ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

VŨ MINH NHẬT

KHÓA LUẬN TỐT NGHIỆP

KHUYẾN NGHỊ TIN TỨC PHÙ HỢP SỞ THÍCH NGƯỜI ĐỌC

Personalized News Recommendation

KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM

TP. HỒ CHÍ MINH, 2018

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA CÔNG NGHỆ PHẦN MỀM

VŨ MINH NHẬT – 13520580

KHÓA LUẬN TỐT NGHIỆP

KHUYẾN NGHỊ TIN TỨC PHÙ HỢP SỞ THÍCH NGƯỜI ĐỌC

Personalized News Recommendation

KỸ SƯ NGÀNH KỸ THUẬT PHẦN MỀM

GIẢNG VIÊN HƯỚNG DẪN

TS. HUỲNH NGỌC TÍN

TP. HỒ CHÍ MINH, 2018

DANH SÁCH HỘI ĐỒNG BẢO VỆ KHÓA LUẬN

Hội đồng chấm khóa luận tốt nghiệp, thành lập theo Quyết định số …… ngày …….. của Hiệu trưởng Trường Đại học Công nghệ Thông tin.

1. TS. Nguyễn Tấn Trần Minh Khang……. – Chủ tịch.
2. ThS. Huỳnh Tuấn Anh……. – Thư ký.
3. ThS. Phan Nguyệt Minh……. – Ủy viên.

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc Lập - Tự Do - Hạnh Phúc

TP. HCM, ngày…5..tháng…2..năm 2018…..

NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP

CỦA CÁN BỘ HƯỚNG DẪN

Tên khóa luận:

Khuyến nghị tin tức phù hợp sở thích người đọc

Nhóm SV thực hiện:		**Cán bộ hướng dẫn:**
Vũ Minh Nhật	MSSV:13520580	TS. Huỳnh Ngọc Tín

Đánh giá Khóa luận

1. Về cuốn báo cáo:

Số trang	83	Số chương	6
Số bảng số liệu	16	Số hình vẽ	17
Số tài liệu tham khảo	11	Sản phẩm	01

Một số nhận xét về hình thức cuốn báo cáo:
Bố cục báo cáo bao gồm các chương:
Chương 1: Mở đầu
Chương 2: Tổng quan về hệ khuyến nghị
Chương 3: Bài toán khuyến nghị tin tức phù hợp sở thích người đọc
Chương 4: Hiện thực hệ thống

2. Về nội dung nghiên cứu:

Đề tài tìm hiểu các kiến thức cơ bản về hệ khuyến nghị, các phương pháp khuyến nghị cơ bản, thuật toán gom cụm Local Sensitive Hashing (LSH), xác suất Bayesian, dữ liệu lớn, tính toán phân tán, song song.

Đề tài rèn luyện một số kỹ năng như đọc hiểu các bài báo khoa học, triển khai cụm máy tính bằng Apache Hadoop, hệ quản trị cơ sở dữ liệu phân tán với Apache Cassandra,…

3. Về chương trình ứng dụng:

Hệ thống khuyến nghị tin tức phù hợp sở thích người đọc đang được triển khai

tại trang CafeBiz

4. Về thái độ làm việc của sinh viên:

Sinh viên thể hiện thái độ nghiêm túc khi hẹn và làm việc với giáo viên hướng dẫn

Đánh giá chung: Khóa luận đạt yêu cầu của một khóa luận tốt nghiệp kỹ sư, xếp loại Xuất sắc

Điểm từng sinh viên:

Vũ Minh Nhật **:……/10**

Người nhận xét

Huỳnh Ngọc Tín

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc Lập - Tự Do - Hạnh Phúc

TP. HCM, ngày…5..tháng…2..năm 2018…..

NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP

CỦA CÁN BỘ PHẢN BIỆN

Tên khóa luận:

Khuyến nghị tin tức phù hợp sở thích người đọc

Nhóm SV thực hiện:		**Cán bộ phản biện:**
Vũ Minh Nhật	MSSV:13520580	Phan Nguyệt Minh

Đánh giá Khóa luận

1. Về cuốn báo cáo:

Số trang	83	Số chương	6
Số bảng số liệu	16	Số hình vẽ	17
Số tài liệu tham khảo	11	Sản phẩm	01

Một số nhận xét về hình thức cuốn báo cáo:
Bố cục báo cáo bao gồm các chương:
Chương 1: Mở đầu
Chương 2: Tổng quan về hệ khuyến nghị
Chương 3: Bài toán khuyến nghị tin tức phù hợp sở thích người đọc
Chương 4: Hiện thực hệ thống

2. Về nội dung nghiên cứu:

Đề tài đã tìm hiểu các kiến thức cơ bản về hệ khuyến nghị, tính toán phân tán, song song, thuật toán gom cụm Local Sensitive Hashing (LSH), xác suất Bayesian, … đồng thời rèn luyện một số kỹ năng như triển khai cụm máy tính bằng Apache Hadoop, hệ quản trị cơ sở dữ liệu phân tán với Apache Cassandra,…

3. Về chương trình ứng dụng:

Hệ thống khuyến nghị tin tức phù hợp sở thích người đọc đã đi vào hoạt

động, đang được triển khai tại trang CafeBiz

4. Về thái độ làm việc của sinh viên:

Sinh viên thể hiện thái độ nghiêm túc khi hẹn và làm việc với giáo viên phản biện.

Đánh giá chung: Khóa luận đạt yêu cầu của một khóa luận tốt nghiệp kỹ sư, xếp loại Giỏi

Điểm từng sinh viên:

Vũ Minh Nhật **:……../10**

Người nhận xét

Phan Nguyệt Minh

LỜI CẢM ƠN

Trước tiên, em xin gửi lời cảm ơn chân thành đến TS. Huỳnh Ngọc Tín, người đã tận tình giúp đỡ, trực tiếp chỉ bảo, hướng dẫn em trong suốt quá trình thực hiện khóa luận tốt nghiệp này. Những kinh nghiệm, lời nhận xét và chia sẻ của thầy truyền đạt cho em thật sự rất quý báu và ý nghĩa.

Em cũng gửi lời cảm ơn đến quý thầy cô đang công tác tại Trường Đại học Công Nghệ Thông Tin nói chung và Khoa Công nghệ phần mềm nói riêng đã dạy dỗ, truyền đạt những kiến thức, kinh nghiệm vô cùng quý báu, giúp em vận dụng trong quá trình thực tập.

Ngoài ra, xin cảm ơn sự hỗ trợ quý giá của các bạn trong nhóm AdTech tại công ty VCCorp trong quá trình xây dựng hệ thống, đặc biệt là anh Phan Văn Tân.

Cuối cùng, em xin chúc thầy luôn mạnh khỏe và đạt nhiều thành công trong cuộc sống.

Tp. HCM, tháng 1 năm 2018

Sinh viên thực hiện

Vũ Minh Nhật

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc Lập - Tự Do - Hạnh Phúc

ĐỀ CƯƠNG CHI TIẾT

TÊN ĐỀ TÀI: KHUYẾN NGHỊ TIN TỨC PHÙ HỢP SỞ THÍCH NGƯỜI ĐỌC

Cán bộ hướng dẫn: TS. Huỳnh Ngọc Tín

Thời gian thực hiện:Từ ngày 04/09/2017 đến ngày 04/01/2018

Sinh viên thực hiện:

Vũ Minh Nhật - 13520580

Nội dung đề tài:

1. Mục tiêu đề tài:
    - Tìm hiểu và chọn ra phương pháp phù hợp đối với bài toán khuyến nghị tin tức phù hợp sở thích người đọc.
    - Xây dựng hệ thống khuyến nghị tin tức dựa trên phương pháp đã chọn.
2. Phạm vi đề tài:
    - Nguồn dữ liệu: cơ sở dữ liệu báo chí và hệ thống ghi log của công ty.
    - Ngôn ngữ: Tiếng Việt.
    - Phương pháp xây dựng hồ sơ người dùng được đề xuất trong bài báo *"Personalized News Recommendation Based on Click Behavior"* và thuật toán LSH với phương pháp MinHash trong bài báo *"Google news personalization: scalable online collaborative filtering"*
3. Phương pháp thực hiện:
    - Tìm hiểu về cách thức xây dựng hệ thống Google News.
    - Tìm hiểu công nghệ Apache Hadoop, Apache Spark, Apache Cassandra.
    - Thu thập và lọc nhiễu dữ liệu.
    - Phân tích lịch sử người dùng.
    - Cài đặt thuật toán xây dựng hồ sơ người dùng và thuật toán LSH với phương pháp MinHash.
    - Hiện thực hệ thống
4. Kết quả mong đợi:
    - Hệ thống khuyến nghị tin tức phù hợp sở thích người đọc.
    - Báo cáo đề tài

Kế hoạch thực hiện:

- Viết thuyết minh đề tài (04/09/2017 - 05/09/2017)
- Tìm hiểu framework Apache Hadoop, Apache Spark (06/09/2017 - 04/10/2017)
- Cài đặt cụm máy Hadoop, phục vụ cho việc lưu trữ, tính toán phân tán (05/10/2017 - 06/10/2017)
- Tìm hiểu, làm rõ bài báo "*Personalized News Recommendation Based on Click Behavior"* (07/10/2017 - 21/10/2017)

- Hiện thực thuật toán xây dựng hồ sơ người dùng (21/10/2017 - 4/11/2017)
- Tìm hiểu, làm rõ bài báo *"Google news personalization: scalable online collaborative filtering"*. (05/11/2017 - 19/11/2017)
- Hiện thực hiện thuật toán LSH (20/11/2017 - 04/12/2017)
- Kết hợp các thành phần của hệ thống lại với nhau (05/12/2017 - 08/12/2018)
- Tìm hiểu Apache Cassandra để tối ưu hóa hệ thống. (09/12/2017 - 23/12/2017)
- Xây dựng API (24/12/2017 - 31/12/2017)
- Triển khai hệ thống (01/01/2017 - 04/01/2017)
- Viết báo cáo (24/12/2017 - 04/01/2018)

Xác nhận của CBHD (Ký tên và ghi rõ họ tên)	**TP. HCM, ngày….tháng …..năm…..** **Sinh viên** (Ký tên và ghi rõ họ tên)

Mục lục

Danh mục các ký hiệu, thuật ngữ và chữ viết tắt

Nearest Neighbor Search	: tìm láng giềng gần nhất
Locality Sensitive Hashing	: thuật toán Locality Sensitive Hashing
Item	: đối tượng mà chúng ta muốn gom cụm.
Hash function	: hàm ánh xạ item thành một giá trị đại diện cho item đó.
Hash Code	: giá trị đại diện cho item sau khi đã qua Hash function.
Bucket	: tâp hợp các item cùng hash code.
Candidate pair	: cặp item rơi vào chung một bucket.
Band	: tập hợp một nhóm các hash function.
Characteristic Matrix	: ma trận đặc trưng mô tả sự tương quan giữa các item theo đặc trưng cụ thể để tiến hành gom cụm.
Signature Matrix	: ma trận có tính chất tương đương với Characteristic Matrix

Danh sách bảng

Danh sách hình vẽ

TÓM TẮT KHÓA LUẬN

Việc đọc tin tức trực tuyến đã trở nên rất phổ biến khi mà ta có thể truy cập tin tức từ hàng triệu nguồn khác nhau trên thế giới. Riêng tại kho dữ liệu tin tức của công ty VCCorp, cứ mỗi giờ, trung bình có 2118 bài viết mới [1] được đăng trên các trang tin tức của Việt Nam. Điều này dẫn đến việc quá tải thông tin, cũng như đặt ra câu hỏi cho các hệ thống tin tức trực tuyến: "Làm sao để cung cấp tin tức phù hợp với sở thích người đọc? Làm sao để giữ người dùng trên website lâu hơn?". Và hệ khuyến nghị là một giải pháp đáng được quan tâm nhằm giải quyết các câu hỏi trên.

Khóa luận hướng tới việc hiện thực hóa lại hệ thống *Google News* của Google dựa trên hai bài báo khoa học được công bố *"Google news personalization: scalable online collaborative filtering"* [1] năm 2007 và *"Personalized news recommendation based on click behavior"* [2] năm 2010. Sau khi hoàn thành, hệ thống sẽ được đánh giá nghiêm túc trước khi triển khai thực trên các trang tin tức trực tuyến mà công ty Vccorp đang quản lý.

Sau quá trình thực hiện đề tài khóa luận, em thu được các kết quả sau:

- Kiến thức, khả năng làm việc với các framework hỗ trợ việc tính toán, xử lý phân tán như Apache Hadoop, Apache Spark.
- Kiến thức về hệ thống khuyến nghị.
- Hệ thống khuyến nghị tin tức phù hợp sở thích người đọc

[1] Theo thống kê từ dữ liệu thu thập bởi công ty VCCorp tính đến tháng 7/2017

Chương 1

MỞ ĐẦU

1.1 Dẫn nhập

Việc đọc tin tức trực tuyến đã trở nên rất phổ biến khi mà ta có thể truy cập tin tức từ hàng triệu nguồn tin khác nhau trên toàn thế giới. Thách thức chính của các trang tin tức điện tử là giúp người dùng tìm đúng bài báo mà họ thích đọc, họ muốn đọc hay họ cần đọc tại thời điểm hiện tại. Các công cụ tìm kiếm có thể giúp giải quyết một khía cạnh của vấn đề nếu như thông tin bạn cần đã được định hình thành các từ khóa (keywords). Tuy nhiên, trong nhiều trường hợp, người dùng thậm chí không biết mình sẽ cần tìm thông tin gì. Họ mở các trang báo, mở hết tab này đến tab khác, kéo lên kéo xuống với thái độ cần tìm một cái gì đó thú vị. Hệ khuyến nghị ra đời với tham vọng giải quyết được nhu cầu này của của người dùng.

Về cơ bản, hệ thống khuyến nghị sẽ "đem" những thông tin mà người dùng quan tâm, có ích đến với người dùng, thay vì phải đi tìm như trước. Trong ngữ cảnh hệ khuyến nghị, nó sẽ đem những thông tin cần thiết hoặc phù hợp sở thích đến cho người dùng, người dùng không còn phải đi tìm thông tin như trước.

Để khuyến nghị một đối tượng cho người dùng, ta có nhiều cách thức khác nhau. Khuyến nghị những đối tượng người dùng hay xem, khuyến nghị những đối tượng liên quan đến đối tượng họ đã xem hoặc đang xem hay khuyến nghị đối tượng phù hợp với sở thích của người dùng, tức khuyến nghị theo hướng cá nhân hóa. Khuyến nghị theo hướng cá nhân hóa ngày càng quan trọng khi lượng dữ liệu được tạo ra ngày càng lớn, khi mà các phương pháp khuyến nghị đối tượng liên quan đến đối tượng đang xem hay đã xem không còn phù hợp với nhu cầu hiện tại.

Công ty VCCorp quản lý [1] cũng như hợp tác [2] với một lượng lớn các tờ báo điện tử, có thể kể đến CafeF, CafeBiz, Kênh 14, GenK, Afamily, Dân Trí, Thanh niên, Pháp luật, Người lao động,... Do đó, dữ liệu công ty cần phải lưu trữ và xử lý là rất lớn. Cụ thể, tính đến ngày 29.12.2017, cơ sở dữ liệu tin tức của công ty là 564.3GB. Dữ liệu về lịch sử truy cập các trang web tin tức trung bình một ngày là 23GB, một năm khoảng 8.4TB. Trên từng trang tin tức mà công ty đang sở hữu, sẽ có một số khung (box) hiển thị tin tức với nội dung được tạo ra tự động từ các thuật toán. Mỗi khung sẽ có một nhiệm vụ, chức năng khác nhau như: khung chuyên cung cấp các tin mới nhất, khung chuyên cung cấp các tin liên quan đến tin đang đọc, ... Hiện nay, công ty muốn triển khai thử nghiệm khung chuyên cung cấp tin tức phù hợp sở thích người đọc (Box for you) lên trang tin tức CafeBiz. Nếu sau khi triển khai, lượng người dùng tăng, thời gian người dùng ở lại trên trang CafeBiz lâu hơn, traffic tăng,... thì công ty sẽ triển khai lên toàn bộ các trang còn lại. Bài toán Khuyến nghị tin tức phù hợp sở thích người đọc là cốt lõi cho hệ thống trên.

Có nhiều khó khăn khi giải quyết bài toán này. Một số khó khăn cơ bản:

- Nhập nhằng người dùng: Việc định danh người dùng gặp nhiều khó khăn. Làm thế nào để xác định là cùng một người nếu họ sử dụng các thiết bị khác nhau (máy tính, điện thoại, tablet) ?. Đây thật sự là câu hỏi không dễ để giải quyết vì không giống như các trang mạng xã hội, người dùng đọc tin tức không cần phải đăng nhập để đọc tin. Nếu việc nhập nhằng này không được xử lý tốt, dữ liệu bị nhiễu nhiều sẽ ảnh hưởng lớn đến thuật toán khuyến nghị bởi vì ta không xây dựng được bộ hồ sơ người dùng đủ tốt.

- Dữ liệu lớn

- Sở thích người dùng thay đổi theo thời gian: vì sở thích thay đổi theo thời gian, ta buộc phải cập nhật lại hồ sơ người dùng thường xuyên. Chi phí cho việc xây dựng hồ sơ người dùng là lớn, do đó, ta phải nghiên cứu phương pháp tối ưu cho việc xây dựng hồ sơ người dùng.

Trong ngữ cảnh của khóa luận, em hướng tới việc áp dụng các nghiên cứu mới của

[1] https://vccorp.vn/san-pham-dich-vu.htm
[2] https://vccorp.vn/doi-tac.htm

Google về vấn đề khuyến nghị tin tức theo hướng cá nhân hóa. Các nghiên cứu này hướng đến giải quyết vấn đề về dữ liệu lớn và sở thích người dùng thay đổi theo thời gian. Với sự giúp đỡ, tạo điều kiện thuận lợi từ công ty VCCorp, khóa luận sẽ xây dựng một hệ thống khuyến nghị tin tức phù hợp sở thích người đọc cho các trang báo điện tử mà công ty đang quản lý.

1.2 Mục tiêu đề tài

- Tìm hiểu và chọn ra phương pháp phù hợp đối với bài toán khuyến nghị tin tức phù hợp sở thích người đọc.
- Giải quyết vấn đề dữ liệu lớn và sở thích người dùng thay đổi theo thời gian.
- Xây dựng hệ thống khuyến nghị tin tức dựa trên phương pháp đã chọn.

1.3 Nội dung thực hiện

- Tìm hiểu về cách thức xây dựng hệ thống Google News.
- Seminar, làm rõ các phần sẽ thực hiện.
- Thu thập và lọc nhiễu dữ liệu.
- Phân tích log.
- Cài đặt thuật toán xây dựng hồ sơ người dùng và LSH với phương pháp MinHash.
- Xây dựng hệ thống
  - Tìm hiểu về Apache Hadoop và Apache Spark. Thiết lập Hadoop cluster.
  - Xây dựng kiến trúc hệ thống
  - Hiện thực hóa các module
  - Triển khai hệ thống

1.4 Phạm vi đề tài

- Nguồn dữ liệu: cơ sở dữ liệu báo chí và hệ thống ghi log của công ty.
- Ngôn ngữ: Tiếng Việt.
- Phương pháp xây dựng hồ sơ người dùng được đề xuất trong bài báo *"Personalized News Recommendation Based on Click Behavior"* và thuật toán LSH với phương pháp MinHash trong bài báo *"Google news personalization: scalable online collaborative filtering"*.

