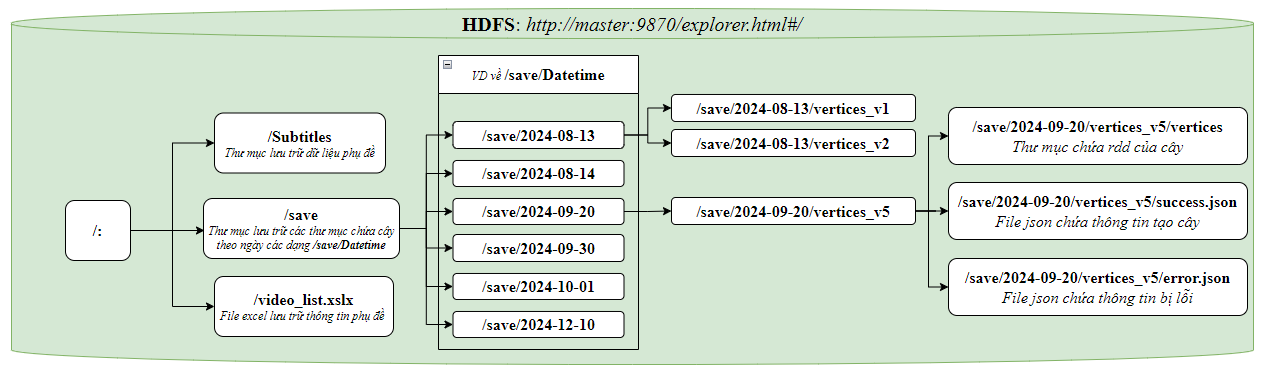
Bảng 3.3. Các công cụ phần mềm hỗ trợ

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên công cụ** | **Chức năng** |
|  | Hadoop 3.3.1 | Tính toán dữ liệu phân tán |
|  | Apache Spark 3.4 | Xử lý dữ liệu lớn mã nguồn mở |
|  | Python 3.8.10 | Ngôn ngữ lập trình |
|  | openjdk-8-jdk | Biến môi trường |
|  | Streamlit | Framework phát triển ứng dụng web bằng Python |



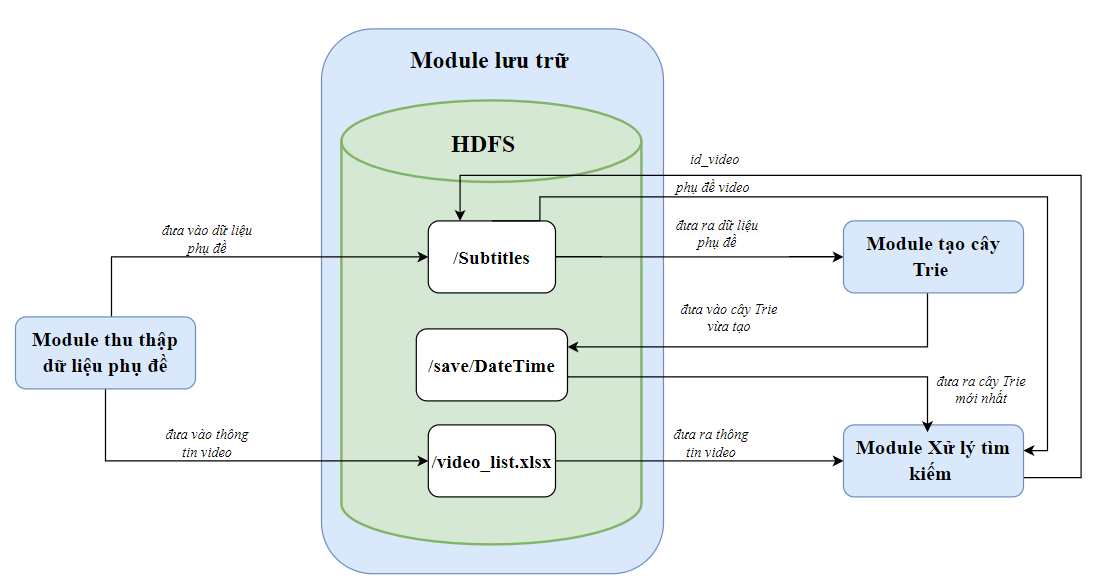
Hình 3.12. Giao diện quản trị máy chủ VMware ESXi.

Sau khi cấu hình máy như thông số kỹ thuật trên, ta tiến hành cấu hình HDFS thành các thư mục như sau:



Hình 3.13. Cấu trúc thư mục lưu trữ trên HDFS

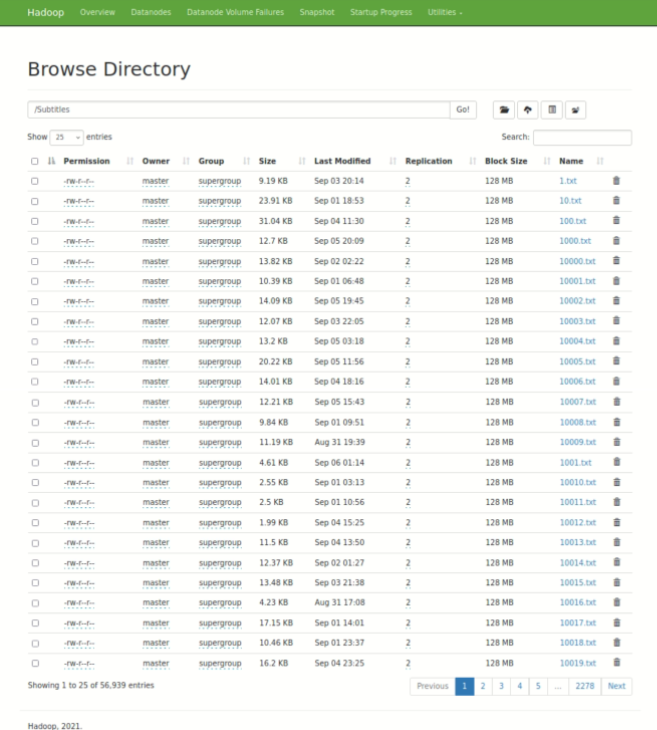
**Quy trình hoạt động**



Hình 3.14. Quy trình hoạt động của Module lưu trữ

**Mô tả quy trình hoạt động**

**Module thu thập dữ liệu** phụ đề sẽ đưa vào thư mục **/Subtitles** các phụ đề video được đánh số theo ID đã phân ban đầu, giúp dễ dàng quản lý và truy xuất sau này. Mỗi file phụ đề sẽ được đặt tên theo quy tắc nhất định để đảm bảo tính nhất quán, đồng thời tạo điều kiện cho việc tìm kiếm và tổ chức dữ liệu. Ngoài ra, hệ thống sẽ tự động đẩy file thông tin video **video\_list.xlsx** lên HDFS, chứa các thông tin chi tiết như ID video, tiêu đề, thời gian xuất bản, và kênh phát sóng. Quá trình này không chỉ giúp duy trì một kho dữ liệu có cấu trúc mà còn đảm bảo khả năng truy cập nhanh chóng và an toàn cho người dùng. **Module thu thập dữ liệu** sẽ đẩy dữ liệu cập nhật tự động theo lịch trình đã định, nhằm đảm bảo luôn có sẵn thông tin mới nhất cho các video được thu thập.



