ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH
oOo

BÁO CÁO MÔN HỌC BIG DATA

BUILDING A SCALABLE MOVIE RECOMMENDATION SYSTEM USING APACHE SPARK

GVHD: PGS.TS THOẠI NAM

HỌC VIÊN THỰC HIỆN
2480859 - Nguyễn Tấn Phú
2270594 - Đặng Thanh Tùng
2370499 - Nguyễn Văn Khoa

TP. HỒ CHÍ MINH – 2025

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

ALS	Alternating Least Squares
$LightGCN$	Light Graph Convolutional Network
$RMSE$	Root Mean Square Error
MAE	Mean Absolute Error
$HDFS$	Hadoop Distributed File System
CF	Collaborative Filtering
CBF	Content-Based Filtering
SVM	Support Vector Machine

DANH MỤC CÁC HÌNH

DANH MỤC CÁC BẢNG

Chương 1.

TỔNG QUAN

1.1 Bối cảnh và động lực nghiên cứu

Trong thời đại dữ liệu bùng nổ, các nền tảng giải trí như Netflix, Amazon Prime, hay YouTube không chỉ cung cấp nội dung đa dạng mà còn cần đảm bảo mỗi người dùng đều được tiếp cận với nội dung phù hợp nhất với sở thích và hành vi cá nhân. Chính vì vậy, hệ thống gợi ý (Recommendation System) đã trở thành một thành phần không thể thiếu, đóng vai trò như "người trợ lý thông minh", giúp định hướng trải nghiệm người dùng một cách hiệu quả [4].

Tuy nhiên, khi khối lượng dữ liệu và số lượng người dùng ngày càng tăng, việc xây dựng một hệ thống gợi ý không chỉ chính xác mà còn có khả năng mở rộng và tối ưu hiệu năng tính toán là một bài toán phức tạp. Các kỹ thuật truyền thống như Alternating Least Squares (ALS) tuy mang lại kết quả tốt với tập dữ liệu lớn, nhưng vẫn còn hạn chế trong việc khai thác sâu mối quan hệ phi tuyến giữa người dùng và sản phẩm [5]. Trong khi đó, LightGCN, một mô hình học sâu dựa trên Graph Convolutional Network, đã cho thấy hiệu quả vượt trội trong việc khai thác cấu trúc tương tác giữa người dùng và item thông qua lan truyền biểu diễn trên đồ thị [3].

Xuất phát từ những điểm mạnh của cả hai kỹ thuật trên, nghiên cứu này hướng đến việc kết hợp mô hình ALS và LightGCN trong môi trường tính toán phân tán Apache Spark, nhằm xây dựng một hệ thống gợi ý phim vừa có khả năng mở rộng, vừa khai thác được thông tin ngữ cảnh sâu từ dữ liệu tương tác người dùng.

1.2 Mục tiêu nghiên cứu

Nghiên cứu đặt ra các mục tiêu cụ thể như sau:

- Xây dựng hệ thống có khả năng mở rộng trên nền tảng Apache Spark.
- Ứng dụng mô hình ALS để khai thác nhân tố ẩn giữa người dùng và phim.
- Triển khai LightGCN để học biểu diễn từ đồ thị tương tác.
- Đề xuất mô hình kết hợp ALS và LightGCN nhằm tận dụng ưu điểm của cả hai.

- Đánh giá hiệu năng qua các chỉ số: RMSE, MAE, Precision@k, Recall@k.
- Khảo sát khả năng mở rộng khi áp dụng trên dữ liệu lớn.

1.3 Phạm vi và giới hạn

Nghiên cứu tập trung vào bài toán xây dựng hệ thống gợi ý phim dựa trên lịch sử tương tác của người dùng, tức là dựa vào các đánh giá, lượt xem hoặc hành vi trước đó để đưa ra các đề xuất phù hợp. Dữ liệu sử dụng trong nghiên cứu chủ yếu đến từ tập MovieLens 1M – một bộ dữ liệu phổ biến và được chuẩn hóa trong cộng đồng nghiên cứu hệ thống gợi ý. Hai mô hình là Alternating Least Squares (ALS) và Light Graph Convolutional Network (LightGCN) được lựa chọn và kết hợp nhằm khai thác đồng thời các yếu tố tiềm ẩn và mối quan hệ tương tác trên đồ thị, từ đó cải thiện độ chính xác của hệ thống gợi ý. Toàn bộ quá trình được triển khai trong môi trường Apache Spark nhằm tận dụng khả năng xử lý phân tán và đánh giá tính mở rộng của hệ thống khi dữ liệu tăng lên.

Tuy nhiên, vẫn tồn tại một số giới hạn nhất định. Thứ nhất, hệ thống chưa xét đến các yếu tố ngữ cảnh như thời gian, thiết bị truy cập hay vị trí địa lý, vốn có thể làm tăng tính cá nhân hóa trong gợi ý. Thứ hai, quá trình kết hợp giữa ALS và LightGCN hiện chỉ được thực hiện ở mức biểu diễn, chưa tích hợp thành một mô hình học chung có thể tối ưu toàn cục. Cuối cùng, do hạn chế về hạ tầng kỹ thuật, hệ thống chủ yếu được triển khai trong môi trường Spark local hoặc trên cụm máy nhỏ, chưa thể đánh giá đầy đủ khả năng hoạt động trên quy mô lớn thực tế.