1.5 Cấu trúc báo cáo

Luận văn được bố cục thành chương mục như sau:

- **Chương 1**: Giới thiệu về đề tài.
- **Chương 2**: Trình bày cơ sở lý thuyết và các phương pháp thường dùng trong hệ khuyến nghị, các khó khăn chung thường gặp.
- **Chương 3**: Trình bày cơ sở lý thuyết, phương pháp tiếp cận cho bài toán khuyến nghị tin tức phù hợp sở thích người đọc.
- **Chương 4**: Trình bày về kiến trúc, các thành phần và việc hiện thực hệ thống.
- **Mục Tài liệu tham khảo**
- **Phụ lục A.** Giới thiệu về Apache Spark.
- **Phụ lục B.** Giới thiệu về Spring Boot.

Chương 2

TỔNG QUAN VỀ HỆ KHUYẾN NGHỊ

2.1 Mở đầu.

Dựa trên mục tiêu và đối tượng nghiên cứu, chương này sẽ phát biểu bài toán khuyến nghị ở mức tổng quát, trình bày các phương pháp tiếp cận căn bản nhằm giúp người đọc có cái nhìn tổng quát về hệ khuyến nghị.

2.2 Khái niệm hệ khuyến nghị.

Theo Ricci [3], hệ khuyến nghị là những công cụ phần mềm và kỹ thuật nhằm cung cấp, gợi ý những đối tượng mà có thể hữu ích với người dùng. Những gợi ý liên quan đến quyết định của người dùng như: nên mua gì, nên nghe nhạc gì hay nên đọc tin tức gì.

Hệ khuyến nghị sẽ chủ động hướng người dùng đến những đối tượng mà họ quan tâm, yêu thích. Điều này đặc biệt có ý nghĩa khi ngày nay, lượng thông tin được tạo ra quá lớn, dẫn đến tình trạng người dùng bị quá tải thông tin.

Hệ khuyến nghị đã chứng tỏ được giá trị của mình khi được áp dụng vào các hệ thống của các công ty lớn, có thể kể đến Google, Amazon, Netflix, LinkedIn [1].

[1]https://engineering.linkedin.com/recommender-systems/browsemap-collaborative-filtering-linkedin

2.3 Phát biểu bài toán khuyến nghị.

Hiện nay, có nhiều nghiên cứu về hệ khuyến nghị và có nhiều công trình cố gắng định nghĩa, phát biểu bài toán khuyến nghị một cách tổng quát nhất. Trong khóa luận này, em sử dụng định nghĩa của Adomavicius và cộng sự[4] để định nghĩa và phát biểu bài toán khuyến nghị.

Định nghĩa 2.1. Không gian người dùng là tập tất cả các người dùng được quan sát bởi hệ thống nhằm thực hiện khuyến nghị, ký hiệu là U, $U = \{u_1, u_2, ..., u_n\}$.

Định nghĩa 2.2. Không gian đối tượng khuyến nghị là tập hợp tất cả đối tượng sẽ được khuyến nghị cho người dùng. Ký hiệu là P, $P = \{p_1, p_2, ..., p_m\}$.

Định nghĩa 2.3. Hàm hữu ích là một ánh xạ $f : U \times P \to \mathbb{R}$, $\mathbb{R}$ thuộc tập hợp có thứ tự (ví dụ như số nguyên không âm hay số thực trong một khoảng nào đó). Hàm hữu ích f ước lượng mức độ hữu ích của $p \in P$ với $u \in U$.

Lưu ý rằng, đối tượng khuyến nghị được xác định tùy thuộc vào ngữ cảnh cụ thể như sách (với trường hợp của Amazon), phim (với trường hợp của Netflix),.... Trong khóa luận, đối tượng khuyến nghị là những tin tức, bài báo đã được đăng.

Phát biểu bài toán khuyến nghị.

Cho trước:

- Không gian người dùng: $U = \{u_1, u_2, ..., u_n\}$
- Không gian đối tượng khuyến nghị: $P = \{p_1, p_2, ..., p_m\}$

Mục đích của hệ khuyến nghị là đi tìm hàm hữu ích $f(u, p)$. Giá trị của hàm $f(u, p)$ giúp tiên đoán u thích p nhiều hay ít hay p hữu ích thế nào đối với u. Với mỗi người dùng $u \in U$, hệ khuyến nghị cần trả về một danh sách TopN các đối tượng khuyến nghị có giá trị hàm hữu ích được sắp xếp theo thứ tự giảm dần.

Tùy thuộc vào bài toán, ngữ cảnh cụ thể mà ta có cách ước lượng hàm hữu ích f phù hợp. Phần tiếp theo sẽ trình bày về các phương pháp tiếp cận truyền thống, phổ biến nhất hiện nay.

2.4 Các phương pháp tiếp cận phổ biến.

Những năm gần đây, nhiều phương pháp xây dựng hệ khuyến nghị đã được phát triển và theo Adomavicius và cộng sự[4], hệ khuyến nghị chia làm ba hướng tiếp cận chính:

- Tiếp cận nội dung (Content-based recommendations): Người dùng sẽ được khuyến nghị những đối tượng giống đối tượng mà người dùng yêu thích trong quá khứ.
- Tiếp cận lọc cộng tác (Collaborative recommendations): Người dùng sẽ được khuyến nghị đối tượng mà nhóm người có cùng sở thích với người dùng yêu thích trong quá khứ.
- Tiếp cận lai (Hybrid approaches): Đây là phương pháp kết hợp phương pháp tiếp cận nội dung và lọc cộng tác lại với nhau.

Tiếp theo, ta sẽ trình bày chi tiết và phần tích ưu, nhược điểm của từng phương pháp.

2.4.1 Tiếp cận nội dung (Content-based recommendations).

Trong phương pháp tiếp cận nội dung, chúng ta sử dụng đặc trưng, nội dung của đối tượng (item) và sở thích của người dùng trên một tập đối tượng (trong quá khứ) để ước lượng sở thích của người dùng. Sau đó, ta sẽ sử dụng độ ước lượng này để quyết định xem đối tượng nào sẽ phù hợp nhất với sở thích người dùng. Quá trình này còn được gọi là việc xây dựng hồ sơ người dùng. Ví dụ, trong hệ khuyến nghị phim, hệ thống sẽ cố gắng hiểu những tính chất chung nhất của các bộ phim mà người dùng cho điểm cao trong quá khứ (tính chất như diễn viên, đạo diễn, thể loại,...). Sau đó, chỉ những bộ phim có tương đồng cao với sở thích người dùng mới được khuyến nghị.

Để ước lượng có hay không người dùng u sẽ thích đối tượng khuyến nghị p và thích nhiều hay ít (tức việc xây dựng và ước lượng giá trị hàm hữu ích $f(u, p)$), các phương pháp dựa trên tiếp cận nội dung thông thường sẽ thực hiện các bước sau [5]:

- Bước 1: Biểu diễn nội dung của đối tượng khuyến nghị $p \in P$, ký hiệu $Content(p)$.
- Bước 2: Mô hình hóa sở thích người dùng $u \in U$, gọi tắt là hồ sơ người dùng, ký hiệu $UserProfile(u)$.

- Bước 3: Ước lượng giá trị hàm hữu ích dựa trên độ tương tự nội dung của đối tượng khuyến nghị p với hồ sơ người dùng u. Hệ thống sẽ ưu tiên khuyến nghị những đối tượng có nội dung tương tự cao so với hồ sơ người dùng u.

$$f(u,p) = Sim\big(UserProfile(u), Content(p)\big) \tag{2.1}$$

Biểu diễn nội dung đối tượng khuyến nghị.

Với mô hình không gian vector (như phương pháp biểu diễn TF/IDF), nội dung của đối tượng $p \in P$, ký hiệu $Content(p)$, được biểu diễn dưới dạng vector đặc trưng như sau:

$$Content(p) = \vec{w}_p = (w_{1,p}, w_{2,p}, ..., w_{k,p}) \tag{2.2}$$

Trong đó:

- k: là số đặc trưng để biểu diễn nội dung đối tượng.
- $w_{i,p}$: là trọng số đặc trưng thứ i của đối tượng p.

Ngoài ra, theo tác giả Isinkaye [6], ta có thể biểu diễn nội dung khuyến nghị mô hình xác suất, với các phương pháp như Navie Bayes, Decision Tree hoặc Neural Networks.

Xây dựng hồ sơ người dùng.

Hồ sơ người dùng giúp hệ thống có thể hiểu được sở thích và tiên đoán mức độ hữu ích của một đối tượng đối với người dùng. Hồ sơ người dùng được xây dựng dựa trên nội dung các đối tượng mà họ thể hiện sự quan tâm, đánh giá khi tương tác, sử dụng hệ thống [5]. Như vậy, nếu dùng mô hình không gian vector, hồ sơ người dùng u được biểu diễn bằng một vector đặc trưng với k chiều, bằng số chiều biểu diễn đối tượng.

$$UserProfile(u) = \vec{w}_u = (w_{1,u}, w_{2,u}, ..., w_{k,u}) \tag{2.3}$$

Trong đó:

- k: là số đặc trưng biểu diễn hồ sơ người dùng.
- $w_{i,u}$: là trọng số đặc trưng thứ i trong hồ sơ người dùng u.

Xác định hàm hữu ích.

Thông thường, khi hồ sơ người dùng và nội dung của đối tượng được mô hình bằng không gian vector, độ đo cosine được sử dụng làm hàm hữu ích. Khi đó, ta có công thức sau:

$$f(u,p) = cosine(\vec{w}_u, \vec{w}_p) = \frac{\vec{w}_u \bullet \vec{w}_p}{\parallel \vec{w}_u \parallel \cdot \parallel \vec{w}_p \parallel} \tag{2.4}$$

Trong đó, $\bullet$ thể hiện tích vô hướng giữa hai vector và $\parallel \vec{w} \parallel$ thể hiện độ dài của $\vec{w}$.

Ví dụ, nếu người dùng u đọc nhiều bài báo về chủ đề chính trị thì những bài báo này có nhiều từ ngữ, keyword (đặc trưng) liên quan đến chính trị hơn những bài báo khác. Do đó, giá trị trọng số các đặc trưng liên quan đến chính trị trong hồ sơ người dùng u, vốn được định nghĩa bằng vector đặc trưng $\vec{w}_u$, sẽ cao. Hệ quả là những bài báo nào có giá trị trọng số các đặc trưng liên quan đến chính trị cao sẽ được khuyến nghị cho người dùng.

Cập nhật hồ sơ người dùng.

Trên thực tế, sở thích người dùng thay đổi theo thời gian. Tùy từng lĩnh vực ứng dụng mà sở thích thay đổi nhanh hay chậm. Để đương đầu với tình trạng này, việc đề xuất các giải pháp khác nhau cho việc xây dựng và cập nhật hồ sơ người dùng là cần thiết.

Hạn chế, khó khăn.

Tiếp cận dựa trên nội dung bị giới hạn bởi những đặc trưng (features) của đối tượng mà hệ thống muốn khuyến nghị [4]. Do đó, để có đủ đặc trưng, nội dung phải được phân tích tự động bằng máy tính (trường hợp văn bản, hình ảnh,...) hoặc được gán bằng cách thủ công. Trong khi trích xuất thông tin (information retrievel) hoạt động tốt với dữ liệu văn bản thì một vài loại dữ liệu khác gặp khó khăn trong quá trình tự động phân tích đặc trưng như audio stream, video stream,... [4]

Một vấn đề khác liên quan đến việc phân tích nội dung, đó là nếu hai đối tượng hoàn toàn khác nhau được biển diễn bởi tập đặc trưng giống nhau, ta không thể phân biệt được chúng [4]. Hệ thống sẽ không thể phân biệt được chất lượng bài viết tốt hay không tốt nếu hai bài viết đó được biểu diễn bởi tập đặc trưng giống nhau.

Nếu có một người dùng mới tương tác với hệ thống, hệ khuyến nghị nội dung sẽ không thực hiên được vì nó chưa hiểu sở thích người dùng là gì.

2.4.2 Tiếp cận lọc cộng tác (Collaborative recommendations).

Trái ngược với hướng tiếp cận nội dung, tiếp cận lọc cộng tác cố gắng dự đoán độ hữu ích của đối tượng p cho người dùng u dựa trên những đối tượng được đánh giá bởi những người dùng khác, có sở thích, quan tâm tương tự u. [4]. Ý tưởng cơ bản của lọc cộng tác là nếu người dùng có cùng sở thích ở quá khứ thì sẽ có có cùng sở thích ở tưởng lai. Phần này, ta sẽ trình bày chi tiết về lọc cộng tác và các nghiên cứu liên quan.

Định nghĩa 2.1. Ma trận đánh giá

Cho không gian người dùng $U = \{u_1, u_2, ..., u_n\}$ và không gian các đối tượng khuyến nghị $P = \{p_1, p_2, ..., p_m\}$. Ma trận A kích thước $n \times m$, chứa các giá trị đánh giá $a_{i,j}$, với $i \in 1 \dots n, j \in 1 \dots m$. Giá trị đánh giá $a_{i,j}$ thể hiện mức độ hữu ích của đối tượng p_j với người dùng u_i (hay $f(u_i, p_j) = a_{i,j}$. Giá trị $a_{i,j}$ có thể là nguyên hay thực trong một khoảng cho trước tùy bài toán cụ thể. Nếu một người dùng u_i chưa thể hiện đánh giá đối tượng p_j thì $a_{i,j} = \oslash$ và cần được tính toán, xác định (dấu chấm hỏi (?) trong hình).

	p_1	p_2	p_3	p_4	p_5	...	p_m
u_1	1	?	5	?	4	?	?
u_2	?	2	?	4	3	?	?
u_3	?	2	3	?	?	?	1
u_4	4	1	?	?	3	?	?
...	?	?	?	?	?	?	?
u_n	?	3	?	5	1	?	5

Hình 2.1: Dấu ? là các giá trị cần tiên đoán trong ma trận đánh giá.

Các phương pháp lọc cộng tác nói chung được phân thành hai nhóm chính: lọc cộng tác dựa trên bộ nhớ (memory based) và lọc cộng tác dựa trên mô hình (model based).

Tiếp cận lọc cộng tác dựa trên bộ nhớ.

Các thuật toán dựa trên bộ nhớ tìm cách ước lượng hàm hữu ích $f(u, p)$ của đối tượng khuyến nghị p với người dùng u dựa trên những đánh giá của người đồng sở

thích của u đối với p (lọc dựa trên người dùng) hoặc dựa trên những đánh giá của u với các đối tượng khuyến nghị p' tương tự p (lọc dựa trên đối tượng khuyến nghị) [5]. Cụ thể, giá trị hàm hữu ích là trung bình trọng số của những đánh giá được tạo ra bởi những người dùng khác và trọng số chính là độ "tương tự" giữa các người dùng (hoặc đối tượng).

Lọc cộng tác dựa trên bộ nhớ được chia làm hai loại: lọc cộng tác dựa trên người dùng và lọc cộng tác dựa trên đối tượng khuyến nghị. Về cơ bản, các thuật toán lọc cộng tác dựa trên người dùng và lọc cộng tác dựa trên đối tượng khuyến nghị là tương tự nhau. Có chăng là kích thước của không gian người dùng và không gian đối tượng khuyến nghị sẽ ảnh ưởng đến tốc độ tính toán khi xác định nhóm các đối tượng tương tự.

Với các phương pháp lọc dựa trên người dùng, hệ thống thường phải thực hiện ba bước chính: một là xác định tập TopN những người có sở thích tương tự nhất với u, tức S_u; hai là ước lượng giá trị hàm hữu ích $f(u, p)$ của đối tượng khuyến nghị p cho người dùng u bằng cách tổng hợp những đánh giá của S_u đối với p; ba là thực hiện khuyến nghị dựa trên giá trị hàm hữu ích đã ước lượng được. Thực hiện tương tự với lọc cộng tác dựa trên đối tượng khuyến nghị.

Tiếp cận lọc cộng tác dựa trên mô hình

Khác với lọc cộng tác dựa trên bộ nhớ, lọc cộng tác dựa trên mô hình sẽ dùng tập các đánh giá có sẵn trong ma trận đánh giá A để học một mô hình đánh giá cho mỗi người dùng. Sau đó, mô hình học được sẽ dùng để tiên đoán cho các đối tượng khác. Có hai cách tiếp cận dựa trên mô hình: mô hình gom cụm (cluster model) và mô hình Bayesian.

Theo Adomavicius[4], việc xây dựng mô hình có thể dùng các kỹ thuật máy học (machine learning) và khai phá dữ liệu (data mining). Những kỹ thuật có thể khuyến nghị nhanh tập đối tượng mà nó dùng để xây dựng model và chúng đã được chứng minh là đưa ra kết quả khuyến nghị tương tự với kỹ thuật khuyến nghị dựa vào bộ nhớ. Một số kỹ thuật như hồi quy (Regression), gom cụm (Clustering), giảm chiều (Singular Value Decomposition - SVD).

Hạn chế, khó khăn.

Khác với tiếp cận nội dung, các hệ thống lọc cộng tác không bị các hạn chế về mặt phân tích nội dung vì chúng dùng thông tin từ ma trận đánh giá. Vì vậy, các hệ thống lọc cộng tác có thể áp dụng cho nhiều dạng đối tượng, nhiều kiểu nội dung khác nhau. Tuy nhiên, hệ thống lọc cộng tác vẫn có những giới hạn nhất định [4].

- **Người dùng mới:** Tương tự với tiếp cận nội dung, để có thể tạo ra những khuyến nghị chính xác, hệ thống phải học được sở thích của người dùng từ những tương tác, đánh giá trong quá khứ. Đối với người dùng chưa có hoặc có rất ít thông tin, hệ thống không thể biết được sở thích của người dùng. Do đó, không thể có những khuyến nghị hữu ích.

- **Đối tượng khuyến nghị mới:** Các đối tượng khuyến nghị mới được thêm vào hệ thống một cách thường xuyên. Các hệ thống lọc cộng tác chỉ dựa trên sở thích người dùng để thực hiện khuyến nghị, do đó nếu có những đối tượng mới chưa đủ thông tin đánh giá thì nó không thể thực hiện khuyến nghị. Vấn đề người dùng mới và đối tượng khuyến nghị mới còn được gọi là khởi động lạnh (cold start problem).

- **Ma trận thưa:** Trong bất kỳ hệ khuyến nghị nào, số lượng đánh giá quan sát được là rất nhỏ so với số lượng đánh giá cần phải tiên đoán. Điều đó ảnh hưởng đến độ chính xác tiên đoán, cũng như chất lượng đối tượng khuyến nghị.

- **Dữ liệu lớn:** Không gian người dùng và đối tượng khuyến nghị rất lớn, chi phí cho việc tìm người dùng lân cận theo đó cũng tăng theo.