Hình 3.15. Danh sách tệp phụ đề .txt được lưu trữ trên HDFS

**Module tạo cây Trie** sẽ lấy lần lượt các file phụ đề video ở trong **/Subtitles** và xử lý tạo cây và lưu lại vào folder /**save.** Folder /**save** sẽ chứa các phiên bản cây theo ngày tạo ra. Ví dụ cây phiên bản 3, 4 tạo ra ngày 02-09-2024 thì sẽ được lưu vào **/save/2024-09-02/vertices\_v3** và **/save/2024-09-02/vertices\_v4.** Ở trong mỗi thư mục này sẽ chứa:

* Folder **/vertices** chứa rdd của cây Trie
* File **/success.json** chứa id của các video tạo ra cây Trie
* File **/error.json** chứa id các video đã lỗi trong quá trình tạo cây

**Module xử lý tìm kiếm** sẽ gọi đến Folder **/vertices** trong **/save/*{Datetime}/{newest\_version\_vertices}*** để lấy cây Trie về xử lý và lấy **/video\_list.xlsx** về để lấy thông tin của các video.

3.6. Xây dụng Module API.

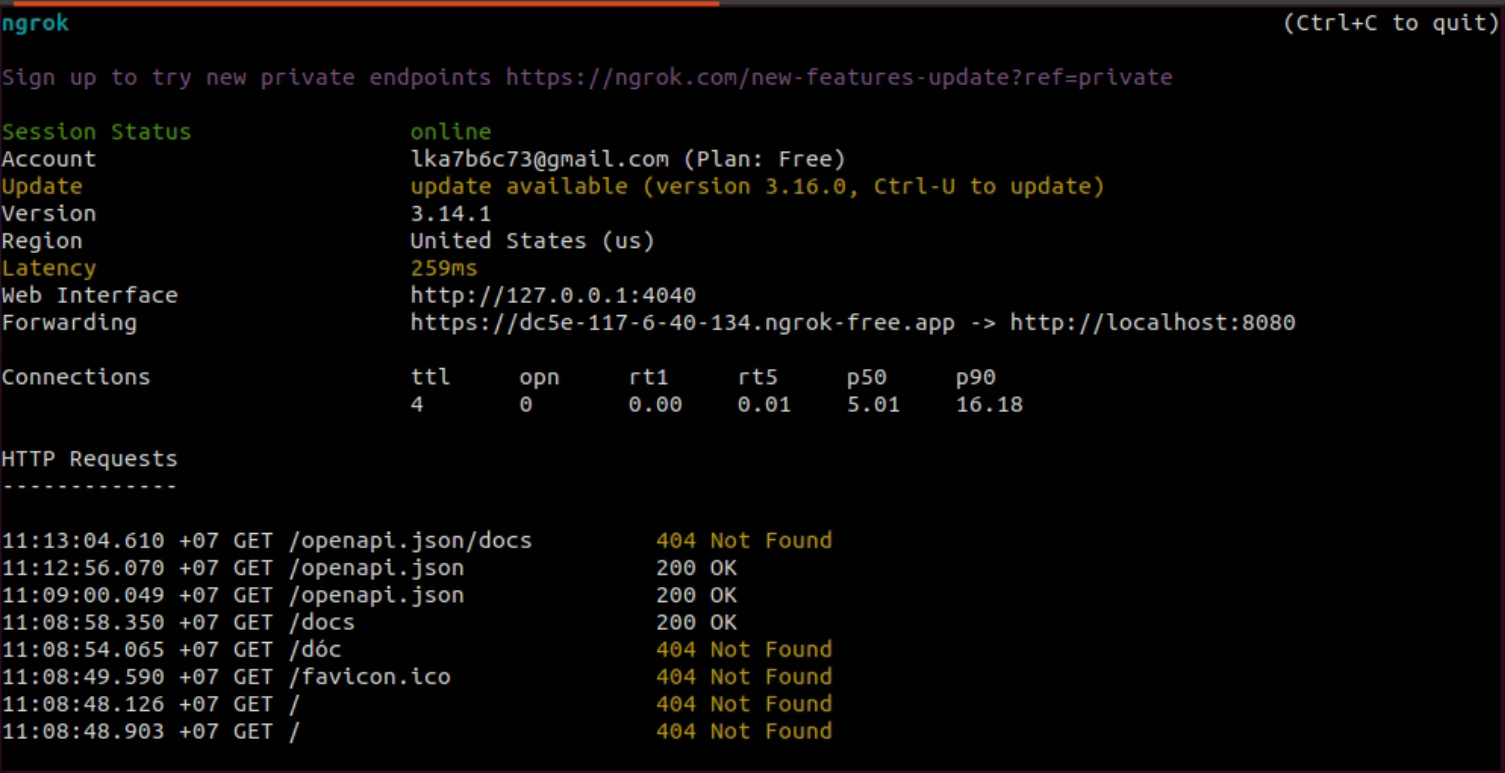
Khi xây dựng **Module API** ta sử dụng **FastAPI** và triển khai nó với **ngrok.**

**FastAPI** là một framework Python mạnh mẽ cho việc phát triển API, với tốc độ cao, dễ sử dụng và có khả năng tự động tạo tài liệu API. Ta sẽ chạy FastAPI ở [*http: //localhost: 8080*](http://localhost:8080)trên máy server

**ngrok** là một công cụ mạng giúp tạo các đường hầm (tunnel) bảo mật từ môi trường nội bộ (localhost) ra môi trường internet. Điều này rất hữu ích khi bạn đang phát triển một API trên máy cá nhân và muốn cho phép người khác hoặc hệ thống khác (ví dụ như webhook) truy cập tạm thời mà không cần triển khai ứng dụng lên một server công khai. Câu lệnh khởi chạy ngrox:

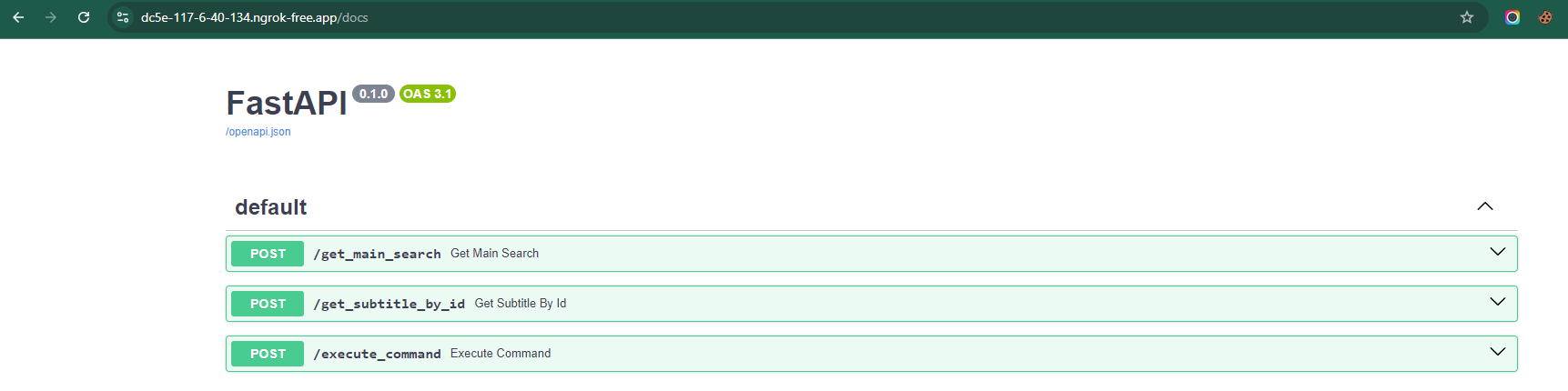
*ngrok http [http: //localhost: 8080](http://localhost:8080)*

Ta được miền [*https: //dc5e-117-6-40-134.ngrok-free.app*](https://dc5e-117-6-40-134.ngrok-free.app)để truy cập vào FastAPI.



Hình 3.16. Giao diện khởi chạy ngrox thành công

**Kết hợp FastAPI và ngrok**: Khi phát triển API bằng FastAPI, thường sẽ chạy ứng dụng trên máy cục bộ bằng lệnh uvicorn. Tuy nhiên, để truy cập API từ các thiết bị hoặc môi trường ngoài mạng cục bộ, ta sử dụng ngrok để tạo một URL công khai cho ứng dụng FastAPI đó. Bằng cách này, bạn có thể thử nghiệm API của mình với các dịch vụ bên ngoài mà không cần phải triển khai lên một máy chủ thực tế. Bằng việc kết hợp cả 2 đã cho ta được kết quả là:



Hình 3.17. Giao diện docs của FastAPI

Ta gọi api 3 hàm chính:

**/get\_main\_search**: đưa ra các thông tin về video có chứa chuỗi từ khóa đưa vào.

**/get\_subtitle\_by\_id**: đưa ra phụ đề của video khi truyền vào id\_video.

**/execute\_commad**: đưa ra các câu lệnh thao tác terminal trên máy server.

3.7. Xây dựng Module xử lý tìm kiếm.

Đây sẽ module xử lý các yêu cầu về api từ máy client và trả lại kết quả thông qua api.

**Quy trình hoạt động:**

****

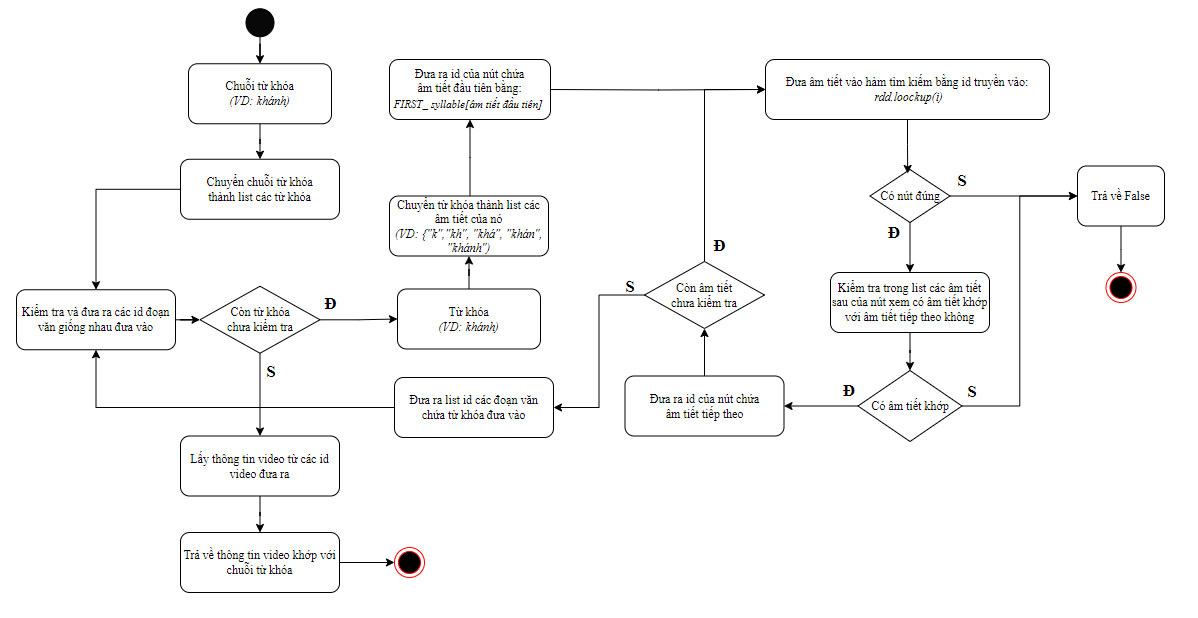
Hình 3.18. Quy trình hoạt động của Module xử lý tìm kiếm

***Mô tả quy trình hoạt động:***

**Module xử lý tìm kiếm** sẽ gồm 2 hàm chính:

***get\_main\_search (Chuỗi từ khóa):***

Khi truyền vào chuỗi từ khóa từ **Module API** hàm sẽ gọi đến **Module lưu trữ** để lấy thông tin của video từ file excel video\_list.xlsx và tiến hành đọc rdd của cây Trie mới nhất được tạo. Quy trình hoạt động cụ thể của hàm:



Hình 3.19. Quy trình xử lý tìm kiếm và đưa ra thông tin video

Kết quả thu được từ hàm sẽ có dạng trả về json chứa các thuộc tính list thông tin video (ID, Video ID, Video titile, Publish Time, Channel) , thời gian chạy, thời điểm chạy code.