1.4 Phương pháp tiếp cận

Quá trình triển khai được thực hiện theo các bước chính như sau: (1) - Tiền xử lý dữ liệu: Làm sạch, chuẩn hóa và chuyển đổi dữ liệu tương tác sang định dạng phù hợp; (2) - Huấn luyện mô hình riêng biệt: ALS được triển khai trên Apache Spark MLlib để huấn luyện các biểu diễn tuyến tính của người dùng và item. LightGCN được huấn luyện trên PyTorch/LightGCN framework để học biểu diễn từ đồ thị tương tác; (3) - Tổng hợp biểu diễn: Biểu diễn đầu ra từ hai mô hình được kết hợp bằng kỹ thuật hợp nhất tuyến tính hoặc mạng học sâu (MLP); (4) - Đánh giá mô hình: So sánh kết quả của từng mô hình đơn lẻ và mô hình kết hợp trên các tiêu chí đo lường phổ biến; (5) - Phân tích khả năng mở rộng: Quan sát sự thay đổi hiệu suất và thời gian xử lý khi tăng quy mô dữ liệu.

1.5 Cấu trúc báo cáo

Báo cáo được tổ chức thành 5 chương như sau:

- **Chương 1:** Tổng quan
- **Chương 2:** Cơ sở lý thuyết
- **Chương 3:** Phương pháp thực hiện
- **Chương 4:** Kết quả thực nghiệm
- **Chương 5:** Kết luận và hướng phát triển.

Chương 2.

CƠ SỞ LÝ THUYẾT

2.1 Khái niệm và đặc trưng dữ liệu lớn

Dữ liệu lớn (Big Data) là một khái niệm mô tả các tập dữ liệu có khối lượng lớn, tốc độ phát sinh nhanh và có cấu trúc đa dạng đến mức vượt quá khả năng lưu trữ, quản lý và xử lý của các hệ thống cơ sở dữ liệu truyền thống [6]. Khái niệm này lần đầu tiên được hệ thống hóa bởi [7] với mô hình 3V gồm: Volume (Khối lượng), Velocity (Tốc độ) và Variety (Đa dạng).

- Volume (Khối lượng): Đại diện cho kích thước khổng lồ của dữ liệu được tạo ra từ nhiều nguồn như mạng xã hội, thiết bị IoT, nhật ký hệ thống, hoặc dữ liệu giao dịch thương mại điện tử. Những hệ thống hiện đại có thể thu thập dữ liệu ở mức hàng terabyte đến hàng petabyte mỗi ngày [8].
- Velocity (Tốc độ): Chỉ tốc độ tạo ra, truyền tải và xử lý dữ liệu. Trong nhiều ứng dụng như hệ thống gợi ý, dữ liệu người dùng phát sinh liên tục và yêu cầu được xử lý gần như ngay lập tức để phản hồi theo thời gian thực [9].
- Variety (Đa dạng): Mô tả sự đa dạng về định dạng và nguồn dữ liệu. Dữ liệu có thể là có cấu trúc (structured – ví dụ bảng cơ sở dữ liệu), bán cấu trúc (semi-structured – ví dụ XML, JSON), hoặc phi cấu trúc (unstructured – ví dụ văn bản tự do, hình ảnh, video) [6].

Để phản ánh rõ hơn tính chất phức tạp của dữ liệu lớn, mô hình 3V đã được mở rộng thành 5V, bổ sung thêm hai yếu tố:

- Veracity (Độ tin cậy): Dữ liệu lớn thường chứa thông tin không đồng nhất, thiếu chính xác hoặc có nhiễu, đặt ra yêu cầu cao trong việc làm sạch và xác thực dữ liệu [10].
- Value (Giá trị): Giá trị thực sự của dữ liệu lớn chỉ đạt được khi có thể khai thác và phân tích dữ liệu một cách hiệu quả để tạo ra tri thức mới hoặc hỗ trợ ra quyết định có ích [6].

Ngoài ra, một số học giả còn đề cập đến yếu tố Variability (tính biến động theo thời gian) và Visualization (khả năng trực quan hóa), cho thấy rằng dữ liệu

lớn không chỉ thách thức về mặt kỹ thuật, mà còn về khả năng hiểu và trình bày thông tin sao cho có ý nghĩa với con người [9].

2.2 Vai trò và thách thức

Dữ liệu lớn ngày càng trở thành một tài sản chiến lược, đóng vai trò trung tâm trong nhiều lĩnh vực như y tế, tài chính, giáo dục, và đặc biệt là các hệ thống cá nhân hóa. Trong bối cảnh cạnh tranh số, việc khai thác hiệu quả dữ liệu lớn giúp tổ chức đưa ra quyết định nhanh chóng, giảm thiểu rủi ro và tối ưu hóa hiệu suất hoạt động.