2.4.3 Tiếp cận lai (Hybrid approaches).

Nhiều hệ thống khuyến nghị sử dụng tiếp cận lai bằng cách kết hợp tiếp cận lọc cộng tác và tiếp cận dựa trên nội dung, nhằm tránh đi những hạn chế của hai phương pháp này. Phương pháp lai kết hợp kết quả khuyến nghị của các phương pháp khuyến nghị khác theo một cách nào đó nhằm đưa ra một hệ thống khuyến nghị tốt hơn. Có nhiều cách lai khác nhau. Ở đây, ta sẽ khảo sát phương pháp lai đơn giản nhất là sử dụng phương pháp lai có trọng số (Weighted Hybrid).

Chúng ta đã biết, về cơ bản, mỗi phương pháp khuyến nghị phải đi tìm hàm hữu ích $f(u,p)$ của đối tượng p với người dùng u. Tiếp cận lai có trọng số sẽ tính toán giá trị hàm hữu ích $f_{hybrid}(u,p)$ dựa trên kết quả của tất cả các hàm hữu ích $f(u,p)$ của các phương pháp khuyến nghị khác tồn tại trong hệ thống, đơn giản nhất là ta kết hợp tuyến tính các hàm $f(u,p)$ này lại với nhau.

Theo tác giả Huỳnh Ngọc Tín [5], phương pháp lai có trọng số có ưu và nhược điểm như sau:

- **Ưu điểm:** Tất cả khả năng, phương pháp khác nhau của hệ thống được tham gia vào quá trình khuyến nghị một cách minh bạch, tự nhiên, dễ dàng thực hiện, dễ hàng hiệu chỉnh.
- **Hạn chế:** Việc ước lượng trọng số lớn hay nhỏ phải phù hợp với những phương pháp khác nhau.

2.5 Ứng dụng của hệ khuyến nghị.

Hiện nay, hệ khuyến nghị được sử dụng bởi nhiều công ty lớn, đặc biệt là trong lĩnh vực thương mại điện tử. Ta hãy đi khảo sát hệ khuyến nghị của một số công ty lớn. [1]

Đầu tiên, ví dụ tuyệt vời nhất chính là hệ thống khuyến nghị của LinkedIn's. Hệ thống này sẽ gợi ý những người mà bạn có thể biết. Thay vì phải gợi ý một lượng lớn "kết nối" (có khoảng 500 triệu người dùng đã đăng ký sử dụng LinkedIn's), hệ khuyến nghị sẽ khoanh vùng, giảm bớt chỉ còn một số "kết nối" phù hợp với bạn nhất, dựa trên dữ liệu của bạn như các "kết nối" mà bạn có.

Tiếp đến, phải kể đến Netflix. Theo một nghiên cứu từ McKinsey, 75% những gì người dùng coi trên Netflix đến từ hệ thống khuyến nghị.

Kế đến là Amazon. Cũng theo McKinsey, 35% những gì người dùng mua trên Amazon đến từ hệ khuyến nghị. Amazon sử dụng nhiều thuật toán khuyến nghị khác nhau để có thể đạt được điều này. Đầu tiên, họ cho người dùng xem những đối tượng/mặt hàng thường hay mua kèm với những đối tượng/mặt hàng mà người dùng đã chọn mua. Họ cũng có thể cho người dùng xem những đối tượng/mặt hàng tương tự với cái

[1] https://www.quora.com/Which-companies-use-recommender-recommendation-systems

mà người dùng vừa xem. Ngoài ra, họ còn gửi những đối tượng khuyến nghị qua hòm thư điện thử của người dùng.

Best Buy cũng là một công ty thành công trong việc ứng dụng hệ khuyến nghị khi vào quý 2 năm 2016, lợi nhuận của mảng bán hàng trực tuyến đã tăng 23.7%.

Cuối cùng, không thể không kể đến hệ khuyến nghị tin tức Google News của Google.

2.6 Xu hướng mới cho hệ khuyến nghị.

Luận văn Tiến sĩ của tác giả Huỳnh Ngọc Tín [5] tổng hợp, liệt kê một số xu hướng mới về hệ khuyến nghị mà cộng đồng đang quan tâm như sau:

- Stefanidis và cộng sự chỉ ra rằng, các phương pháp truyền thống chưa quan tâm xem xét sự ảnh hưởng của yếu tố thời gian, xu hướng đến kết quả khuyến nghị như thế nào.

- Làm thế nào để kết hợp thông tin xã hội rõ ràng, cũng như tiềm ẩn vào các phương pháp truyền thống.

- Sử dụng thông tin nhận biết ngữ cảnh (context-aware) để thực hiện khuyến nghị. Liên quan đến tiếp cận khai thác thông tin ngữ cảnh, Gediminas Adomavicius và cộng sự khảo sát phân tích các nghiên cứu khác nhau. Thông tin ngữ cảnh giúp các hệ khuyến nghị cung cấp những khuyến nghị phù hợp với người dùng theo thời gian, địa điểm. Cùng với sự phát triển mạnh mẽ của công nghệ di động, hướng tiếp cận khai thác thông tin ngữ cảnh được đánh giá là rất tiềm năng đối với các hệ thống khuyến nghị trên thiết bị di động.

- Tiếp cận lai nhằm giải quyết những hạn chế của mỗi phương pháp khác nhau.

- Thu thập, sử dụng thông tin tiềm ẩn của người dùng từ Internet để xác định sở thích của họ.

- Nhóm tác giả Gunawardana và Shani đã chỉ ra rằng, với mỗi bài toán cần có phương pháp khuyến nghị phù hợp. Đồng thời với mỗi phương pháp khuyến nghị, cũng cần có phương pháp đánh giá phù hợp. Phương pháp đánh giá sẽ quyết định độ chính xác tiên đoán, kết quả của phương pháp khuyến nghị. Việc

nghiên cứu phát triển các phương pháp đánh giá kết quả khuyến nghị cũng nằm trong xu hướng và quan tâm của cộng đồng.

2.7 Kết chương.

Trong chương này khóa luận đã trình bày tổng quan về hệ khuyến nghị, các hướng tiếp cận truyền thống, ưu và nhược điểm của từng hướng tiếp cận. Chương tiếp theo sẽ trình bày về các cơ sở lý thuyết của bài toán.

Chương 3

BÀI TOÁN KHUYẾN NGHỊ TIN TỨC PHÙ HỢP SỞ THÍCH NGƯỜI ĐỌC

3.1 Mở đầu

Chương trước đã trình bày khái quát về hệ khuyến nghị và các phương pháp tiếp cận truyền thống. Chương này sẽ phát biểu bài toán, trình bày phương pháp tiếp cận và nói chi tiết cách hiện thực từng thành phần của hệ thống.

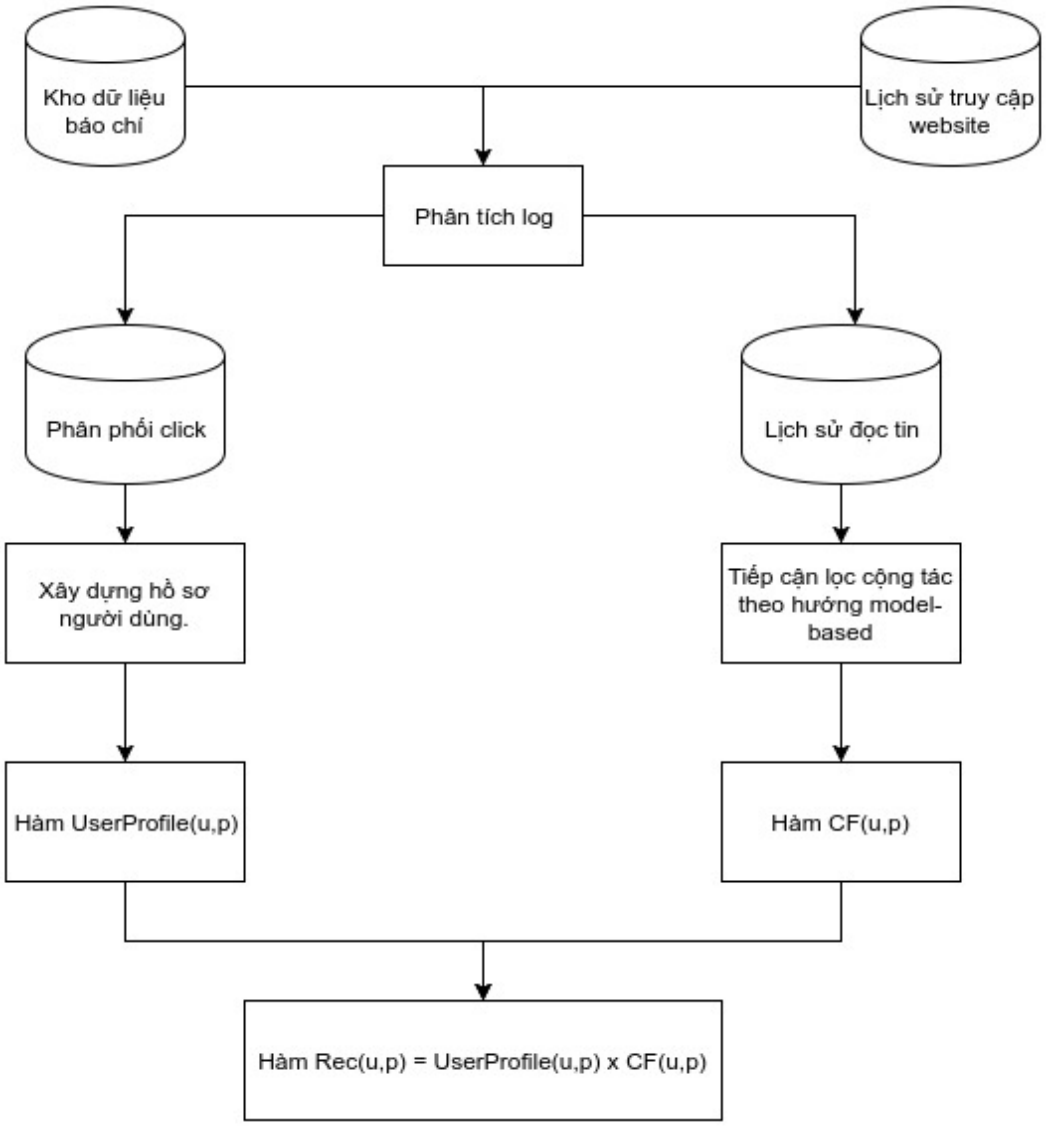
3.2 Phát biểu bài toán

Ta định nghĩa đầu vào và đầu ra của bài toán như sau:

- Đầu vào:
  - $U = \{u_1, u_2, ..., u_n\}$: Tập tất cả người dùng trong kho dữ liệu.
  - $P = \{p_1, p_2, ..., p_m\}$: Tập tất cả bài báo trong kho dữ liệu.
  - Lịch sử đọc tin tức của người dùng.
- Đầu ra:
  - Hàm hữu ích $f(u, p)$ xác định giá trị hữu ích của của bài báo p với người dùng u

Lưu ý rằng, đầu vào của bài toán có được sau khi thực hiện phân tích lịch sử truy cập các trang tin tức của người dùng. Dựa trên những nghiên cứu của Google về hệ

khuyến nghị tin tức phù hợp sở thích người đọc được áp dụng cho hệ thống Google News, khóa luận sẽ hiện thực lại một phần hệ thống này và được mô tả tổng quát ở hình sau:



Hình 3.1: Mô tả hệ thống khuyến nghị tin tức phù hợp sở thích người đọc.

Về cơ bản, từ dữ liệu thô ban đầu (kho dữ liệu tin tức và lịch sử truy cập trang web), ta sẽ phân tích, trích xuất các thông tin cần thiết, loại bỏ nhiễu (bad record) và lưu lại vào kho dữ liệu. Dữ liệu từ kho này sẽ phục vụ cho việc xây dựng hồ sơ người dùng cũng như các thuật toán lọc cộng tác. Xây dựng hồ sơ người dùng và các thuật toán lọc cộng tác đều cho ra một dạng hàm hữu ích phản ánh mức độ yêu thích của người dùng với đối tượng được khuyến nghị. Sau cùng, ta sẽ kết hợp các hàm hữu ích

này lại để cho ra hàm hữu ích sau cùng. Phần tiếp theo sẽ trình bày nguyên nhân đàng sau của việc kết hợp các thuật toán lọc cộng tác và việc xây dựng hồ sơ người dùng, cũng như đi sâu vào chi tiết của từng phần.

3.3 Phương pháp tiếp cận

Lọc cộng tác *(Collaborative Filtering)* là một kỹ thuật cho phép học được sở thích của người dùng và tạo ra nội dung khuyến nghị dựa trên dữ liệu của người dùng và cộng đồng. Tuy nhiên, lọc cộng tác gặp phải hai hạn chế chính[2]. Thứ nhất, hệ thống không thể khuyến nghị tin chưa được đọc bởi bất kỳ người dùng nào hoặc được đọc rất ít. Hay nói cách khác, chúng ta không có đủ thông tin để thực hiện các phép suy luận cho người dùng. Đối với hệ thống khuyến nghị tin tức, đây là một vấn đề quan trọng vì các trang tin có xu hướng cập nhật thông tin mới nhất và phải mất một khoảng thời gian để hệ thống thu thập đủ thông tin để có thể khuyến nghị. Thứ hai, không phải mọi người dùng đều ngang hàng nhau. Ví dụ, tin tức giải trí rất phổ biến nên nó có xu hướng được khuyến nghị cho cả những người dùng chưa từng click vào bởi vì những tin tức này luôn có đủ click bởi những người dùng lân cận.

Một giải pháp cho những hạn chế của lọc cộng tác là xây dựng hồ sơ người dùng. Hồ sơ này được xây dựng dựa trên hành vi, sở thích của người dùng trong quá khứ và dùng để dự đoán sở thích hiện tại của người dùng. Nó sẽ giúp lọc bớt những tin tức mà người dùng không thích hoặc khuyến nghị những tin tức phù hợp sở thích người dùng nhưng chưa được click bởi những người dùng khác.

3.4 Khó khăn, thách thức

- Đối với phương pháp khuyến nghị lọc cộng tác:
  - Dữ liệu lớn.
  - Ma trận thưa.
  - Người dùng mới.
  - Đối tượng khuyến nghị mới.
- Đối với việc xây dựng hồ sơ người dùng:

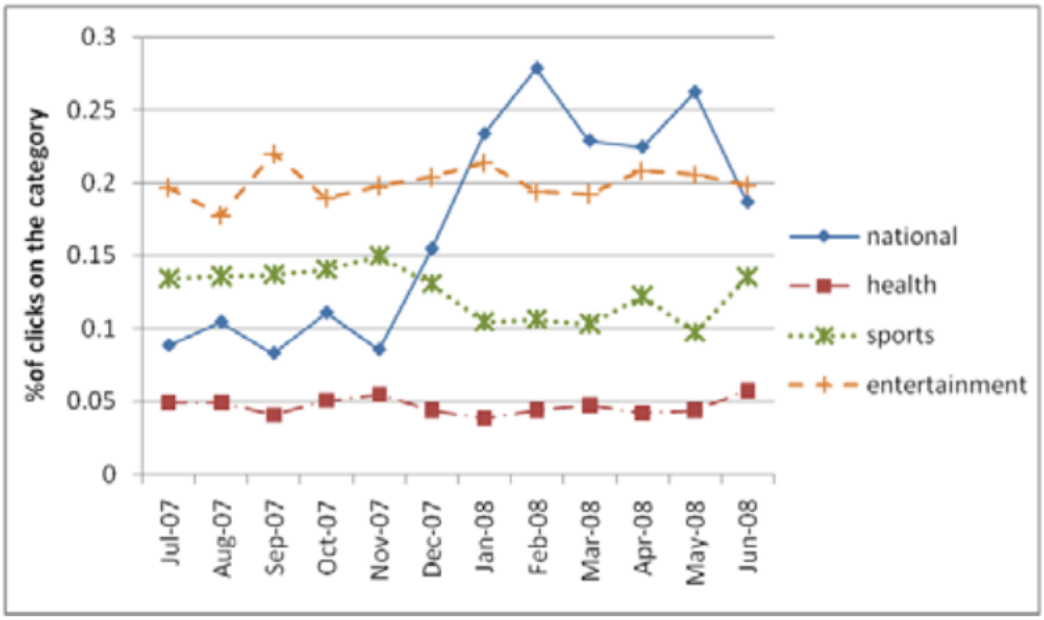
– Sở thích của người dùng thay đổi theo thời gian nên việc cập nhật hồ sơ người dùng thường xuyên là cần thiết. Tuy nhiên, khi dữ liệu lớn dần lên, việc cập nhật thường xuyên này sẽ rất tốn thời gian. Do đó, đòi hỏi phải áp dụng các framework hỗ trợ việc tính toán, xử lý phân tán.

– Một số hệ thống dựa vào các hành vi tường minh (explicit behaviour) để xây dựng hồ sơ người dùng như: hành vi like, dislike, cho điểm rating,.. Tuy nhiên, việc này gây "gánh nặng" lên người dùng, chỉ một số ít người dùng sẵn lòng phản hồi mà thôi. Do đó, hệ thống phải có khả năng xây dựng hồ sơ dựa vào các tương tác của người dùng trên hệ thống.

3.5 Xây dựng hồ sơ người dùng

3.5.1 Ý tưởng

Hồ sơ người dùng được xây dựng dựa trên hành vi, sở thích của người dùng trong quá khứ và dùng để dự đoán sở thích tương lai của người dùng. Theo Liu và cộng sự [2], một vấn đề quan trọng trong việc mô hình hóa người dùng, đặc biệt với tin tức, đó là sở thích người dùng thay đổi theo thời gian và lịch sử càng xa thì càng ít ảnh hưởng tới việc dự đoán sở thích ở tưởng lai. Billsus và Pazzani [7] cho rằng, sở thích người dùng được chia làm hai loại: ngắn hạn (shor-term) và dài hạn (long-term). Sở thích ngắn hạn thường liên quan đến tin tức, sự kiện nóng và thay đổi nhanh chóng. Ngược lại, sở thích dài hạn thường phản ảnh thực chất sở thích của người dùng.



Hình 3.2: Phân phối sở thích của người dùng Mỹ theo thời gian.
(Nguồn: Trích trong bài báo *"Personalized news recommendation based on click behavior"*.)

Hình 3.2 thể hiện phân phối click của dân Mỹ theo thời gian (giai đoạn 07.2007 - 06.2008). Để cho đơn giản, hình trên chỉ thể hiện bốn chủ đề. Ta thấy rằng, có sự biến động về sở thích của người dùng theo thời gian. Chủ đề về "national" có sự biến động lớn khi so sánh với các chủ đề khác (như "health"). Điều này chỉ ra rằng đang có một sự kiến lớn về "national", thu hút nhiều quan tâm của dư luận nhiều hơn "health". Cần nhớ lại, năm 2008 diễn ra cuộc bầu cử tổng thống Mỹ.