Ví dụ:

*{*

*"result":*

*[*

*{*

*"ID": 148,*

*"Video ID": "mV0MmZZ\_29o",*

*"Video Title": "Toàn cảnh thời sự quốc tế sáng 7/10: Hé lộ tác động của việc Mỹ cắt giảm viện trợ cho Ukraine",*

*"Publish Time": "2023-10-06T21: 44: 09Z",*

*"Channel": "Tin24h"*

*},*

*{*

*"ID": 218,*

*"Video ID": "p\_bEwRFDp8Q",*

*"Video Title": "Cuba chỉ trích Mỹ lề mề khi thừa nhận ‘Hội chứng Havana’ | Tin tức 24h mới nhất | TV24h",*

*"Publish Time": "2023-03-05T11: 34: 03Z",*

*"Channel": "Tin24h"*

*}*

*],*

*"time\_load": 0.1307063102722168,*

*"time\_search": "11: 45: 08 20-09-2024"*

*}*

***Get\_subtitles\_by\_id (id\_video):*** được thiết kế để nhận vào ID của video mà người dùng yêu cầu. Khi hàm này được gọi, server sẽ tiến hành liên lạc với **Module lưu trữ** để truy xuất phụ đề tương ứng với ID video đã được truyền vào. Sau khi thu thập hoặc lấy được phụ đề, server sẽ trả về kết quả qua API cho máy client. Dữ liệu này sẽ được định dạng thành TXT để dễ dàng xử lý và hiển thị trên giao diện người dùng, giúp người dùng tiếp cận thông tin một cách nhanh chóng và thuận tiện.

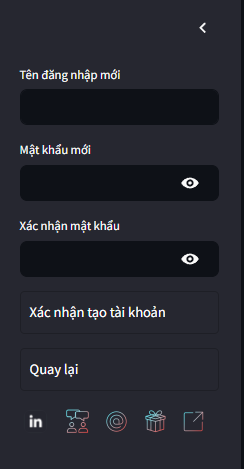
3.8. Xây dựng Module giao diện.

Để minh họa cụ thể kết quả đạt được ta xây dựng **Module giao diện**, chúng ta sẽ xây dựng một ứng dụng web đơn giản bằng **Streamlit** cho phép người dùng nhập câu truy vấn và hiển thị kết quả tìm kiếm.

Đầu tiên để thiết kế 1 ứng dụng web nhiều người dùng thì ta thiết kế trang web đăng nhập để phân chia quyền và cũng như lưu trữ dữ liệu dễ hơn.

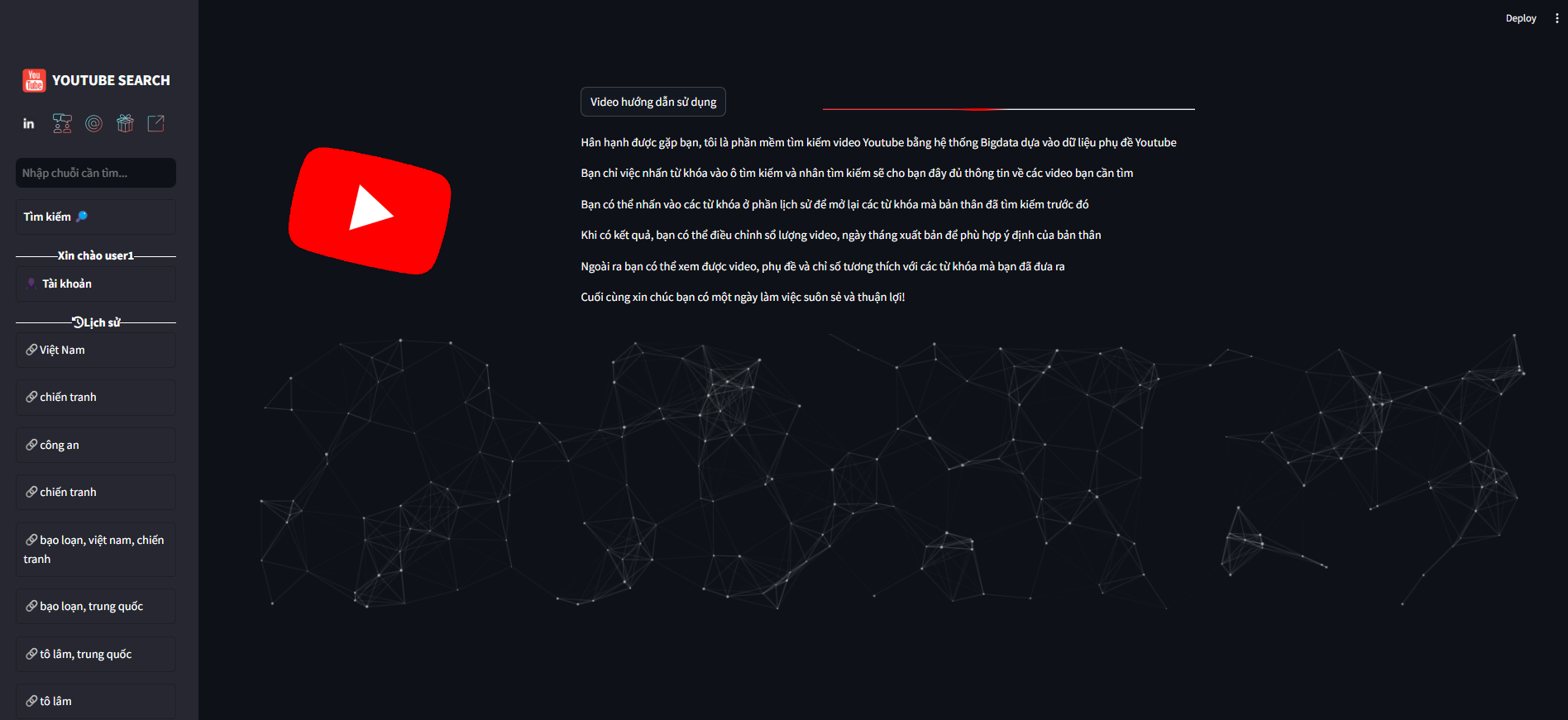


Hình 3.20. Giao diện đăng nhập.



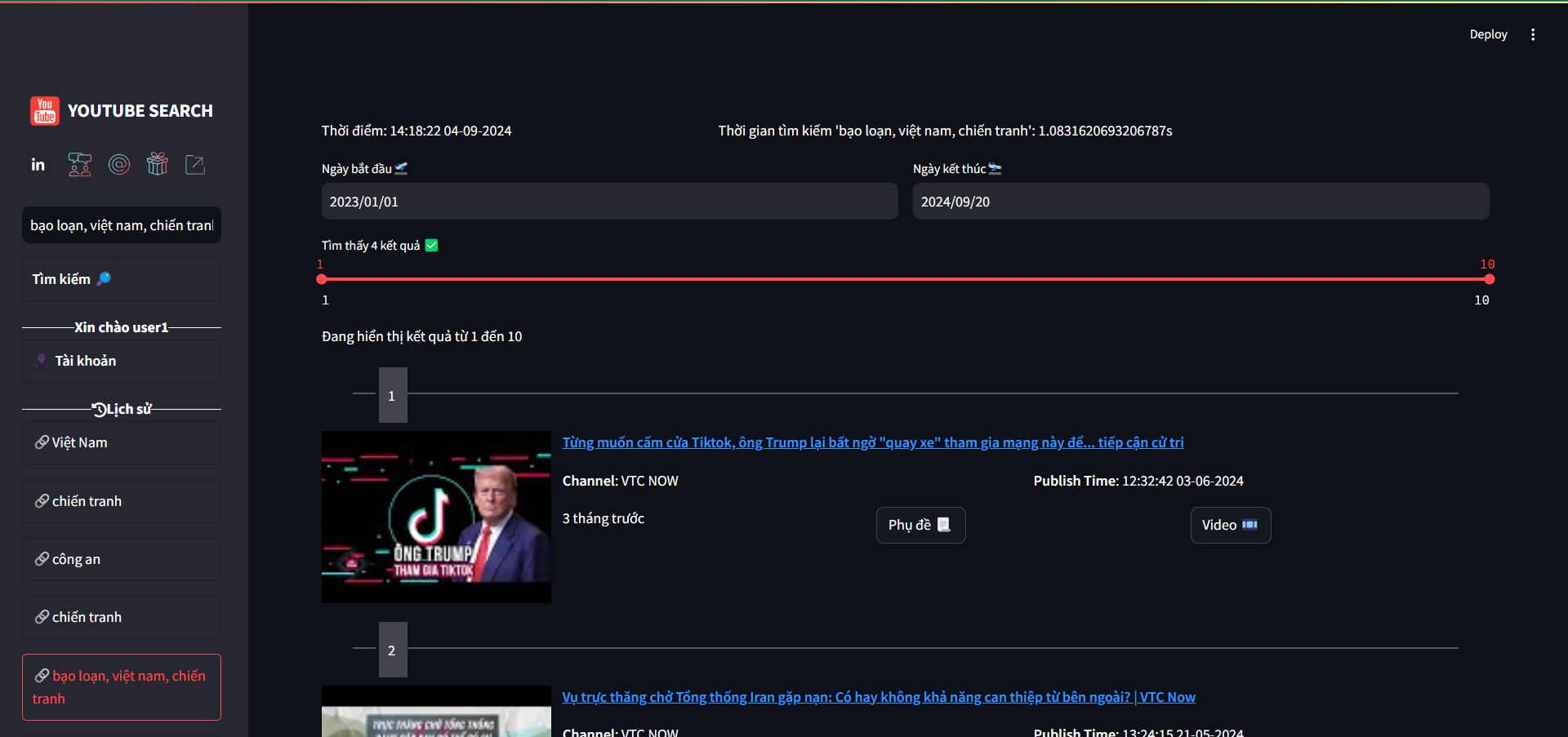
Hình 3.21. Giao diện tạo tài khoản.

Sau khi đăng nhập thành công, giao diện người dù,ng sẽ hiện ra hướng dẫn sử dụng kèm với video hướng dẫn.



Hình 3.22. Giao diện home của ứng dụng.

Ở kết quả tìm kiếm ta có thể điều chỉnh được ngày thời gian tìm kiếm, số lượng cần hiển thị ra và hiên thị ra thời gian chạy tìm kiếm. Kết quả thì phải có đầy đủ các thông tin tiêu đề, ảnh video, ngày xuất bản, kênh xuất bản. Kêt quả sẽ được sắp xếp từ ngày gần nhất đến xa nhất.

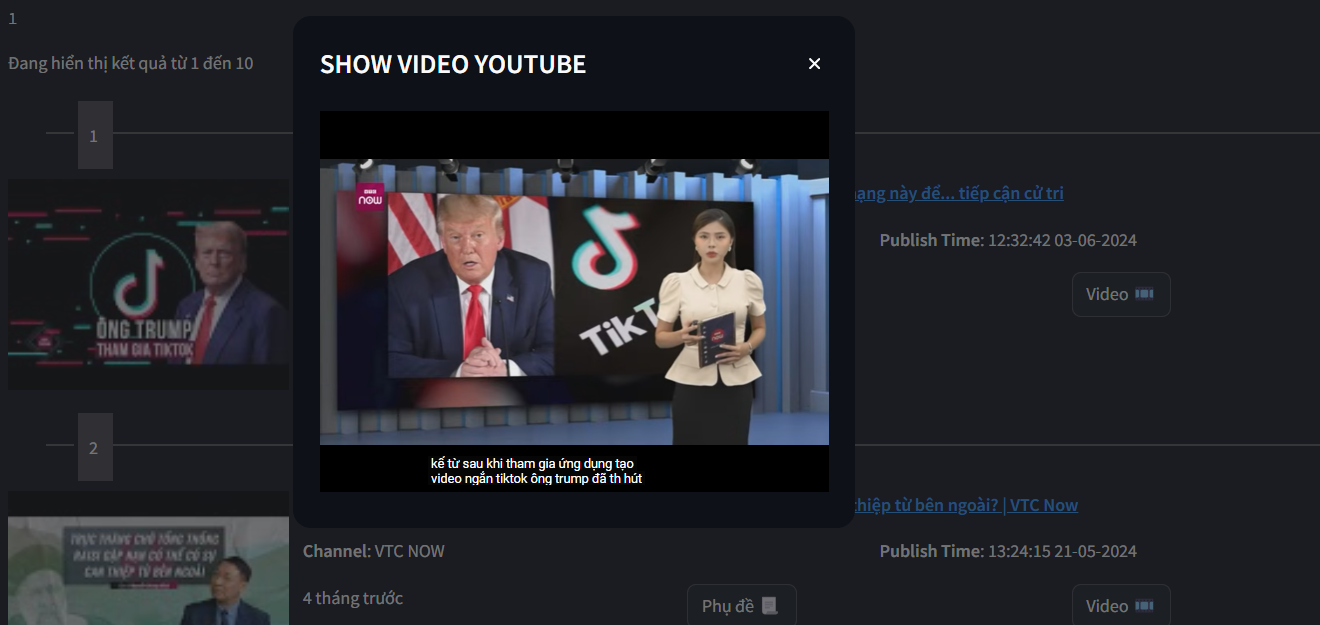


Hình 3.23. Giao diện tìm kiếm thành công.