Theo Liu [2], để xây dựng được hồ sơ người dùng, ta cần định nghĩa một số khái niệm liên quan sau:

- $C = \{c_1, c_1, ..., c_n\}$: tập hợp chứa các chủ đề mà các bài báo được phân loại vào như thể thao, kinh tế, chính trị, ...
- t: khoảng thời gian (time period).
- u: người dùng (user)
- $D(u, t)$: phân phối click *(click distribution)* vào mỗi chủ đề c_i của người dùng u

trong khoảng thời gian t. $D(u,t)$ được tính theo công thức:

$$D(u,t) = (\frac{N_1}{N_{total}}, \frac{N_2}{N_{total}}, ..., \frac{N_n}{N_{total}}), \quad N_{total} = \sum_{i=1}^{n} N_i \tag{3.1}$$

N_i là số lượng click vào bài báo thuộc chủ đề c_i được tạo bởi người dùng u trong khoảng thời gian t. $D(u,t)$ có thể đại diện cho lượng thời gian mà người dùng sử dụng để đọc tin tức về một chủ đề nào đó và nó phản ảnh sự phân phối sở thích của người dùng trong khoảng thời gian t.

- $D(t)$: phân phối click vào chủ đề c_i được tạo ra bởi tất cả người dùng của một quốc gia/vùng lãnh thổ trong khoảng thời gian t. $D(t)$ được tính theo công thức:

$$D(t) = (\frac{N_1}{N_{total}}, \frac{N_2}{N_{total}}, ..., \frac{N_n}{N_{total}}), \quad N_{total} = \sum_{i=1}^{n} N_i \tag{3.2}$$

N_i là số lượng click vào bài báo thuộc chủ đề c_i được tạo bởi tất cả người dùng trong khoảng thời gian t. $D(t)$ phản ánh xu hướng tin tức *(news trend)* và xu hướng này tương ứng với một sự kiện lớn nào đó. Ví dụ như bầu cử tổng thống Mỹ. Lưu ý thêm rằng, tại một thời điểm, có thể tồn tại nhiều xu hướng tin khác nhau ứng với những vị trí địa lý khác nhau.

3.5.2 Các bước thực hiện

Để thực hiện xây dựng hồ sơ người dùng, ta sẽ thực hiện tuần tự các bước sau: đầu tiên, hệ thống dự đoán sở thích thực chất của người dùng trong từng khoảng thời gian. Tiếp theo, để tính sở thích dài hạn của người dùng, ta kết hợp sở thích thực chất của người dùng trong nhiều khoảng thời gian lại. Cuối cùng, hệ thống sẽ dự đoán sở thích hiện tại của người dùng bằng cách kết hợp sở thích dài hạn của người dùng và xu hướng tin tức hiện tại tại nơi người dùng đó sống.

Bước 1: Tính sở thích thực chất của người dùng trong từng khoảng thời gian

Sở thích thực chất của người dùng vào chủ đề c_i chính là xác suất mà người dùng sẽ click vào tin tức thuộc chủ đề c_i và được mô hình hóa bởi công thức $p^t(click|category =$

c_i). Sử dụng luật Bayesian, $p^t(click|category = c_i)$ được tính bằng công thức:

$$\begin{aligned} interest^t(category = c_i) &= p^t(click|category = c_i) \\ &= \frac{p^t(category = c_i|click)p^t(click)}{p^t(category = c_i)} \end{aligned} \tag{3.3}$$

trong đó:

- $p^t(category = c_i|click)$: là xác suất mà click của người dùng thuộc vào chủ đề c_i. Nghĩa là, khi đã click thì xác suất mà người dùng click vào chủ đề c_i là bao nhiêu. Ta có thể ước lượng xác suất này bằng phân phối click $D(u,t)$ quan sát được trong khoảng thời gian t. $D(u,t)$ được định nghĩa tại công thức (3.1).

- $p^t(click)$: là xác suất mà người dùng sẽ click vào một bài báo mà không quan tâm đến chủ đề nó thuộc về.

- $p^t(category = c_i)$: là xác suất mà một bài báo thuộc về chủ đề c_i. Đây chính là tỉ lệ bài báo về chủ đề c_i được xuất bản trong khoảng thời gian t. Tại cùng thời điểm t, tỉ lệ này có liên quan đến xu hướng tin tại khu vực mà người dùng sống. Càng nhiều sự kiện về chủ đề c_i thì càng có nhiều bài viết về chủ đề c_i. Do đó, ta có thể xấp xỉ xác suất này bằng phân phối click của được tạo ra bởi tất cả người dùng $D(t)$. $D(t)$ được định nghĩa tại công thức (3.2).

Theo công thức 3.3, ta kết luận rằng: nếu người dùng u đọc nhiều tin tức về chủ đề c_i, trong khi rất nhiều người dùng cũng đọc tin tức về chủ đề này thì có thể, sở thích thực chất của người dùng u không phải là c_i. người dùng đọc tin về c_i bởi vì nó đang là xu hướng mà thôi.

Bước 2: Tính sở thích dài hạn của người dùng.

Công thức 3.3 tính sở thích thực chất của người dùng dựa trên phân phối click trong trong một khoảng thời gian cụ thể t nào đó. Để tính toán sở thích thực chất của người dùng một cách chính xác hơn, ta sẽ kết hợp nhiều khoảng thời gian lại với nhau. Sở thích dài hạn của người dùng được tính bởi công thức:

$$\begin{aligned} interest(category = ci) &= \frac{\sum_t \left(N^t \times interest^t(category = c_i)\right)}{\sum_t N^t} \\ &= \frac{\sum_t \left(N^t \times \frac{p^t(category=c_i|click)p^t(click)}{p^t(category=c_i)}\right)}{\sum_t N^t} \end{aligned} \tag{3.4}$$

trong đó N^t là tổng số lượng click được tạo bởi user u trong khoảng thời gian t. Giả sử, xác suất mà người dùng click vào một tin tức bất kỳ là hằng số theo thời gian. Khi đó, công thức 3.4 trở thành:

$$interest(category = ci) = \frac{p(click) \times \sum_t \left(N^t \times \frac{p^t(category=c_i|click)}{p^t(category=c_i)} \right)}{\sum_t N^t} \tag{3.5}$$

Bước 3: Tính sở thích hiện tại của người dùng.

Như đã nói ở trên, sở thích của người dùng được chia làm hai loại: sở thích dài hạn và sở thích ngắn hạn. Sở thích dài hạn phản ánh sở thích thực chất của người dùng, còn ngắn hạn phản ánh sự ảnh hưởng của xu hướng tin nơi người dùng sống. Ở bước 2, ta đã tính được sở thích dài hạn của người dùng. Để có được xu hướng tin tức hiện tại, ta sử dụng phân phối click được tạo ra bởi tất cả người dùng trong một khoảng thời gian ngắn (ví dụ trong vài giờ hoặc một ngày), đại diện bởi $p^0(category = c_i)$ (hay $D(0)$). Bởi vì lượng người dùng lớn, nên chỉ cần một khoảng thời gian ngắn, sẽ có đủ click để ta ước lượng được xu hướng tin hiện tại.

Mục tiêu cuối cùng là dự đoán phân phối click của người dùng trong tương lai gần hay nói cách khác chính là sở thích hiện tại của người dùng. Ta lại tiếp tục dùng luật Bayesian:

$$\begin{aligned} currentInterest &= p^0(category = c_i|click) \\ &= \frac{p^0(click|category = c_i)p^0(category = c_i)}{p^0(click)} \end{aligned} \tag{3.6}$$

trong đó:

- $p^0(click|category = c_i)$: là sở thích dài hạn của người dùng, được tính bởi công thức 3.5
- $p^0(category = c_i)$: là xu hướng tin tức hiện tại.
- $p^0(click)$: là xác suất mà người dùng click vào một bài báo mà không quan tâm đến chủ đề của nó.

Như đã đề cập ở trên, ta coi xác suất $p^0(click)$ là hằng số nên, ta có:

$$\begin{aligned} p^0(category = c_i|click) &= \frac{interest(category = c_i)p^0(category = c_i)}{p(click)} \\ &= \frac{p^0(category = c_i) \times \sum_t \left(N^t \times \frac{p^t(category=c_i|click)}{p^t(category=c_i)} \right)}{\sum_t N_t} \end{aligned} \tag{3.7}$$

Tiếp theo, ta sẽ thêm một biến số G để làm mịn việc tính sở thích hiện tại của người dùng. G là số lượng click ảo, $G = 10$. Khi hệ thống quan sát thấy người dùng ít hoặc không tương tác với hệ thống, thể hiện qua việc có ít hoặc không có click, hệ thống sẽ dự đoán sở thích người dùng dựa trên xu hướng tin hiện tại. Mặt khác, nếu $\sum_t N_t >> G$, hệ thống dự đoán dựa trên sở thích thực chất của người dùng. Công thức cuối cùng có dạng:

$$p^0(category = c_i|click) = \frac{p^0(category = c_i) \times \left(\sum_t \left(N^t \times \frac{p^t(category=c_i|click)}{p^t(category=c_i)} \right) + G \right)}{\sum_t N_t + G} \tag{3.8}$$

3.5.3 Một số lưu ý

Đối với khóa luận, sở thích dài hạn của người người dùng sẽ được tính trong vòng một năm, xu hướng tin hiện tại được lấy trong ngày hôm trước (ngày trước khi chạy thuật toán xây dựng hồ sơ người dùng) và mỗi khoảng thời gian t có độ dài 15 ngày.

3.6 Thuật toán LSH với phương pháp MinHash

3.6.1 Ý tưởng

Đối với lớp bài toán đòi hỏi việc tìm kiếm các đối tượng tương tự nhau như tìm láng giềng gần nhất (nearest neighbor), near-duplicate detection, gom cụm (clustering),... thì số lượng phép so sánh cần thực hiện là rất lớn. Khi dữ liệu lớn lên, chi phí và thời gian thực hiện cũng tăng theo. Thay vào đó, ta có thể giải bài toán tìm *xấp xỉ* láng giềng gần nhất. Một phương pháp để giải quyết vấn đề này là sử dụng Locality Sensitive Hashing *(LSH)*, một thuật toán được đề xuất bởi Piotr Indyk và cộng sự [8]. Ý tưởng chính của LSH là thay vì phải đi so sánh một item với tất cả item còn lại, thì nay, ta chỉ so sánh item đó với những item có khả năng tương tự cao với item đó mà

thôi.

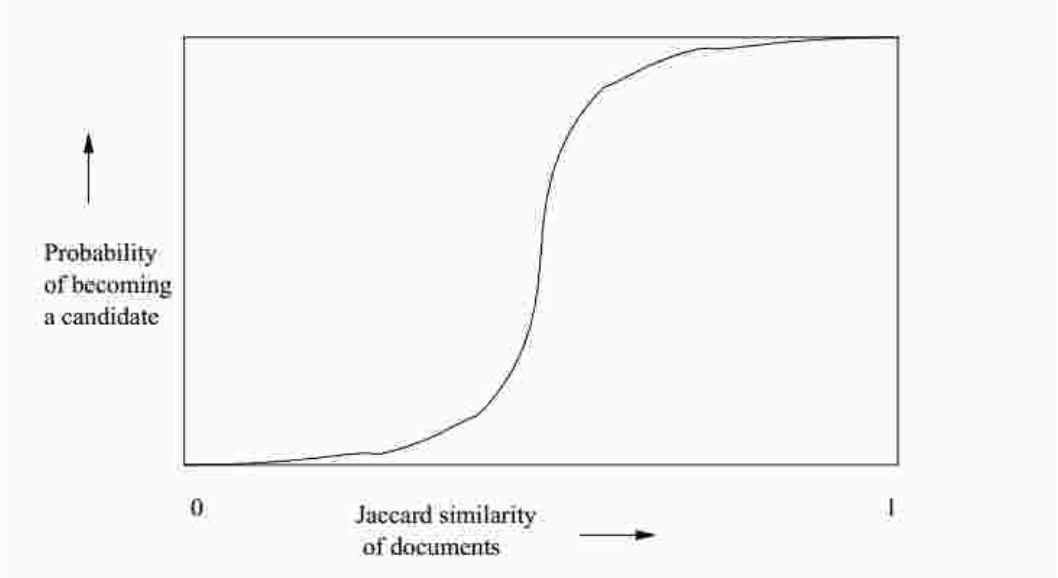
LSH hoạt động bằng cách thông qua hàm hash, gán hash code cho các item, những item nào trùng hash code thì được xem là một candidate pair (tạm dịch: cặp tiềm năng). Từ đó, ta chỉ cần xét các item là candidate pair với nhau mà không cần phải so sánh hết tất cả item. Hàm hash được chọn có tính chất: các item càng tương tự nhau thì sẽ có khả năng trùng hash code càng cao. $\boldsymbol{Probability\big(H(a) = H(b)\big) \sim Similarity(a, b)}$. Tùy thuộc vào độ đo tương tự nào được sử dụng, sẽ có cách thiết kế hàm hash phù hợp với độ đo đó.

3.6.2 Xác định cặp ứng viên bằng phương pháp chia band.

Giả sử ta có n hàm hash, ta sẽ chia n hàm hash thành thành từng nhóm (còn được gọi là band), mỗi band có r hàm hash. Gọi số lượng band là b, ta có $b \times r = n$. Giả sử $n = 100, b = 20, r = 5$. Lúc này, ta có 20 band, mỗi band gồm 5 hàm hash. Band 1 là tập hợp các hàm $\{h_1, h_2, h_3, h_4, h_5\}$, band 2 gồm các hàm $\{h_6, h_7, h_8, h_9, h_{10}\}$,...

Hai item A và B được gọi là một candidate pair nếu tồn tại ít nhất một band mà tất cả hash hash code trong band đó đôi một bằng nhau hay $\exists b : h_i(A) = h_i(B), i = 1...r$

Giả sử, độ tương đồng giữa hai item A và B là s, xác suất để A và B là 1 cặp candidate pair (hay xác suất tồn tại ít nhất 1 band mà A, B có hash code đôi một bằng nhau) là $1 - (1 - s^r)^b$. Hàm $1 - (1 - s^r)^b$ có hình dạng đường cong chữ S, được miêu tả bởi hình 3.3. Ngưỡng là giá trị của s mà xác suất trở thành một cặp candiate pair là 1/2. Ta nhận thấy rằng, khi độ tương đồng của một cặp item vượt qua ngưỡng, xác xuất chúng trở thành một cặp candidate pair tăng nhanh.



Hình 3.3: Mối liên hệ giữa độ tương đồng và xác suất trở thành một cặp candidate pair.

(Nguồn: Trích trong sách *Mining of massive datasets.*)

3.6.3 MinHash

Như đã nói ở phần ý tưởng, tùy thuộc vào độ đo tương tự nào được sử dụng, ta sẽ có cách thiết kế hàm hash phù hợp với độ đo đó, sao cho $Probability\big(H(a) = H(b)\big) \sim Similarity(a, b)$. Trong bài toán này, ta dùng độ đo Jaccard và theo như tác giả Leskovec [9] đề cập trong cuốn sách *Mining of massive datasets*, ta dùng phương pháp hash có tên MinHash.

Trong Khoa học máy tính, MinHash *(the min-wise independent permutations locality sensitive hashing scheme)* là kỹ thuật cho phép ước lượng nhanh độ tương tự Jaccard của hai tập hợp mà không cần phải tính toán giao và hợp của hai tập hợp đó.

Để hiểu rõ được MinHash, ta sẽ lần lượt định nghĩa và giải thích các thành phần liên quan.

Độ tương tự Jaccard

Độ tương tự Jaccard của hai tập hợp được định nghĩa là tỉ lệ giữa kích thước tập giao và tập hợp của hai tập hợp. Độ tương tự Jaccard được công thức hóa như sau:

$$J(A, B) = \frac{\mid A \cap B \mid}{\mid A \cup B \mid} \tag{3.9}$$

$J(A, B)$ càng cao thì hai tập A, B càng có nhiều phần tử giống nhau, càng tương tự nhau và ngược lại.

Characteristic matrix

Characteristic matrix (ma trận đặc trưng) là ma trận đặc trưng mô tả sự tương quan giữa các item theo đặc trưng cụ thể để tiến hành gom cụm. Characteristic matrix có cột đại diện cho item và dòng là đặc trưng của các item. Quy ước, ma trận này có M dòng. Dòng r và cột c có giá trị là 1 nếu item tại cột c có đặc trưng là dòng r. Ngược lại, giá trị sẽ là 0.

Ví dụ: Item 1, item 2, item 3, item 4 gồm các đặc trưng thuộc lần lượt các tập hợp sau: $S_1 \in \{a, d\}, S_2 \in \{c\}, S_3 \in \{b, d, e\}, S_4 \in \{a, c, d\}$. Biểu diễn characteristic matrix cho 4 item này.

Item	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Hình 3.4: Characteristic matrix biểu diễn 4 item

MinHash

Để tìm giá trị MinHash cho một tập bất kỳ, ta làm hai bước sau:

- Chọn một hoán vị các dòng của characteristic matrix.
- Định nghĩa một hàm hash $h(S)$ sao cho giá trị của $h(S)$ là chỉ số dòng đầu tiên (theo thứ tự hoán vị) mà dòng đó có giá trị là 1.

Giả sử, kết quả sau khi hoán vị dòng characteristic matrix ở hình 3.4 là:

Item	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

Hình 3.5: Characteristic matrix biểu diễn 4 item sau khi hoán vị dòng

Khi đó, giá trị MinHash của S_1, S_2, S_3, S_4 lần lượt là $h(S_1) = a, h(S_2) = c, h(S_3) = b$ và $h(S_4) = a$.

MinHash Signature

Vì characteristic matrix có số dòng và cột rất lớn nên việc tính toán, tìm tương đồng sẽ rất tốn kém. Do đó, ta sẽ dùng một ma trận khác có kích thước nhỏ hơn nhưng cho kết quả xấp xỉ ma trận cũ. Ma trận đó gọi là signature matrix. Signature matrix là một ma trận có cùng số cột với characteristic matrix nhưng chỉ có n dòng (n « M, M là số dòng của characteristic matrix).

Để tạo ra được signature matrix với n dòng, ta cần chọn ngẫu nhiên n hoán vị từ ma trận characteristic matrix. Tuy nhiên, việc hoán vị hàng triệu hoặc hàng tỉ dòng của characteristic matrix là tốn thời gian. Do đó, ta sẽ mô phỏng quá trình tạo ra n hoán vị từ ma trận characteristic matrix bằng cách dùng n hàm hash ngẫu nhiên mà số bucket bằng với số dòng của characteristic matrix. Vậy, thay vì chọn n hoán vị ngẫu nhiên từ M dòng, ta chọn n hàm hash ngẫu nhiên.

Gọi $h_1, h_2, ..., h_n$ lần lượt là các hàm hash ngẫu nhiên và $h_1(S), h_2(S), ..., h_n(S)$ lần lượt là giá trị MinHash của tập S tương ứng với n hàm hash ngẫu nhiên. Gọi vector $[h_1(S), h_2(S), ..., h_n(S)]$ là MinHash signature của tập S. Signature matrix là ma trận có cùng số cột với characteristic matrix và cột thứ i của được thay thế bởi MinHash signature S_i.

Ví dụ, cho characteristic matrix và hai hàm hash h_1 và h_2 như bảng sau. Tính signature matrix ?

Default Order	S1	S2	S3	S4	h1 = x+1 mod 5	h2 = 3x+1 mod 5
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Hình 3.6: Characteristic matrix và hai hàm h_1, h_2

Như đã nói ở trên, ta sẽ không hoán vị các dòng mà sẽ dùng hàm hash để mô phỏng cho quá trình này (ở đây ta dùng hàm h_1 và h_2). Giả sử, ban đầu dòng 0 là có giá trị

$(1, 0, 0, 1)$ thì sau khi qua hàm h_1, dòng 0 là $(0, 0, 1, 0)$ và qua hàm h_2, dòng 0 có giá trị là $(1, 0, 1, 1)$.

Dựa vào định nghĩa, ta có giá trị MinHash $h_1(S_1) = 1, h_1(S_2) = 3, h_1(S_3) = 0, h_1(S_4) = 1, h_2(S_1) = 0, h_2(S_2) = 2, h_2(S_3) = 0, h_2(S_4) = 0$. Từ đó, ta có signature matrix như sau:

	S1	S2	S3	S4
h1	1	3	0	1
h2	0	2	0	0

Hình 3.7: Singature matrix

Mã giả

Mã giả cho việc tính signature matrix từ characteristic matrix và n hàm hash ngẫu nhiên cho trước.

Algorithm 1 Thuật toán tính MinHash Signature

Input: Characteristic Matrix, tập các hàm hash $(h_1, h_2, ..., h_n)$
Output: Signature Matrix

```
foreach dòng r trong Characteristic Matrix do
    foreach hàm h_i, i = 1...n do
        Tính h_i(r)
    end for
    foreach cột c trong Characteristic Matrix do
        if Characteristic[r][c] == 1 then
            foreach j=1...n do
                SignatureMatrix[j][c] = min(SignatureMatrix[j][c], h_j(r))
            end for
        end if
    end for
end for
```

3.6.4 Áp dụng LSH vào bài toán khuyến nghị tin tức phù hợp sở thích người dùng

Tới bước này, ta đã có 2 công cụ giúp tính nhanh độ đo Jaccard nhưng vẫn đảm bảo kết quả xấp xỉ kết quả ban đầu. MinHash giúp chuyển characteristic matrix ban đầu về signature matrix có số dòng nhỏ hơn (n«M) và LSH giúp ta giảm số lần so sánh giữa các item (thay vì phải so sánh toàn bộ thì nay, ta chỉ so sánh những item có khả

năng tương đồng cao mà thôi). Ở đây, ta sẽ mô tả thuật toán LSH với phương pháp MinHash.

Mã giả

Algorithm 2 Thuật toán LSH với phương pháp MinHash

Input: CM, $band, row$
Output: hashTable

```
1: hashFunctions = Khởi tạo ngẫu nhiên b * r hàm hash
2: foreach item in CM do
3:    foreach i, i = 0...band do
4:       concatHashValue = ""
5:       foreach j, j = 0...row do
6:          concatHashValue += hashFunctions[i*row + j].hash(item)
7:       end for
8:       hashTable.add((i,concatHashValue),item.Id)
9:    end for
10: end for
```

Chú tích thuật toán:

- CM: Characteristic matrix
- band: số lượng band
- row: số lượng dòng trong 1 band hay nói cách khác, là số hàm hash trong 1 band.
- hashTable: kết quả sau khi chạy thuật toán. HashTable chứa thông tin $((bandID, concatHash$ Thông tin này được dùng để tìm danh sách $itemID$ có chung $(bandID, concatHashCode)$ hoặc để tìm danh sách $(bandID, concatHashCode)$ mà $itemID$ thuộc về.

3.6.5 Cách sử dụng cụm người dùng để khuyến nghị

Sau khi chạy thuật toán LSH, ta có được thông tin $((bandID, concatHashCode), itemID)$. Theo như sự gợi ý ở bài báo *Google news personalization: scalable online collaborative filtering* [1], ta sẽ bỏ đi $bandID$ và coi $concatHashCode$ như là $clusterID$. Từ đó, ta có được cặp key-value với key là $clusterID$ và value là $itemID$. Tiếp theo, ta tiến hành gom nhóm $itemID$ có cùng $clusterID$ và bỏ cluster có ít thành viên. Đồng thời, ta cũng phải tạo một bảng chứa danh sách clusterID và lịch sử click bởi một userID. Bảng này được gọi là UserTable **(UT)**.

Tới đây, ta sẽ tính điểm khuyến nghị cho một tin ứng viên s với người dùng u dựa vào các thông tin đã chuẩn bị ở trên.

Mã giả

Algorithm 3 Thuật toán tính điểm khuyến nghị LSH

Input: s, u, UT
Output: score
1: Lấy danh sách cluster mà người dùng u thuộc về
2: **foreach** $cluster_i$ thuộc danh sách cluster **do**
3: $clickCount_s$ = tổng lượng click vào tin s của tất cả thành viên trong $cluster_i$
4: $clickCount_{all}$ = tổng lượng click được tạo ra bởi tất cả thành viên trong $cluster_i$
5: $clusterScore_i = clickCount_s / clickCount_{all}$
6: **end for**
7: $score$ = tổng tất cả $clusterScore_i$ / số lượng cluster

Chú tích thuật toán:

- s: tin ứng viên cần tính điểm khuyến nghị
- u: người dùng
- UT:bảng UserTable
- score: điểm khuyến nghị. $0 < score < 1$

3.7 Kết hợp các thành phần lại với nhau

Phần này sẽ trình bày cách kết hợp hồ sơ người dùng và điểm khuyến nghị LSH để tạo ra điểm khuyến nghị sau cùng cho người dùng u

$$Rec(article) = BuildProfile(article) * LSH(article) \tag{3.10}$$

Với:

- $Rec(article)$: điểm khuyến nghị sau cùng cho một bài báo
- $BuildProfile(article)$: sở thích về chủ đề mà bài báo thuộc về. (Như đã biết, sau khi xây dựng hồ sơ người dùng, ta có được sở thích người dùng về một chủ đề)
- $LSH(article)$: điểm khuyến nghị cho bài báo khi sử dụng thuật toán gom cụm LSH

3.8 Kết chương

Chương này trình bày chi tiết toàn bộ ý tưởng, phương pháp tiếp cận cho bài toán khuyến nghị tin tức phù hợp sở thích người đọc, từ việc xây dựng hồ sơ người dùng để phản ánh sở thích về một chủ đề nào đó tới việc gom cụm người dùng và sử dụng cụm đó để tính điểm khuyến nghị. Chương tiếp theo sẽ trình bày về việc hiện thực các lý thuyết đã nêu ở chương này thành một hệ thống thực tế.

Chương 4

HIỆN THỰC HỆ THỐNG.

4.1 Mở đầu

Chương trước đã trình bày chi tiết về cơ sở lý thuyết, phương pháp để xây dựng hệ thống. Chương này, chúng ta sẽ đi sâu hơn, chi tiết hơn về kiến trúc, công nghệ xử lý bên dưới.

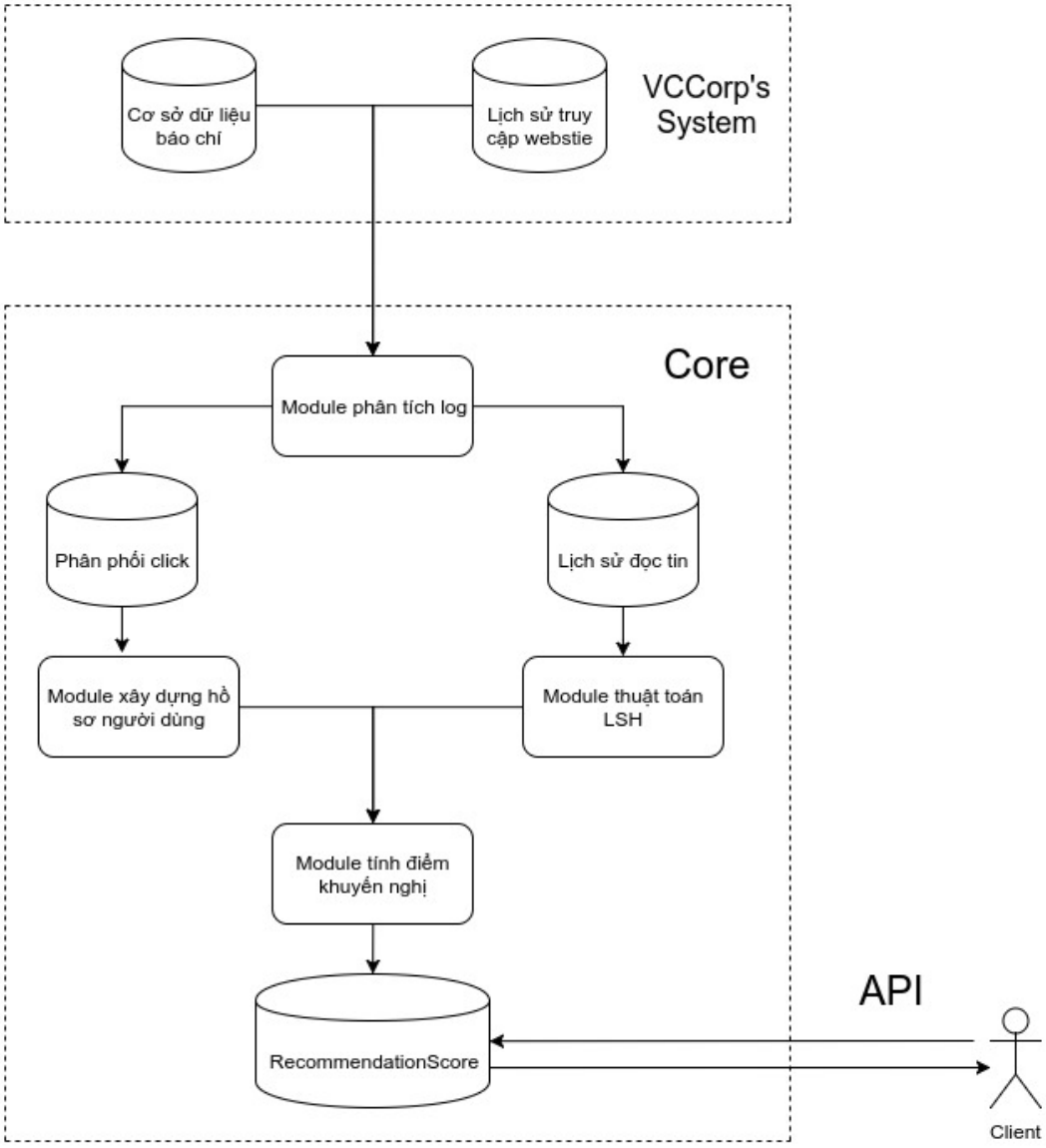
4.2 Yêu cầu hệ thống

Hệ thống xây dựng phải đáp ứng các yêu cầu sau:

- Phân tích lịch sử truy cập các trang tin tức của người dùng thường xuyên.
- Cập nhật hồ sơ người dùng thường xuyên. Như chúng ta đã biết, sở thích người dùng thay đổi theo thời gian nên việc cập nhật hồ sơ người dùng là cần thiết. Tuy nhiên, việc xây dựng hồ sơ cho vài chục triệu người dùng là một thách thức.
- Về mặt kỹ thuật, thời gian phản hồi cho một yêu cầu từ người dùng là $100 - 200ms/request$, khả năng chịu tải từ $10000 - 20000request/s$

4.3 Kiến trúc hệ thống

4.3.1 Kiến trúc hệ thống tổng quát



Hình 4.1: Kiến trúc hệ thống.

Hình trên trình bày tổng quan kiến trúc hệ thống. Ta sẽ lần lượt đi vào chi tiết từng module.

4.3.2 Module phân tích log

Module phân tích log: có nhiệm vụ phân tích, loại bỏ nhiễu trong lịch sử truy cập trang web của người dùng và chuẩn bị dữ liệu cho module xây dựng hồ sơ người dùng và module thuật toán LSH. Module sẽ lập lịch, định thời chạy phân tích và lưu kết quả phân tích dưới dạng tệp tin Parquet. (Parquet là một dạng định dạng lưu trữ tồn tại trong hệ sinh thái Hadoop. Theo một bài viết của IBM [1], chuyển đổi định dạng văn bản như csv sang parquet giúp tăng tốc độ truy xuất lên 30 lần (hoặc hơn) tùy trường hợp, bộ nhớ tiết kiệm đến 75%). Để hiểu rõ hơn, trước tiên, ta sẽ đi sơ lược về hệ thống ghi nhận lịch sử truy cập các trang web của công ty. Sau đó, định nghĩa đầu vào, đầu ra của module và nêu chi tiết về cách hoạt động của module cũng như công nghệ sử dụng.

Đầu tiên, nói về hệ thống ghi nhận lịch sử truy cập các trang web của công ty. Công ty có rất nhiều loại log phục vụ những mục đích khác nhau. Ở đây, ta quan tâm đến một loại gọi là LogPageView. Vậy, LogPageView là gì? LogPageView là loại log sẽ ghi lại thời điểm mà một người dùng click vào một đường dẫn nào đó bất kỳ. Nó cũng ghi nhận các thông tin khác đi kèm với người dùng. LogPageView được lưu dưới dạng tệp tin *.dat*, có 24 cột và mỗi cột cách nhau bởi ký tự tab. Ta không có mô tả đầy đủ cho 24 cột, chỉ có một số cột như sau:

STT	Chỉ số cột	Nội dung
1	0	ClickTime: thời điểm người dùng click vào một đường dẫn nào đó.
2	1	Cookies-Create: thời điểm cookie được tạo.
3	3	IP: địa chỉ IP của người dùng.
4	5	OS: Hệ điều hành người dùng sử dụng.
5	8	Domain: tên miền mà người dùng click vào.
6	13	UserID: id của người dùng.
7	16	Resolution: Độ phân giải màn hình của người dùng.
8	21	Full-Path: đường dẫn đầy đủ mà người dùng click vào.

Bảng 4.1: Mô tả các cột trong LogPageView.

Lưu ý, chỉ số cột tính từ 0. Thi thoảng, sẽ có những dòng có số cột nhỏ hơn 24 hoặc dữ liệu trong từng cột bị lỗi định dạng, nói chung là bị nhiễu. Đây là điều cần lưu ý

[1] https://developer.ibm.com/hadoop/2015/12/03/parquet-for-spark-sql/

với dữ liệu có được từ hệ thống log của công ty. Dữ liệu log một ngày tương đối lớn, khoảng từ 23GB/ngày.

Tiếp theo, ta sẽ đi mô tả về cơ sở dữ liệu báo chí của công ty. Ngoài các trang báo mà công ty đang quản lý và hợp tác, công ty còn viết thêm một công cụ nhằm lấy thêm từ các trang báo khác. Hệ thống đảm bảo rằng sẽ lấy được những bài viết mới nhất trong vòng 30s kể từ khi được xuất bản trên mạng. Cơ sở dữ liệu báo chí gồm những thông tin sau:

STT	Trường	Nội dung
1	id	Id của bài báo.
2	title	Tiêu đề của bài báo
3	content	Nội dung của bài báo
4	source	Nguồn của bài báo. Lưu ý rằng chỉ có domain mà thôi.
5	create_time	Thời điểm mà bài báo được đăng lên trang tin.
6	description	Phần mô tả về bài báo. Thường là sau tiêu đề và trước nội dung chính.
7	url	Dường dẫn đầy đủ của bài báo.
8	get_time	Thời điểm mà bài báo được thu thập về cơ sở dữ liệu
9	catid	Mã chuyên mục của bài báo.

Bảng 4.2: Mô tả các trường trong cơ sở dữ liệu báo chí.

Như đã trình bày ở trên, module phân tích log sẽ chuẩn bị dữ liệu đầu vào cho các module còn lại. Module xây dựng hồ sơ người dùng cần đầu vào là $D(u,t)$ (phân phối click vào mỗi chủ đề c_i của người dùng u trong khoảng thời gian t) và $D(t)$ (phân phối click vào chủ đề c_i được tạo ra bởi tất cả người dùng của một quốc gia/vùng lãnh thổ trong khoảng thời gian t). Module thuật toán LSH có đầu vào là characteristic matrix. Để cho tiện lợi, ta không tạo characteristic matrix ở giai đoạn phân tích log này, ta chỉ tạo bảng lịch sử click (click history) của người dùng. Ta lần lượt mô tả ba bảng $D(u,t), D(t)$, lịch sử click và hai bảng Category (chứa danh sách các chủ đề mà ta xét) và TimePeriod (chứa danh sách các khoảng thời gian).

STT	Trường	Nội dung
1	id	Id của một chủ đề
2	category	Tên của chủ đề.

Bảng 4.3: Mô tả các trường cho bảng Category.

STT	Trường	Nội dung
1	id	Id của một khoảng thời gian t.
2	startTime	Ngày bắt đầu khoảng thời gian.
3	endTime	Ngày kết thúc khoảng thời gian.

Bảng 4.4: Mô tả các trường cho bảng TimePeriod.

STT	Trường	Nội dung
1	userID	Id của một người dùng
2	categoryID	Id của một chủ đề.
3	allClick	Tổng số click được tạo ra bởi người dùng có id là userID trong khoảng thời gian có id là timePeriodID
4	clickDistribution	Phần trăm click vào chủ đề có id là categoryID được tạo ra bởi người dùng có id là userID trong khoảng thời gian có id là timePeriodID.
5	timePeriodID	Id của một khoảng thời gian

Bảng 4.5: Mô tả các trường cho bảng $D(u, t)$.

STT	Trường	Nội dung
1	categoryID	Id của một chủ đề.
2	allClick	Tổng số click được tạo ra trong khoảng thời gian có id là timePeriodID
3	clickDistribution	Phần trăm click vào chủ đề có id là categoryID trong khoảng thời gian có id là timePeriodID.
4	timePeriodID	Id của một khoảng thời gian

Bảng 4.6: Mô tả các trường cho bảng $D(t)$.

STT	Trường	Nội dung
1	userID	Id của người dùng.
2	newsID	Id của bài báo.
3	catID	Chủ đề bài báo thuộc về.
4	clickTime	Thời điểm người dùng click vào bài báo.

Bảng 4.7: Mô tả các trường cho bảng lịch sử click.

Như vậy, ta xác định đầu vào, đầu ra của module phân tích log như sau:

- **Đầu vào:**
  - Kho dữ liệu tin tức, ký hiệu $newsDB$.

– Lịch sử truy cập các trang tin tức, ký hiệu $logPageView$

- **Đầu ra:**

  – $D(u,t), D(t)$, lịch sử click của người dùng.

Ban đầu, em lưu tất cả các bảng này vào Cassandra. Tuy nhiên, khi bộ dữ liệu lớn dần lên, việc đọc và ghi bằng Cassandra đã không phù hợp. Lý do vì thiết kế của Cassandra không phù hợp cho việc lấy một lượng lớn dòng một lúc, vậy nên em quyết định lưu $D(u,t), D(t)$ và lịch sử click dưới dạng tệp tin Parquet. Sau khi chuyển sang dùng Parquet, hệ thống không những ổn định hơn mà thời gian thực thi module phân tích log cũng nhanh hơn. Sau đây là bảng so sánh mà em đã ghi nhận lại.