Hình 3.24. Giao diện hiển thị phụ đề của một video.

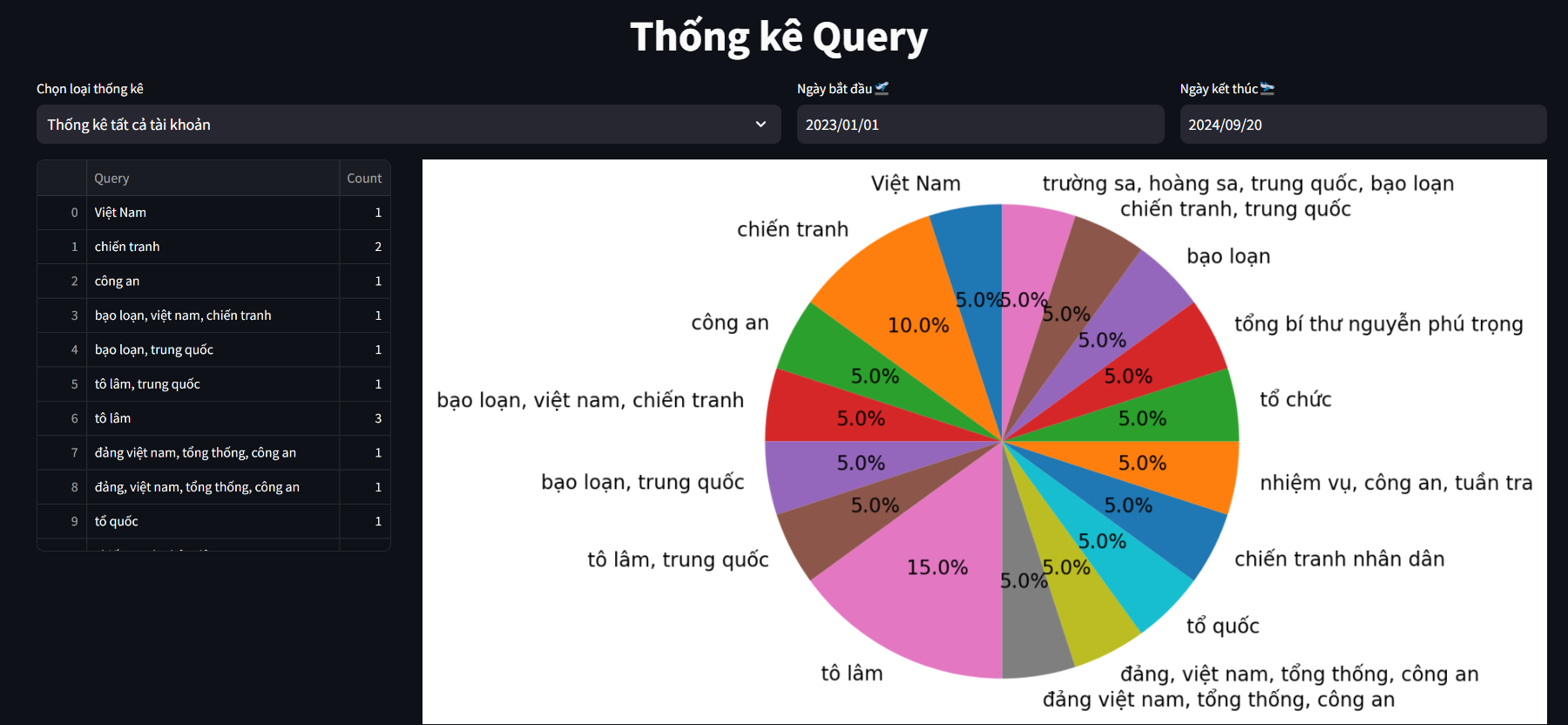
Ta có thế nhân vào button phụ đề để hiển thị ra phụ đề của video cũng như bôi màu các từ khóa đưa vào trong phụ đề và hiển thị số lần xuất hiện. Ta có thể tải về phần phụ đề này để phục vụ mới các mục đích khác. Người dùng cũng có thể nhấn vào tiêu đề của video để chuyển hướng đến Youtube để xem video hoặc đơn giản chỉ cần nhấn vào button video để hiện thị ra 1 box xem video nhỏ tiện ích.



Hình 3.25. Giao diện xem video bằng box của video

Để đảm bảo cho việc xem lại phần lịch sử truy vấn nên ta sẽ tạo ra 1 cột lịch sử ở phần sidebar để có thế click vào chuyển hướng đến nội dung ta đã tìm kiếm trước. Đối với việc lưu giữ này thì ta có thể sử dụng **Mongodb** để lưu trữ lại lịch sử duyệt web.

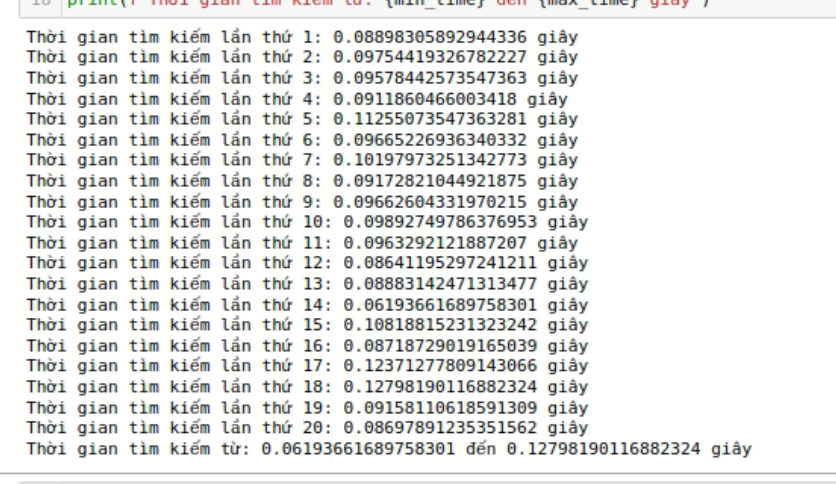
Admin có thể quản lý tài khoản và thống kê các chuỗi từ khóa được truy vấn từ người dùng.