STT	Ngày phân tích log	Cassandra	Parquet
1	20-09-2917	5mins, 13sec	3mins, 8sec
2	21-09-2017	10mins, 24sec	3mins, 20sec
3	22-09-2017	13mins, 24sec	3mins, 13sec
4	23-09-2017	14mins, 10sec	2mins, 54sec
5	24-09-2917	14mins, 17sec	3mins, 0sec
6	25-09-2017	16mins, 16sec	3mins, 14sec
7	26-09-2017	17mins, 46sec	3mins, 22sec
8	27-09-2017	18mins, 59sec	3mins, 21sec
9	28-09-2017	20mins, 36sec	3mins, 47sec
10	29-09-2017	21mins, 05sec	3mins, 36sec
11	30-09-2017	21mins, 16sec	3mins, 22sec
12	01-10-2017	23mins, 33sec	3mins, 15sec
13	02-10-2017	25mins, 26sec	3mins, 54sec
14	03-10-2017	25mins, 42sec	3mins, 20sec
15	04-10-2017	28mins, 28sec	3mins, 42sec
16	05-10-2017	5mins, 10sec	3mins, 9sec
17	06-10-2017	10mins, 34sec	2mins, 48sec
18	07-10-2017	12mins, 49sec	2mins, 49sec
19	08-10-2017	12mins, 50sec	2mins, 37sec
20	09-10-2017	16mins, 1sec	2mins, 54sec
21	10-10-2017	17mins, 18sec	3mins, 18sec
22	11-10-2017	19mins, 10sec	3mins, 26sec

Bảng 4.8: Bảng so sánh thời gian thực thi module phân tích log khi lưu trữ bằng Cassandra và Parquet

Module phân tích log sẽ lập lịch, chạy định kỳ 1 lần/ngày.

4.3.3 Module xây dựng hồ sơ người dùng.

Module này sẽ tiến hành xây dựng hồ sơ người dùng dựa vào phần lý thuyết đã được trình bày tại mục 3.5. Ta xác định đầu vào và đầu ra của module như sau:

- **Đầu vào:** $D(u,t), D(t)$
- **Đầu ra:** Sở thích hiện tại của người dùng.

Sở thích hiện tại của người dùng $p^0(category = c_i|click)$ chính là xác suất mà người dùng click vào một chủ đề khi cho trước click. Sở thích hiện tại người dùng được lưu vào bảng UserCurrentInterest và có mô tả như sau:

STT	Trường	Nội dung
1	userID	Id của một người dùng.
2	catID	Id của một chủ đề.
3	score	Điểm khuyến nghị, thể hiện mức độ yêu thích của người dùng có id là userID với chủ đề có id là catID

Bảng 4.9: Mô tả các trường cho bảng UserCurrentInterest

Module này sẽ lập lịch, định thời chạy xây dựng hồ sơ người dùng và lưu kết quả vào cơ sở dữ liệu Casandra. Nhắc lại rằng, sở thích dài hạn của người người dùng sẽ được tính trong vòng một năm, xu hướng tin hiện tại được lấy trong ngày hôm trước (ngày trước khi chạy thuật toán xây dựng hồ sơ người dùng) và mỗi khoảng thời gian t có độ dài 15 ngày.

4.3.4 Module thuật toán LSH.

Module này sẽ tiến hành gom cụm người dùng, những người dùng nào tương tự nhau sẽ được gom vào chung cụm bằng thuật toán LSH dựa vào phần lý thuyết được trình bày tại mục 3.6.

- **Đầu vào:** Lịch sử click của người dùng
- **Đầu ra:** Thông tin cụm, thông tin về người dùng.

Ở đây, đầu vào của module hơi khác so với đầu vào của thuật toán LSH được mô tả ở mục 3.6, chúng ta phải có một bước chuyển từ lịch sử click của người dùng về

characteristic matrix. Thông tin cụm được lưu vào bảng ClusterTable và thông tin về người dùng được lưu vào bảng UserTable. Mô tả hai bảng này như sau:

STT	Trường	Nội dung
1	clusterID	Id của cụm.
2	listUserID	Danh sách người dùng thuộc vào cụm.

Bảng 4.10: Mô tả các trường cho bảng ClusterTable

STT	Trường	Nội dung
1	userID	Id của người dùng.
2	listClusterID	Danh sách cụm mà người dùng thuộc về.
3	listNewsID	Danh sách bài báo mà người dùng đã click.

Bảng 4.11: Mô tả các trường cho bảng UserTable

Module này sẽ lập lịch, chạy định thời và lưu kết quả dưới dạng tệp tin Parquet. .

4.3.5 Module tính điểm khuyến nghị.

Module này thực hiện tính điểm khuyến nghị, lưu kết quả sau cùng vào cơ sở dữ liệu Cassandra nhằm phục vụ cho việc truy vấn TopN bài báo phù hợp nhất với người dùng. Lý thuyết xem tại mục 3.7. Ta định nghĩa đầu vào và đầu ra của module như sau:

- **Đầu vào:** Thông tin cụm (bảng ClusterTable), thông tin người dùng (bảng UserTable) và sở thích hiện tại của người dùng (bảng UserCurrentInterest)
- **Đầu ra:** Thông tin khuyến nghị.

Sau khi có được cụm người dùng (chạy module thuật toán LSH), ta sẽ dùng giải thuật 3 để tính điểm khuyến nghị cho cụm người dùng. Kết quả lưu vào bảng LSHRecommendationScore. Bảng này được mô tả như sau:

STT	Trường	Nội dung
1	userID	Id của người dùng.
2	newsID	Id của bài báo.
3	score	Điểm khuyến nghị, thể hiện mức độ yêu thích của người dùng có id là userID với bài báo có id là newsID

Bảng 4.12: Mô tả các trường cho bảng LSHRecommendationScore

Sau đó, ta sẽ kết hợp dữ liệu sao khi xây dựng hồ sơ người dùng (bảng UserCurrentInterest) và điểm khuyến nghị sau khi gom cụm (bảng LSHRecommendationScore) lại với nhau, bằng công thức 3.10: $Rec(article) = BuildProfile(article) * LSH(article)$. Sau đó, ta lưu kết quả khuyến nghị cuối cùng vào bảng RecommendationScore. Bảng này được mô tả như sau:

STT	Trường	Nội dung
1	userID	Id của người dùng.
2	newsID	Id của bài báo.
3	score	Điểm khuyến nghị, thể hiện mức độ yêu thích của người dùng có id là userID với bài báo có id là newsID

Bảng 4.13: Mô tả các trường cho bảng RecommendationScore

Ở module này, ta không lập lịch, định thời như các module trước. Vì lượng người dùng lớn và bản thân người làm luận văn chưa thể tối ưu hóa việc tính điểm này (Tính điểm cho một người dùng mất trung bình khoảng 2 phút, với cấu hình 37GB ram, 18 core CPU) nên ta sẽ không tính điểm cho tất cả người dùng, ta sẽ chỉ tính điểm cho những người dùng nào thực sự tương tác với hệ thống mà thôi. Để làm được việc này, ta cần thêm một bảng phụ lưu thời điểm mà một người dùng được tính điểm khuyến nghị. Ta đặt tên bảng này là TrackingRecommendationScore và có mô tả như sau:

STT	Trường	Nội dung
1	userID	Id của người dùng.
2	updateTime	Thời điểm gần nhất mà người dùng được tính điểm khuyến nghị.

Bảng 4.14: Mô tả các trường cho bảng TrackingRecommendationScore

Nếu một người dùng chưa được tính điểm khuyến nghị (chưa có thông tin trong bảng RecommendationScore) hoặc lần sau cùng được tính điểm khuyến nghị lâu hơn một khoảng thời gian cho trước (ví dụ như một ngày), ta sẽ thực hiện tính điểm khuyến nghị cho người dùng đó. Trường hợp nếu đã có thông tin rồi, thì ta sẽ xuất ra thông tin đã có, nếu chưa có thông tin, ta sẽ xuất ra những bài báo đang được xem nhiều nhất.

4.4 Thiết kế API

Vì đây là hệ thống chạy ngầm bên dưới nên sẽ không có giao diện hiển thị, chỉ có API tương tác với phía bên ngoài mà thôi. Hệ thống chỉ có một API với định dạng đầu vào và đầu ra như sau:

- **Đầu vào:**
  - guid: id của người dùng cần được khuyến nghị
- **Đầu ra:**

```
{
 "algid": integer,
 "recommend":  [
      {"id": long},
      {"id": long },
      ...
      {"id": long}
   ]
}
```

Trong đó,

- algid: là mã định danh của thuật toán mà em xây dựng.
- recommend: danh sách tin được khuyến nghị cho người dùng.
- recommend.id: là mã định danh của một bài báo (newsID)

API này sẽ được gắn vào một hệ thống quản lý thuật toán của công ty. Hệ thống này sẽ tự động lấy tiêu đề, mô tả (description/abstract) của bài báo dựa vào newsID để trả về cho người dùng. Hệ thống quản lý thuật toán cũng sẽ có nhiệm vụ đo kiểm tính hiệu quả của các thuật toán.

4.5 Tổng quan công nghệ.

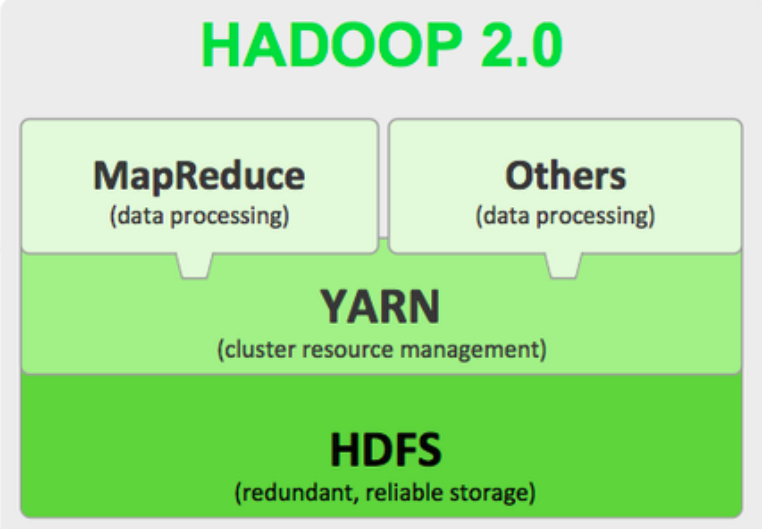
Phần này, ta sẽ nói tổng quan về các công nghệ chính được sử dụng trong khóa luận, các công nghệ cho việc lưu trữ và xử lý dữ liệu lớn. Ta sẽ tập trung vào ba công nghệ chính: Apache Hadoop, Apache Spark và Apache Cassandra.

4.5.1 Giới thiệu Apache Hadoop.

Tổng quan Apache Hadoop.

Apache Hadoop là một framework cho phép xử lý phân tán tập dữ liệu lớn bằng các cụm máy tính [1]. Nó cho phép các ứng dụng làm việc với hàng ngàn máy tính tính toán độc lập và petebyte dữ liệu. Hadoop được bắt nguồn từ bài viết *"The Google File System"* được xuất bản vào tháng 10 năm 2003 của ba tác giả *Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung*. Sau đó, được Doug Cutting phát triển và phiên bản Hadoop 0.1.0 được phát hành vào tháng 4 năm 2006. Tính tới thời điểm 24/12/2017, phiên bản 3.0.0 là bản Hadoop mới nhất.

Kiến trúc Apache Hadoop.



Hình 4.2: Kiến trúc Apache Hadoop

Kiến trúc Hadoop gồm 3 thành phần chính: [10]

- HDFS (Hadoop Distributed Filesystem): Đây là tầng lưu trữ của Hadoop.
- YARN (Yet Another Resource Negotiator): Đây là tầng quản lý tài nguyên (resource management) của Hadoop.
- MapReduce: Đây là tầng xử lý dữ liệu của Hadoop.

Ta sẽ tìm hiểu kỹ hơn về ba thành phần này của Hadoop.

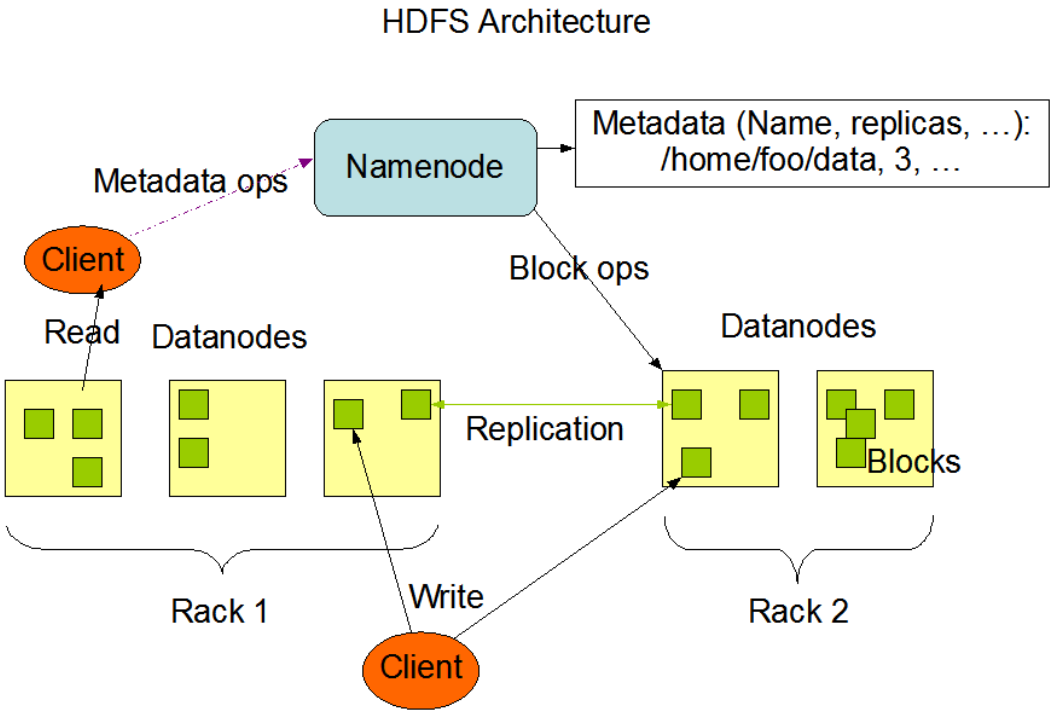
[1] http://hadoop.apache.org/

HDFS

Khi dữ liệu lớn lên, vượt ngoài khả năng lưu trữ của một máy tính đơn lẻ thì nhu cầu chia tách bộ dữ liệu lớn đó thành các phần nhỏ (gọi là partition) và lưu các phần nhỏ đó ở nhiều máy khác nhau là cần thiết. Hệ thống tập tin *(filesystem)* quản lý việc lưu trữ dữ liệu trên nhiều máy gọi là hệ thống tập tin phân tán *(distributed filesystem)*. Hệ thống tập tin phân tán phức tạp hơn hệ thống quản lý file thông thường rất nhiều. HDFS là hệ thống tập tin phân tán của Hadoop, có tính chịu lỗi cao và được thiết kế để chạy trên phần cứng thông thường [10].

Đặc điểm:

- Nó phù hợp cho việc lưu trữ và xử lý phân tán.
- Hadoop cung cấp một giao diện lệnh để tương tác với HDFS.
- Việc xây dựng trong máy chủ của namenode và datanode giúp người dùng dễ dàng kiểm tra tình trạng của cụm máy tính.
- HDFS cung cấp quyền truy cập tập tin và xác thực.



Hình 4.3: Kiến trúc HDFS.
(Nguồn: https://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html)

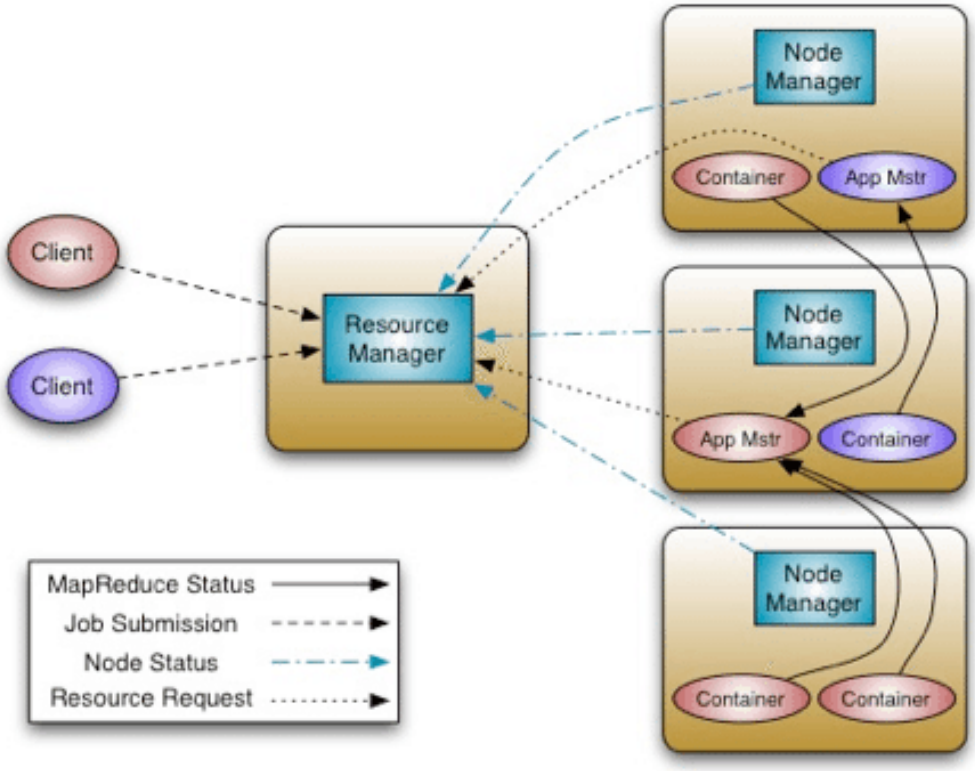
HDFS có kiến trúc master / slave. Một cụm HDFS chỉ có một Namenode (master) duy nhất. Namenode quản lý không gian tên hệ thống tập tin và điều chỉnh quyền truy cập vào các tệp tin của người dùng (client). Namenode thực thi các thao tác như mở, đóng, đổi tên tệp tin và thư mục. Nó cũng xác định vị trí các block dữ liệu trên Datanode (slave). Một cụm HDFS có nhiều Datanodes. Datanodes có trách nhiệm phục vụ các yêu cầu đọc và viết từ người dùng. Datanodes cũng thực hiện các hoạt động như tạo block, xóa và nhân bản theo hướng dẫn của Nnamenode.

HDFS được thiết kế để hỗ trợ các tập tin rất lớn. HDFS thích hợp với các ứng dụng ghi một lần và đọc nhiều lần với tốc độ chấp nhận được. Một tập tin trên HDFS được cắt nhỏ thành nhiều block với kích thước cố định. Mỗi block có thể nằm trên một Datanode khác nhau. Với Apache Hadoop 2.x.x, kích thước mặc định của một block là 128MB. Mỗi block này sẽ được nhân bản dựa vào hệ số nhân bản được thiết lập và mỗi bản sao của block sẽ được lưu trữ ở những Datanode khác nhau. Đây là tính chất chịu lỗi của Hadoop.