Hình 3.26. Giao diện thống kê truy vấn của Admin.

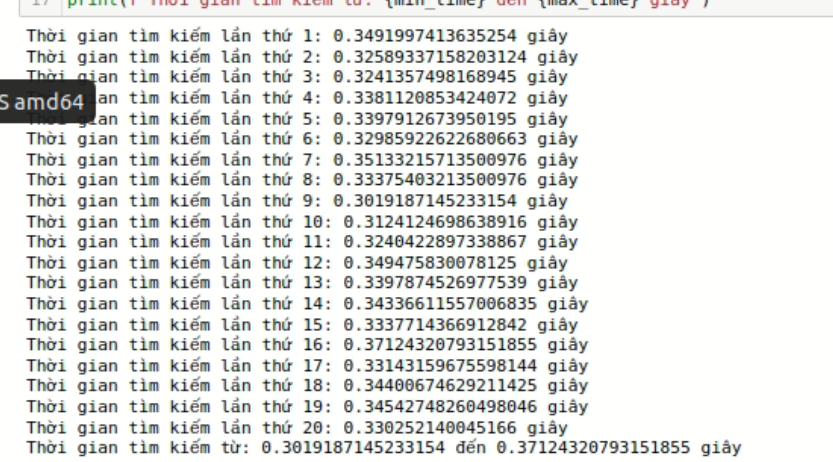
**3.9. Thực nghiệm với các bộ dữ liệu khác nhau**

Thực nghiệm kiểm tra tìm kiếm 1 chuỗi từ khóa gồm 4-5 từ khóa với tập dữ liệu gồm **5, 874** videos được thu thập từ Youtube.



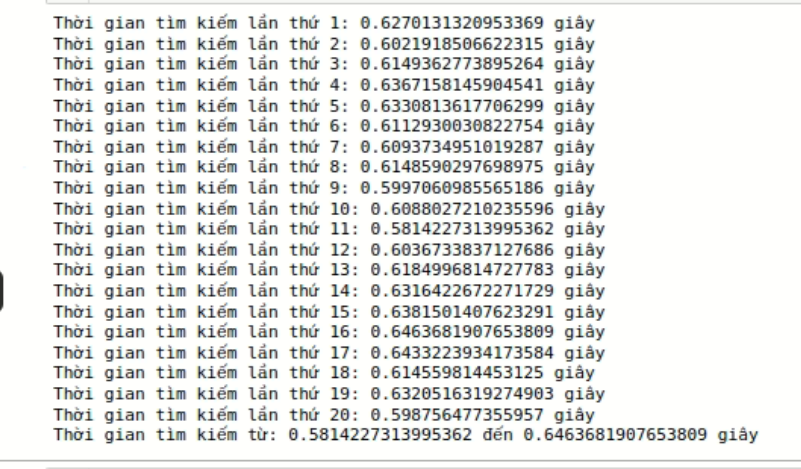
Hình 3.27. Kết quả tìm kiếm 20 lần với bộ dữ liệu 5,874 videos.

Thực nghiệm kiểm tra tìm kiếm 1 chuỗi từ khóa gồm 4-5 từ khóa với tập dữ liệu gồm **59,639** videos được thu thập từ Youtube.



Hình 3.28. Kết quả tìm kiếm 20 lần với bộ dữ liệu 59,639 videos.

Thực nghiệm kiểm tra tìm kiếm 1 chuỗi từ khóa gồm 4-5 từ khóa với tập dữ liệu gồm **105,221** videos được thu thập từ Youtube.



Hình 3.29. Kết quả tìm kiếm 20 lần với bộ dữ liệu 105,221 videos.

**3.10. Kết luận Chương 3**

Kết quả cho thấy, khả năng nhanh chóng và hiệu quả trong việc tìm kiếm dữ liệu phụ đề trên Youtube. Chương 3 đã tập trung vào quá trình phát triển và triển khai các thành phần chính của hệ thống, đảm bảo hệ thống hoạt động hiệu quả, ổn định và đáp ứng được các yêu cầu đặt ra. Từ việc xác định yêu cầu cho đến xây dựng và thử nghiệm, mỗi bước đều được thực hiện một cách cẩn trọng nhằm đảm bảo tính toàn diện và khả năng mở rộng của hệ thống. Quá trình thử nghiệm trên các bộ dữ liệu khác nhau cũng đã chứng minh được tính hiệu quả và linh hoạt của hệ thống, sẵn sàng để áp dụng vào thực tế.

**KẾT LUẬN**

**1. Các kết quả đã thu được.**

Trong nghiên cứu này, tôi đã đề xuất một giải pháp hiệu quả để tăng tốc quá trình tìm kiếm dữ liệu phụ đề tiếng Việt bằng cách sử dụng các kỹ thuật xử lý dữ liệu lớn. Tôi đã tích hợp nền tảng Hadoop và Apache Spark vào quy trình nghiên cứu, đồng thời kết hợp với các kỹ thuật xử lý văn bản. Thành công trong việc xây dựng ứng dụng lưu trữ và tìm kiếm dữ liệu phụ đề video trên mạng xã hội Youtube.

Việc sử dụng kỹ thuật lưu trữ dữ liệu lớn như HDFS mang lại nhiều lợi ích so với các kỹ thuật lưu trữ khác như: Khả năng mở rộng dễ dàng bằng cách thêm các nút vào cụm, phù hợp với việc lưu trữ và xử lý lượng lớn dữ liệu, đồng thời đảm bảo hiệu suất cao khi dung lượng dữ liệu tăng lên. Hiệu suất tốt đối với việc đọc và ghi dữ liệu lớn do được phân phối trên nhiều nút trong cụm. Độ tin cậy cao do dữ liệu được sao lưu trên nhiều nút khác nhau trong cụm. Chi phí cho việc triển khai và duy trì ít tốn kém phù hợp cho nhiều đơn vị trong toàn quân.

Thực nghiệm trên dữ liệu phụ đề Youtube Tiếng Việt đã chỉ ra rằng sử dụng nền tảng Apache Spark bằng cách xây dựng cây Trie đã giảm đáng kể thời gian tìm kiếm, mang lại kết quả nhanh chóng và chính xác hơn. Điều này đã cung cấp một cơ sở vững chắc cho ứng dụng thực tiễn của giải pháp trong việc tìm kiếm và truy xuất dữ liệu phụ đề tiếng Việt trên các nền tảng trực tuyến.