YARN

YARN - Yet Another Resource Negotiator là hệ thống quản lý tài nguyên của cụm Hadoop. YARN được giới thiệu trong Hadoop 2 để cải thiện quá trình thực thi MapReduce. YARN cũng hỗ trợ cho các mô hình tính toán phân tán khác như Apache Spark. [10]

Ý tưởng chính của YARN là chia tách chức năng quản lý tài nguyên và định thời tác vụ (job scheduling/monitoring) vào những tiến trình riêng biệt. Ta sẽ có một thành phần quản lý toàn bộ tài nguyên tính toán của cụm, gọi là Resource Manager (RM) và một thành phần quản lý vòng đời của ứng dụng gọi là Application Master (AM). Mỗi ứng dụng sẽ có riêng một Application Master và nhiều Container. Trên mỗi máy trong cụm sẽ có một thành phần gọi là NodeManger. NodeManager có nhiệm vụ theo dõi tài nguyên (cpu, bộ nhớ, tình trạng đĩa cứng, mạng,...) và sẽ báo cáo kết quả về cho Resource Manager.



Hình 4.4: Kiến trúc YARN.
(Nguồn: https://hadoop.apache.org/docs/r2.7.1/hadoop-yarn/hadoop-yarn-site/YARN.html)

Quá trình một ứng dụng chạy trên YARN được mô tả bằng sơ đồ trên. Cụ thể các bước khởi động một ứng dụng như sau:

- Client giao một task cho Resource Manager
- Resource Manager tính toán tài nguyên cần thiết theo yêu cầu của ứng dụng và tạo một Application Master. Application Master được chuyển đến chạy trên một node tính toán. Application Master sẽ liên lạc với các NodeManager ở các node khác để ra yêu cầu công việc cho node này.
- NodeManager nhận yêu cầu và chạy các task trên container
- Các thông tin trạng thái thay sẽ được gửi đến Application Master.

Ở YARN, tài nguyên không đơn thuần là CPU cores nữa mà bao gồm cả bộ nhớ, DiskIO, và GPUs... Một ứng dụng khi khởi động sẽ yêu cầu:

- Priority: ưu tiên xem ứng dụng nào sẽ được chạy.
- Bộ nhớ (MB): chương trình ứng dụng dự định sẽ sử dụng bao nhiêu (MB) bộ nhớ
- CPU: ứng dụng dự định sử dụng bao nhiêu cores.
- Số lượng containers. Containers liên quan đến số lượng tasks có thể được triển khai song song trên một node tính toán. Giới hạn tài nguyên sử dụng bởi một containers được thực hiện thông qua tính năng cgroups của Linux kernels.

MapReduce

MapReduce là một framework cho phép dễ dàng viết các ứng dụng mà cần xử lý một lượng lớn dữ liệu (vài terabyte) một cách song song trên cụm máy lớn (vài ngàn máy) một cách tin cậy, có khả năng chịu lỗi. [1]

MapReduce hoạt động bằng cách chia quá trình xử lý thành hai giai đoạn: giao đoạn map và giai đoạn reduce. Mỗi giai đoạn đề có đầu vào và đầu ra là một cặp key-value. Kiểu của cặp key-value này được xác định bởi lập trình viên. Đúng như tên

[1] https://hadoop.apache.org/docs/r2.7.3/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

gọi, giai đoạn reduce diễn ra sau khi giai đoạn map được thực thi xong và dữ liệu đầu vào của giai đoạn reduce chính là dữ liệu đầu ra của giai đoạn map.

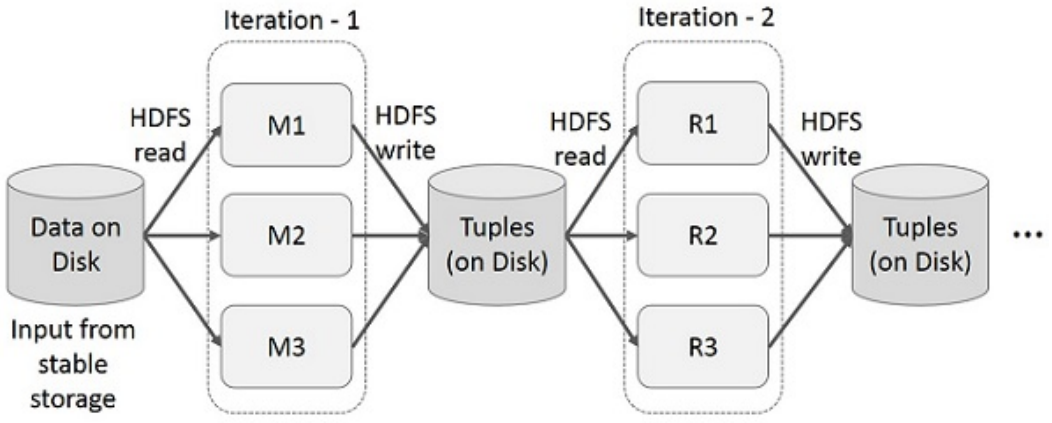
Hàm map hoạt động như giai đoạn chuẩn bị dữ liệu, thiết lập dữ liệu theo cách mà hàm reduce có thể sử dụng. Đây cũng là nơi tốt cho việc loại đi những dữ liệu bị nhiễu, xấu. Hàm reduce sẽ nhận các cặp key-value từ nhiều tác vụ map khác nhau. Hàm reduce sẽ tập kết những giá trị này và cho ra kết quả sau cùng. Hàm reduce không thể bắt đầu nếu tất cả các tác vụ map chưa được hoàn thành.

Việc của lập trình viên là quan tâm tới 2 hàm map và reduce. Còn các vấn đề khác như : phân chia các dữ liệu đầu vào, lịch trình thực thi trên các máy, xử lý nếu có máy bị chết, quản lý việc giao tiếp giữa các máy là việc của framework.

4.5.2 Giới thiệu Apache Spark.

Tổng quan Apache Spark.

Tuy có nhiều ưu điểm, song Hadoop gặp phải một hạn chế chính, đó là sau mỗi lần chạy MapReduce thì kết quả phải được lưu lại trên HDFS. Nếu chương trình cần chạy nhiều tác vụ MapReduce thì quá trình đọc và ghi dữ liệu trên HDFS rất tốn thời gian và chi phí, ta không tận dụng được các kết quả trên trung gian cũng như chưa thể lưu các kết quả này trên RAM để tăng tốc thời gian xử lý. Do đó, Apache Spark ra đời để giải quyết hạn chế này.



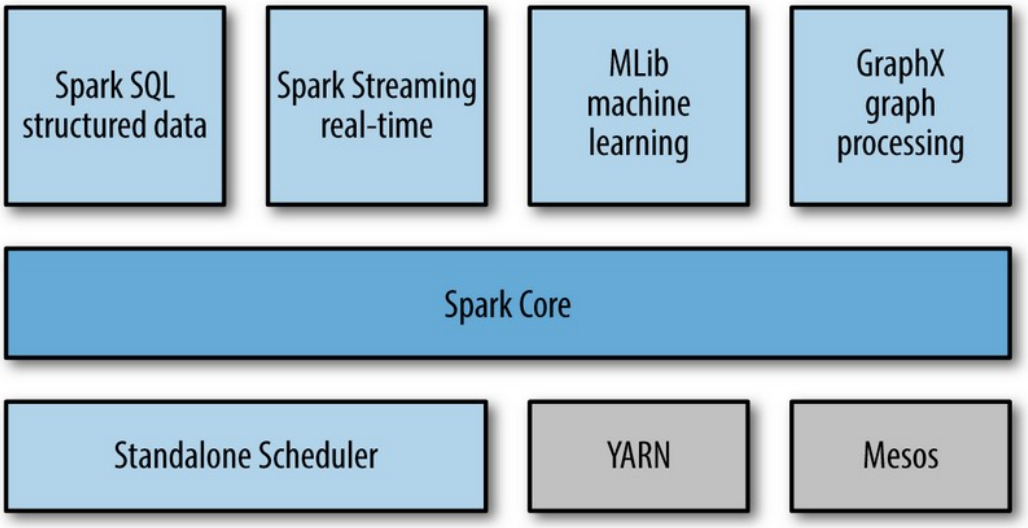
Hình 4.5: Sự hạn chế của Hadoop.
(Nguồn: https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm)

Apache Spark là một nền tảng tính toán trên cụm máy tính được thiết kế với mục

đích tính nhanh và thông dụng [11]. Điểm chính của Spark là nó tính toán hoàn toàn trên RAM, do đó, chạy nhanh hơn MapReduce rất nhiều, từ 10 - 100 lần [1].

Dự án về Spark được khởi sự từ năm 2009 tại phòng phí nghiệm RAD tại trường đại học UC Berkeley. Sau này , Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay. Phiên bản mới nhất hiện tại là 2.2.0.

Thành phần của Apache Spark.



Hình 4.6: Thành phần của Spark.
(Nguồn: Trích trong sách *Learning Spark: Lightning-Fast Big Data Analysis*)

Spark có thể chạy trên nhiều loại Cluster Manager khác nhau như Hadoop YARN, Apache Mesos hoặc trên chính cluster manager được cung cấp bởi Spark được gọi là Standalone Scheduler.

Thành phần trung tâm của Spark là Spark Core. Spark Core cung cấp những chức năng cơ bản nhất của Spark như lập lịch cho các tác vụ, quản lý bộ nhớ, fault recovery, tương tác với các hệ thống lưu trữ.... Đặc biệt, Spark Core cung cấp API để định nghĩa RDD (Resilient Distributed DataSet), là cấu trúc dữ liệu nền tảng của Spark. RDD đại diện cho một tập các đối tượng được phân tán trên các máy của cụm và có thể được xử lý song song.

Ngoài ra, còn có các thành phần khác được xây dựng trên Spark Core như sau:

[1]https://spark.apache.org/

- Spark SQL là một gói cho phép xử lý dữ liệu có cấu trúc. Nó cho phép truy vấn dữ liệu thông qua các câu lệnh SQL. Spark SQL có thể thao tác với nhiều nguồn dữ liệu như Hive tables, Parquet, và JSON. Nó còn cung cấp một công cụ tối ưu hóa có tên Catalyst optimizer, cho phép tối ưu các đoạn code mà người dùng viết một cách tự động.
- Spark Streaming cung cấp API để dễ dàng xử lý dữ liệu stream.
- MLlib Cung cấp rất nhiều thuật toán của học máy như: classification, regression, clustering, collaborative filtering,...
- GraphX là thư viện để xử lý đồ thị.

Tại sao lại sử dụng Apache Spark.

Việc sử dụng Spark trở nên ngày càng phổ biến, thể hiện qua việc có nhiều công ty lớn đã sử dụng Spark trong các sản phẩm của mình. Có thể kể đến Amazon, Baidu, eBay Inc, NTT Data, Yandex,... [1]. Ta sẽ liệt kê một số lý do tại sao nên sử dụng Apache Spark.

- Khả năng tích hợp với Hadoop. Điều này rất quan trọng vì những hệ thống trước đó nếu đã phát triển bằng Hadoop thì nay khi chuyển sang dùng Spark, ta vẫn có thể tận dụng lại những thành phần cũ. Về cơ bản, ta có thể coi Spark như là phần thay thế cho MapReduce.
- Nhanh. Ý tưởng ban đầu của Spark cũng giống như ý tưởng của MapReduce ngoại trừ việc dữ liệu được lưu trữ trên RAM. Điều này khiến cho chương trình trình chạy nhanh hơn vì dữ liệu được truy suất trên RAM.
- Xử lý dữ liệu stream. Thành phần Spark Streaming giúp ta phân tích, xử lý dữ liệu theo thời gian thực khi dữ liệu được thu thập.
- Code viết bằng Spark rõ ràng, dễ hiểu và nhanh hơn so với MapReduce.
- Hỗ trợ nhiều ngôn ngữ như Java, Scala, Python, R.
- Cộng đồng lớn [2].

[1] https://spark.apache.org/powered-by.html
[2] https://data-flair.training/blogs/apache-spark-vs-hadoop-mapreduce/

Nhược điểm lớn nhất của Spark là sử dụng nhiều tài nguyên, cụ thể là RAM so với MapReduce.

Sau đây là bảng so sánh thời gian thực thi module xây dựng hồ sơ người dùng khi không sử dụng và sử dụng Spark.

STT	Dataset	Java multithread	Spark local	Spark cluster
1	20 file log đầu tiên ngày 10.10.2017	3h23p57s	23s	38s
2	50 file log đầu tiên ngày 10.10.2017	13h14p56s	35s	58s
3	100 file log đầu tiên ngày 10.10.2017	1day3h30p16s	58s	1p16s

Bảng 4.15: Bảng so sánh thời gian chạy module xây dựng hồ sơ người dùng khi sử dụng và không sử dụng Spark.

Trong đó, cấu hình để chạy Java multithread, Spark local và Spark cluster như sau:

- Java multithread: 8G RAM, 8 CPU cores.
- Spark local: 8G RAM, 8 CPU cores.
- Spark cluster: 26GB RAM, 16 CPU cores.

Nhìn vào bảng trên, ta thấy rằng việc xây dựng hồ sơ người dùng 100 file log đầu tiên ngày 10.10.2017 khi dùng Java Multithread tốn 1 ngày 3 giờ 30 phút 16 giây, trong khi nếu dùng Spark chế độ local tốn 58 giây và Spark chế độ cluster tốn 1 phút 16 giây. Vậy, ta quyết định sử dụng Spark thay vì sử dụng Java Multithread.

4.5.3 Giới thiệu Apache Cassandra.

Tổng quan về Apache Cassandra.

Cassandra là một quản trị hệ cơ sở dữ liệu phân tán mã nguồn mở được thiết kế để xử lý một khối lượng lớn dữ liệu dàn trải trên nhiều node mà vẫn đảm bảo tính sẵn sàng cao (Highly Availability), khả năng mở rộng hay thu giảm số node linh hoạt (Elastic Scalability) và chấp nhận một số lỗi (Fault Tolerant). Nó được phát triển bởi Facebook và vẫn còn tiếp tục phát triển và sử dụng cho mạng xã hội lớn nhất thới giới này. Năm 2008, Facebook chuyển nó cho cộng đồng mã nguồn mở và được Apache

tiếp tục phát triển đến ngày hôm nay. Cassandra được coi là sự kết hợp của Amazon's Dynamo và Google's BigTable.

Đặc điểm của Cassandra.

Ta liệt kê một số đặc điểm nổi bật của Cassandra như sau:

- Tính phân tán (Distributed): Đặc tính phân tán gần như là đặc tính chung của NoSQL và vì thế nó cũng là một đặc tính quan trọng của Casandra. Dữ liệu có thể được lưu trữ ở nhiều nơi, trên các máy khác nhau hoặc trên các dải mạng khác nhau nhằm đảm bảo khi một máy bị sự cố thì vẫn có thể thao tác được với dữ liệu trên các máy khác. Ngoài ra thì việc tính toán phân tán dữ liệu theo các thuật toán riêng cũng đảm bảo cho người dùng có thể truy cập dữ liệu một cách nhanh nhất (ví dụ 3 bản sao của dữ liệu lưu ở ba châu lục thì IP thuộc châu lục nào sẽ ưu tiên truy cập bản dữ liệu tại châu lục đó, khi đó sẽ nhanh hơn truy cập dữ liệu ở server châu lục khác).

- Kiến trúc ngang hàng (peer to peer) :Các node trong Cassandra có vai trò ngang hàng với nhau. Không giống các hệ thống khác, Cassandra mang kiến trúc peer to peer, do vậy có thể dễ dàng mở rộng cụm bằng cách thêm các node mới vào. Các node trong cluster có nhiệm vụ như nhau, không phân biệt master hay slave. Các node đều đảm nhận việc đọc và ghi dữ liệu. Cassandra được đánh giá nổi bật là dựa trên yếu tố này. Xây dựng dựa trên kiến trúc ngang hàng sẽ làm giảm nguy cơ bị tình trạng thắt cổ chai (đối với kiến trúc master - slave việc ghi dữ liệu chỉ do primary node đảm nhận còn các slave - node chỉ nhận các request đọc dữ liệu, vì nếu nếu có quá nhiều yêu cầu ghi cùng một lúc sẽ gây quá tải cho primary node, dẫn đến tình trạng ta gọi là bị thắt cổ chai). Đồng thời với cơ chế peer to peer cũng cần ít resource hơn rất nhiều so với kiến trúc master - slave.

- Tính co dãn (Elastic Scalability): Hệ thống có thể dễ dàng mở rộng số node trong cluster để có thể phục vụ số lượng request lớn và rút bớt số node khi số lượng request giảm. Hệ thống khả năng đọc/ghi tăng tuyến tính theo số lượng máy được thêm vào cụm máy của bạn mà không có thời gian chết (downtime) hay sự gián đoạn ứng dụng đang chạy của bạn.

- Tính sẵn sàng cao (High Availability): Dữ liệu được sao lưu thành nhiều bản và được lưu trữ ở nhiều node. Điều này mang lại khả năng đáp ứng ngay lập tức cho Cassandra khi người dùng thực hiện tác vụ đọc hay ghi bằng cách thực hiện trên bản sao gần nhất hoặc trên tất cả các bản sao (phụ thuộc vào thông số Consistency Level do người dùng thiết lập).

- Tính chấp nhận lỗi (Fault Tolerance): Do dữ liệu được sao chép thành nhiều bản trên các node của cụm nên kể cả khi dữ liệu ở một node nào đó bị lỗi, bạn vẫn có thể truy xuất dữ liệu của mình trên một node khác.

- Tính hướng cột (Column Oriented): Các RDBMS hướng dòng (row-oriented) phải định nghĩa trước các cột (column) trong các bảng (table). Đối với Cassandra các bạn không phải làm điều đó, đơn giản là thêm vào bao nhiêu cột cũng được tùy theo nhu cầu của bạn

- Hiệu năng cao (High performance): Cassandra được thiết kế riêng biệt từ sơ khai cho đến khi đầy đủ lợi ích cho máy đa luồng/đa lõi và được chạy trên hàng chục những máy được đặt trong các trung tâm dữ liệu với quy mô nhất quán và liên tục với hàng trăm terabyte dữ liệu. Cassandra đã được chứng mình là hoạt động đặc biệt tốt với yêu cầu tải nặng. Nó luôn có thể hiện thị rất nhanh chóng để ghi mỗi giây trên một máy trạm cơ bản. Khi bạn bổ sung thêm các máy chủ, bạn có thể duy trì tất cả các tính chất mong muốn mà hiệu suất không hề bị giảm sút.

Lý do sử dụng Cassandra.

Ngoài những ưu điểm đã được nêu ở phần "Đặc điểm của Apache Cassandra" ở phía trên thì có hai lý do chính khiến em quan tâm đến việc sử dụng Cassandra, đó là:

- Khi sử dụng Cassandra với Spark, Cassandra hỗ trợ việc tự động chia dữ liệu thành các partition nhỏ để cho Spark xử lý. Ban đầu, em sử dung hệ quản trị cơ sở dữ liệu MySQL. MySQL không hỗ trợ việc chia tách dữ liệu thành các partition nhỏ mà dữ liệu đều tập trung hết vào 1 partition. Khi sử dụng MySQL chung với Spark, sẽ dẫn đến việc bị lỗi không đủ bộ nhớ bên các datanode.

- Cassandra hỗ trợ rất tốt yêu cầu truy suất nhanh một hoặc một vài bản ghi bằng id.

4.6 Triển khai hệ thống.

Môi trường triển khai:

- Ngôn ngữ phát triển: Java 8.
- Máy chủ Linux.
- Hệ quản trị cơ sở dữ liệu: Apache Cassandra
- Framework hỗ trợ: Apache Hadoop, Apache Spark, Spring Boot.

Thông tin về máy chủ/cụm máy được mô tả ở bảng sau:

STT	Máy chủ/Cụm máy	Thông tin
1	Cụm Cassandra	48 GB RAM 24 cores CPU
2	Cụm Hadoop - Spark	56 GB RAM 32 cores CPU
3	Webapp	32GB RAM 16 cores CPU

Bảng 4.16: Thông tin máy chủ/cụm máy chủ triển khai hệ thống.

Dữ liệu cần xử lý khi hệ thống chạy thật:

- Dữ liệu báo chí: 2.6 GB (chỉ lấy newsID, url, categoryID).
- Dữ liệu lịch sử truy cập website: 8.4 TB.

4.7 Kết chương

Chương này đã trình bày chi tiết về kiến trúc hệ thống cũng như chi tiết việc cài đặt bên trong và cách triển khai hệ thống lên máy chủ.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết quả đạt được

Trong khóa luận này, em hiện thực lại một phần hệ thống Google News dựa trên hai bài báo khoa học được công bố: "Personalized news recommendation based on click behavior" [2] và "Google news personalization: scalable online collaborative filtering" [1]. Sau quá trình thực hiện khóa luận, em đạt được những kết quả sau:

- Về kiến thức:
  - Kiến thức cơ bản về hệ khuyến nghị.
  - Kiến thức về vấn đề tính toán phân tán, song song.
  - Kiến thức về thuật toán gom cụm Local Sensitive Hashing, hay gọi tắt là LSH.
  - Kiến thức liên quan đến xác suất Bayesian.
- Về kinh nghiệm:
  - Kinh nghiệm trong việc cài đặt, triển khai một cụm máy tính bằng Apache Hadoop, cài đặt hệ cơ sở dữ liệu phân tán với Apache Cassandra.
  - Kinh nghiệm trong việc tính toán phân toán với Apache Spark như việc xử lý vấn đề chia tách dữ liệu trên từng máy và tối ưu hóa chương trình, giảm thiểu tối đa việc truyền dữ liệu giữa các máy.
  - Kỹ năng đọc bài báo khoa học.

  - Kinh nghiệm trong việc thao tác với máy chủ Linux.

- Về hệ thống:

  - Hệ thống khuyến nghị tin tức phù hợp sở thích người đọc đã đi vào hoạt động, đang được triển khai thử nghiệm tại trang CafeBiz.

Hướng phát triển

Ở bước tiếp theo trong tương lai, ta có thể kể đến một số vấn đề sau:

- Hoàn thiện hệ thống bằng cách cài đặt hai thuật toán còn lại của hệ thống Google News là thuật toán PLSI và Covisitation [1].

- Tối ưu hóa việc tính điểm khuyến nghị.

- Do cơ sở dữ liệu tin tức được thu thập từ nhiều nguồn khác nhau, ta có thể tiến hành loại bỏ những bài viết trùng nhau trước khi thực hiện module phân tích log.

TÀI LIỆU THAM KHẢO

[1] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM, 2007. 1, 31, 56, 57

[2] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40. ACM, 2010. 1, 19, 20, 21, 56

[3] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B Kantor. *Recommender systems handbook*. Springer, 2015. 6

[4] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005. 7, 8, 10, 11, 12, 13

[5] Huỳnh Ngọc Tín. *Phát triển một số phương pháp khuyến nghị hỗ trợ tìm kiếm thông tin học thuật dựa trên tiếp cận phân tích mạng xã hội*. PhD thesis, 2014. 8, 9, 12, 14, 15

[6] FO Isinkaye, YO Folajimi, and BA Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273, 2015. 9

[7] Daniel Billsus and Michael J Pazzani. A hybrid user model for news story classification. In *UM99 User Modeling*, pages 99–108. Springer, 1999. 20

[8] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998. 25

[9] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge university press, 2014. 27

[10] Tom White. *Hadoop: The definitive guide*. O'Reilly Media, Inc., 2012. 44, 45, 47

[11] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia. *Learning Spark: Lightning-Fast Big Data Analysis*. O'Reilly Media, 2015. 50

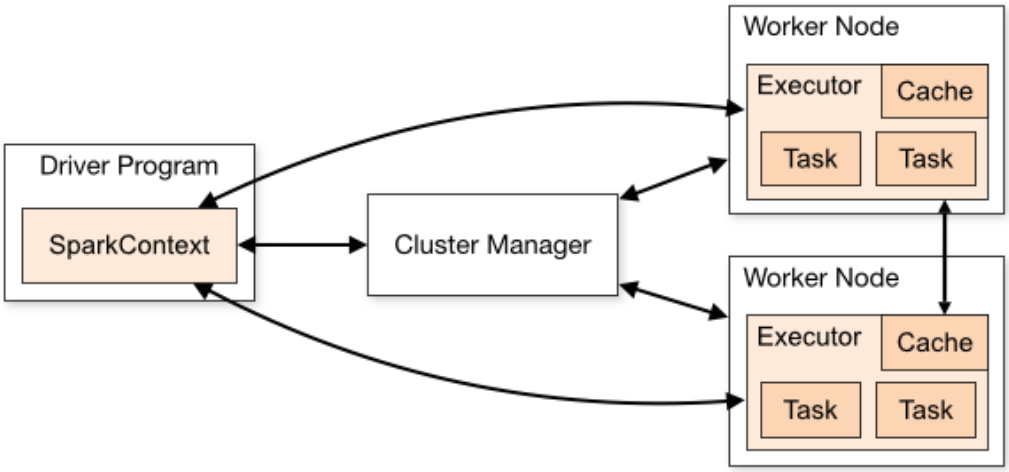
Phụ lục A. Giới thiệu về Apache Spark

A.1 Giới thiệu

Phần giới thiệu tổng quan và các thành phần của Spark đã được nói ở phần trên. Trong phần phụ lục này, em sẽ nói sâu hơn về Spark Core và Spark SQL, hai thành phần chính được sử dụng trong khóa luận.

A.2 Spark core

Ở mức tổng quan, mỗi ứng dụng Spark hoạt động theo kiến trúc *master/slave*. Trong ngữ cảnh của Spark, *master* được gọi là *driver* và *slave* được gọi là *executor*. Driver chứa hàm *main* của chương trình, có tác dụng chạy hàm *main* và phân chia dataset trên cluster, sau đó là áp dụng các hành động trên dataset. Driver không trực tiếp thực thi các hành động trên dataset, executor có nhiệm vụ này. Sau khi thực thi xong các hành động trên executor, dữ liệu cuối cùng sẽ được dồn về driver để tổng hợp lại. Hình dưới mô tả các thành phần driver/executor:



Hình A.1: Kiến trúc master/slave của Spark.

(Nguồn: https://spark.apache.org/docs/latest/cluster-overview.html)

Trong Spark, có một thành phần quan trọng làm việc với dữ liệu, gọi là RDD (Resilient distributed dataset). RDD đơn giản là một tập hợp phân tán không thể thay đổi của các đối tượng (immutable distributed collection of objects). Mỗi RDD được chia thành một hoặc nhiều partition, và các parition này sẽ được tính toán trên nhiều node khác nhau của cluster. RDD có một số đặc điểm sau:

- Khả năng tính toán trên bộ nhớ: RDD lưu các kết quả trung gian trên RAM thay vì lưu trên ổ đĩa (trường hợp của Hadoop)
- Immuatable: một khi tạo rồi thì không thể thay đổi nội dung.
- Partitioning: Mỗi RDD được chia thành nhiều parition nhỏ. Và mỗi partition này sẽ được xử lý trên các node của cluster
- Khả năng chịu lỗi: RDD có khả năng tái tạo lại dữ liệu đã bị mất. Nếu như một node bị chết/sự cố thì partition trên máy đó bị mất. RDD có khả năng tự tạo lại parition này.
- Lazy Evaluation: Là chiến lược tính toán mà sẽ trì hoãn quá trình tính toán một giá trị cho tới khi nào cần thiết sử dụng giá trị đó. Và nếu giá trị đó không bao giờ cần thiết được sử dụng, nó sẽ không bao giờ được tính toán. Về cơ bản, thao

tác trên RDD được chia làm 2 loại: Transformation và Action. Transformation là dạng hàm có kiểu trả về là một hoặc nhiều RDD. Tất cả các transformation đều lazy, tức là các transformation này không được tính ngay lúc gọi nhưng chỉ được "nhớ" là sẽ áp dụng mà thôi. Action là các hành động, hàm mà giá trị trả về không phải RDD. Khi gặp action, RDD mới thức hiện các transformation và trả kết quả về cho driver.

- Persistence: Khi RDD thực hiện xong 1 action, dữ liệu trên RDD sẽ không được lưu lại. Nếu ta muốn dùng RDD đó lại nhiều lần, ta phải persistion nó, tức là lưu nó trên RAM hoặc ổ đĩa.

Để tạo RDD, ta có hai cách:

- Tạo RDD từ collection đã có sẵn trên RAM. Cách này gọi là parallelize collection. Ta sẽ tạo một RDD có tên *wordsRDD* từ một collection.

```
JavaRDD<String> wordsRDD =
        sc.paralleize(Arrays.asList("fish", "cats", "dogs"));
```

- Tạo RDD từ dữ liệu bên ngoài: file, database, ... Ta sẽ tạo một RDD có tên *wordsRDD* từ file bên ngoài.

```
JavaRDD<String> wordsRDD = sc.textFile("/path/to/file.txt");
```

Tiếp theo, ta sẽ đi sơ lược một số API thông dụng. Đầu tiên, nói về tranformation. RDD tranformation là dạng hàm (functions) có kiểu trả về là một hoặc nhiều RDD. Tranformation luôn lazy và tranformation chỉ được thực hiện khi một action được gọi. Tranformation có một số hàm thông dụng sau:

- **map**(func): Trả về 1 RDD mới bằng cách truyền mỗi phần tử đầu vào(nguồn)qua hàm func. Ví dụ ta có 1 RDD {1,2,3,4,5}. Nếu ta sử dụng hàm map *rdd.map(x => x + 2)*, ta sẽ có được kết quả là: {3,4,5,6,7}

- **flatMap**(func): Tương tự map nhưng khác map ở chỗ, mỗi phần tử đầu vào qua flatMap sẽ trả về 0 hoặc nhiều phần tử đầu ra. Trong ngôn ngữ Java, flatMap sẽ trả về 1 List các phần tử.

- **filter**(func): Trả về 1 RDD mới bằng cách chọn những phần tử đầu vào mà hàm *func* trả về kết quả là *true*. Ví dụ, ta có một RDD {1,2,3,4,5}. Nếu ta sử dụng map filter *rdd.filter(x => x > 3)*, ta sẽ có được kết quả là {4,5}.

- **distinct**(): Trả về 1 RDD mới chứa mỗi phần tử là duy nhất của tập dữ liệu đầu vào. Ví dụ ta có 1 RDD {1,1,2,3,2}, nếu ta gọi *rdd.distinct()*, ta có được kết quả {1,2,3}.

- **repartition**(numberPartition): Điều chỉnh số lượng partition dựa vào thông số truyền vào.

- **join**(otherDataset): thực hiện phép join, tương tự như khi sử dụng SQL. Khi gọi tập dữ liệu có kiểu (K,V) và (K,W), nó sẽ trả về 1 cặp mới(K,(V,W))(nối 2 phần tử có cùng key). Lưu ý rằng, phép join chỉ có trong kiểu JavaPairRDD, không có trong JavaRDD.

Action là các hành động, hàm mà giá trị trả về không phải RDD. Giá trị của action được lưu ở driver hoặc external storage system như file, mysql, ... Gọi hàm action là một trong những cách để gửi dữ liệu từ executor về driver. Do đó, nếu gửi quá nhiều dữ liệu về driver, có thể dẫn đến tình trạng driver bị lỗi *OutOfMemory*. Một số hàm action thường thử dụng:

- **count**(): Trả về số phần tử của tập dữ liệu. Ví dụ, ta có 1 RDD {1,2,3,4,5}. Khi thực hiện *rdd.count()*, kết quả trả về là 5.

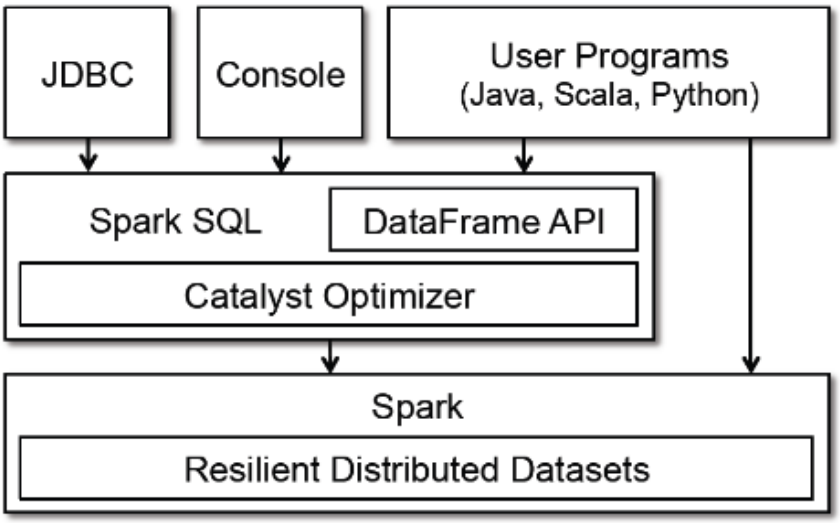
- **collect**(): Trả về tất cả các phần tử của tập dữ liệu như một mảng ở driverProgram. Hàm này hữu ích sau khi lọc hoặc thao tác khác mà trả về tập dữ liệu con đủ nhỏ. Nếu tập dữ liệu lớn, sẽ bị tình trạng *OutOfMemory* tại driver.

- **saveAsTextFile**(path): Ghi các phần tử của tập dữ liệu như 1 file text(hoặc tập file text)lên 1 thư mục trong hệ thống local, HDFS hoặc hệ thống hỗ trợHadoop bất kỳ

A.3 Spark SQL

Spark SQL là một thành phần của Spark dành riêng cho việc xử lý dữ liệu có cấu trúc. Spark SQL khiến việc thao tác với loại dữ liệu này trở nên dễ dàng và hiệu quả hơn. Có nhiều cách để thao tác với Spark SQL bao gồm việc sử dụng các câu lệnh truy vấn SQL hoặc sử dụng API được cung cấp sẵn. Ngoài ra, Spark SQL còn cung cấp bộ công cụ tự động tối ưu hóa code với tên Catalyst optimizer. Spark SQL cung cấp 3 khả năng chính:

- Có thể thao tác với nhiều loại dữ liệu từ nhiều nguồn, nhiều loại khác nhau như: JSON, Apache Hive, Parquet, Mysql, Apache Cassandra,....
- Nó cho phép truy vấn dữ liệu bằng cách dùng lệnh SQL.
- Spark SQL có khả năng tích hợp rất tốt giữa SQL và code Java/Python/Scala thông thường như khả năng join giữa RDD với bảng SQL,...

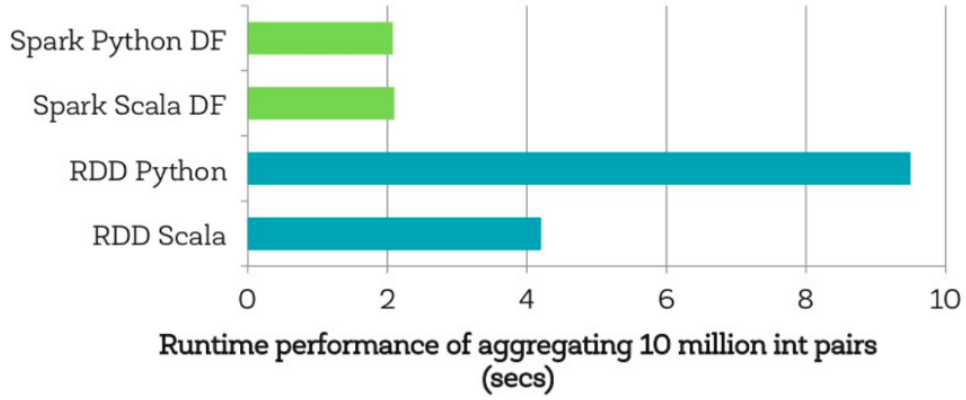


Hình A.2: Kiến trúc của Spark SQL.

(Nguồn: http://i.stanford.edu/ adityagp/courses/cs598/slides/sparksql.pptx)

Hình A.2 mô tả kiến trúc của Spark SQL. Ta thấy rằng, Spark SQL và DataFrameAPI nằm cùng tầng. Chúng đều được tối ưu hóa bằng Catalyst Optimizer trước khi chuyển

code về RDD để thực thi. Vậy, Spark SQL không loại bỏ nhưng được xây dựng trên RDD. Thông thường, chương trình viết bằng Spark SQL sẽ chạy nhanh hơn khi viết bằng RDD. Dưới đây là dẫn chứng thực nghiệm của công ty *Databricks Inc*, một trong những công ty có đóng góp lớn trong sự phát triển và bảo trì Apache Spark.



Hình A.3: So sánh thời gian chạy khi dùng Spark SQL và RDD.
(Nguồn: https://www.slideshare.net/databricks/introducing-dataframes-in-spark-for-large-scale-data-science/10)

Ta nhận thấy rằng, không có sự khác biệt về thời gian thực thi giữa việc lựa chọn ngôn ngữ lập trình nếu sử dụng Spark SQL. Trong khi đó, nếu sử dụng RDD, thời gian thực thi của ngôn ngữ Scala nhanh hơn Python. Đây cũng là một ưu điểm của việc sử dụng Spark SQL.

Phụ lục B. Giới thiệu về Spring Boot.

Spring Framework ban đầu được viết bởi Rod Johnson vào năm 2000 trên ngôn ngữ Java. Spring Framework cung cấp một mô hình lập trình và cấu hình toàn diện cho các ứng dụng doanh nghiệp (enterprise applications) trên bất kỳ nền tảng nào. Spring tập trung vào các ứng dụng doanh nghiệp vì vậy các lập trình viên chỉ việc tập trung vào xử lý các logic nghiệp vụ mà không cần quan tâm đến các yếu tố môi trường phát triển.

Spring Boot là một dự án khá nổi bật trong hệ sinh thái Spring Framework. Nếu như trước đây, công đoạn khởi tạo một dự án Spring khá vất vả từ việc khai báo các dependency trong file pom.xml cho đến cấu hình bằng XML hoặc annotation phức tạp, thì giờ đây với Spring Boot, chúng ta có thể tạo dự án Spring một cách nhanh chóng và cấu hình cũng đơn giản hơn. Dưới đây là một số tính năng nổi bật của Spring Boot:

- Tạo các ứng dụng Spring độc lập
- Nhúng trực tiếp Tomcat, Jetty hoặc Undertow (không cần phải deploy ra file WAR)
- Các starter dependency giúp việc cấu hình Maven đơn giản hơn
- Tự động cấu hình Spring khi cần thiết
- Không sinh code cấu hình và không yêu cầu phải cấu hình bằng XML ...
- Cung cấp nhiều plugins để làm việc với các loại cơ sở dữ liệu khác nhau.

Trong khóa luận, Spring Boot được dùng để viết Restful API.