**2. Hạn chế của đồ án.**

Trong phạm vi nghiên cứu này, khối lượng dữ liệu phụ đề cũng như thời gian thu thập dữ liệu vẫn còn hạn chế, cho nên chưa đánh giá toàn diện về hiệu suất đối với dữ liệu lớn. Vì vậy, đề xuất xây dựng hệ thống theo hướng thời gian thực, tăng khối lượng dữ liệu theo thời thu thập dữ trực tuyến, mang tính thời sự, đảm bảo kịp thời nắm bắt tình hình thực tế.

3. Hướng phát triển của đồ án.

Về mặt lý thuyết, tôi sẽ tìm hiểu kỹ hơn về các kỹ thuật mới, kết hợp các phương pháp khác nhau kèm theo nâng cấp cơ sở hạ tầng cụm máy tính để có thể đánh giá rõ hơn.

Hiện nay có rất nhiều công cụ xử lý ngôn ngữ tự nhiên ví dụ như TF-IDF. Bằng việc kết hợp các công cụ mới vào để giảm khối lượng dữ liệu cần lưu trữ, tập trung lưu trữ những nội dung quan trọng, và có thể tìm kiếm dựa vào ngữ nghĩa của câu từ. Tôi sẽ tìm hiểu thêm công cụ speed-to-text và xây dựng được một mô hình đảm bảo chuyển dữ liệu âm thanh thành văn bản chính xác hơn để tránh lệ thuộc vào phụ đề cung cấp sẵn của Youtube.

Trong tương lai, tôi sẽ tập trung cải thiện các xây dựng cây Trie và các phương pháp khác phục vụ cho quá trình nghiên cứu cũng như áp dụng được vào thực tiễn sau này.

**DANH MỤC TÀI LIỆU THAM KHẢO**

1. Amalina, F., Hashem, I. A. T., Azizul, Z. H., Fong, A. T., Firdaus, A., Imran, M., & Anuar, N. B., Blending Big Data Analytics: Review on Challenges and a Recent Study, IEEE Access, 2019.
2. Aslam, F., The Benefits and Challenges of Customization within SaaS Cloud Solutions, American Journal of Data, Information and Knowledge Management, 4 (1) , 14-22, 2023.
3. Aslam, F., Role of Cloud Computing for Big Data, International Conference on Cloud Computing and Big Data, 2023.
4. Bahrami, M., & Singhal, M., The role of cloud computing architecture in big data, Springer, Retrieved August 11, 2023.
5. Chen, Y., Li, C., & Wang, H, Big Data and Predictive Analytics for Business Intelligence: A Bibliographic Study (2000–2021) , 2021.
6. Cukier, K., Data everywhere: a special report on managing information, Economist Newspaper, 2010.
7. Cutting, D., Apache Lucene - Apache Lucene core, 2019.
8. Das, M., & Dash, R., Role of Cloud Computing for Big Data: A Review, Springer, Retrieved August 11, 2023.
9. Drowning in numbers - digital data will flood the planet - and help us understand it better, Economist, 2011.
10. Fact sheet: Big Data across the federal government, White House, 2012.
11. Gambhir, M., & Gupta, V., Recent Automatic Text Summarization Techniques: A Survey, Artificial Intelligence Review, 47 (1), 1-66, 2017.
12. Gantz, J., & Reinsel, D., Extracting value from chaos, IDC iView, pp. 1-12, 2011.
13. Grainger, T., & Potter, T., Solr in Action, 1st Edition, Manning Publications Co., Greenwich, CT, USA, 2014.
14. Gupta, S., & Narsimha, G., Development of fuzzy search method for creating an efficient information search system in text data, Technology Audit and Production Reserves, 1 (2 (75) ) , 20-24, 2024.
15. Hashem, I., Yaqoob, I., Anuar, N., Mokhtar, S., & Gani, A., The rise of “big data” on cloud computing: Review and open research issues, Retrieved August 11, 2023.
16. Iqbal, W., Malik, W., Bukhari, F., & Nawaz, Z., Big Data Full-Text Search Index Minimization Using Text Summarization, Information Technology and Control, 50 (2) , 375-389, 2021.
17. Kononenko, O., Baysal, O., Holmes, R., & Godfrey, M. W., Mining Modern Repositories with Elasticsearch, Proceedings of the 11th Working Conference on Mining Software Repositories, MSR 2014, ACM, New York, NY, USA, 328-331, 2014.
18. Kleshch, K., Development of fuzzy search method for creating an efficient information search system in text data, Technology Audit and Production Reserves, 1 (2 (75) ) , 20-24, 2024.
19. Labrinidis, A., & Jagadish, H. V., Challenges and Opportunities with Big Data, Proceedings of the VLDB Endowment, 5 (12) , 2032-2033, 2012.
20. Lohr, S., The age of Big Data, New York Times, p. 11, 2012.
21. Manyika, J., McKinsey Global Institute, Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H., Big Data: the next frontier for innovation, competition, and productivity, McKinsey Global Institute, 2011.
22. Munezero, M., Montero, C. S., Sutinen, E., & Pajunen, J., Are they different? Affect, feeling, emotion, sentiment, and opinion detection in text, IEEE Trans. Affect. Comput., 5 (2) , 101–111, 2014.
23. O'Driscoll, A., Daugelaite, J., & Sleator, R., Methodological Review: ‘Big data’, Hadoop and cloud computing in genomics, Retrieved August 11, 2023.
24. Pang, B., Lee, L., & Vaithyanathan, S., Opinion Mining and Sentiment Analysis, IEEE Trans. Knowl. Data Eng., 20 (6) , 866–879, 2008.
25. Pang, B., & Lee, L., Sentiment Analysis and Opinion Mining: A Survey, IEEE Trans. Knowl. Data Eng., 28 (9) , 1299–1323, 2016.
26. Prasad, A., & Patel, D., Lucene Search Engine: An Overview, DRTCHP International, 10, 2005.
27. Purcell, B., Big data using cloud computing, Retrieved August 11, 2023, from citeseerx.ist.psu.edu.
28. Song, X., Liang, X., & Ma, Y., A Sentiment Analysis Approach to Predict Stock Market Trends, 2018 IEEE International Conference on Big Data (Big Data) , Seattle, WA, USA, 3596–3602, 2018.
29. Special online collection: dealing with big data, Science Magazine, 2011.
30. Tang, B., Chen, Z., Hefferman, G., Wei, T., & He, H., A hierarchical distributed fog computing architecture for big data analysis in smart cities, Retrieved August 11, 2023.