Tuy nhiên, khai thác dữ liệu lớn cũng đi kèm với nhiều thách thức: (1): Lưu trữ: Đòi hỏi các giải pháp lưu trữ phân tán như HDFS hoặc cơ sở dữ liệu NoSQL để quản lý dữ liệu phi cấu trúc hiệu quả; (2): Tính toán: Cần đến nền tảng tính toán phân tán như Apache Hadoop, Apache Spark để xử lý dữ liệu với độ trễ thấp; (3): Phân tích thời gian thực: Các ứng dụng như hệ thống gợi ý hoặc giám sát an ninh yêu cầu xử lý dòng dữ liệu theo thời gian thực; (4): Bảo mật: Cần có chính sách kiểm soát truy cập, mã hóa và tuân thủ các quy định về quyền riêng tư dữ liệu [8].

2.3 Dữ liệu lớn trong hệ thống gợi ý

Trong hệ thống gợi ý, dữ liệu người dùng có thể tăng trưởng theo cấp số nhân, đặc biệt ở các nền tảng phim trực tuyến, mạng xã hội hoặc thương mại điện tử. Big Data đóng vai trò nền tảng trong việc:

- Lưu trữ hiệu quả các tương tác giữa người dùng và sản phẩm.
- Huấn luyện các mô hình gợi ý trên toàn bộ tập dữ liệu không cần lấy mẫu.
- Tối ưu hóa khả năng phản hồi theo thời gian thực cho người dùng.

2.4 Tổng quan về hệ thống gợi ý

Hệ thống gợi ý (Recommendation System) là một lĩnh vực nghiên cứu chủ chốt trong khoa học dữ liệu, với mục tiêu chính là đưa ra các dự đoán về mức độ yêu thích của người dùng đối với các đối tượng mà họ chưa từng tương tác. Các hệ thống này đã và đang được ứng dụng rộng rãi trong nhiều nền tảng như thương mại điện tử, giải trí số, mạng xã hội và giáo dục trực tuyến nhằm cá nhân hóa trải nghiệm người dùng [4].

Về mặt phân loại, hệ thống gợi ý thường được chia thành ba nhóm chính: (1): Lọc cộng tác (Collaborative Filtering), tận dụng hành vi tương tác giữa người dùng với các mục tiêu; (2): Lọc dựa trên nội dung (Content-Based Filtering), tập trung vào phân tích đặc trưng của sản phẩm và người dùng; và (3): Mô hình lai (Hybrid Recommendation), kết hợp nhiều phương pháp để khắc phục hạn chế riêng lẻ của từng loại.

2.5 Thuật toán gợi ý phổ biến

2.5.1 Lọc cộng tác (Collaborative Filtering)

Lọc cộng tác (Collaborative Filtering – CF) là một trong những kỹ thuật phổ biến và hiệu quả nhất trong hệ thống gợi ý. Phương pháp này đưa ra đề xuất dựa trên hành vi và sở thích của tập người dùng, thay vì dựa trên nội dung của sản phẩm. Nguyên lý cốt lõi của lọc cộng tác là: "Người dùng có xu hướng yêu thích những sản phẩm mà những người dùng tương tự đã yêu thích" [11]. CF không yêu cầu thông tin về đặc trưng của sản phẩm hay hồ sơ chi tiết của người dùng, do đó rất phù hợp với các bài toán trong đó chỉ có dữ liệu về tương tác (rating, click, mua hàng, v.v.). Có hai nhóm chính trong lọc cộng tác: dựa trên người dùng (user-based) và dựa trên sản phẩm (item-based).

2.5.1.1 Lọc cộng tác dựa trên người dùng (User-based CF)

Trong phương pháp này, hệ thống tìm những người dùng có hành vi tương đồng với người dùng mục tiêu, sau đó dự đoán mức độ yêu thích của người dùng đó với một sản phẩm dựa trên đánh giá của những người dùng tương tự.

Giả sử ta có người dùng u và sản phẩm i, giá trị đánh giá dự đoán $\hat{r}_{u,i}$ được tính bằng trung bình có trọng số các đánh giá của những người dùng tương tự v:

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum\limits_{v \in N(u)} \text{sim}(u,v) \cdot (r_{v,i} - \bar{r}_v)}{\sum\limits_{v \in N(u)} |\text{sim}(u,v)|} \tag{2.5.1}$$

Trong đó:

- $\bar{r}_u$: Trung bình các đánh giá của người dùng u
- $\text{sim}(u,v)$: Độ tương tự giữa người dùng u và v (thường sử dụng Cosine similarity hoặc Pearson correlation)
- $N(u)$: Tập người dùng tương tự với người dùng u

2.5.1.2 Lọc cộng tác dựa trên sản phẩm (Item-based CF)

Thay vì tìm người dùng tương tự, phương pháp này xác định các sản phẩm có đặc điểm tương đồng với sản phẩm đã được người dùng đánh giá, và sử dụng các đánh giá đó để dự đoán đánh giá mới. Công thức dự đoán như sau:

$$\hat{r}_{u,i} = \frac{\sum\limits_{j \in N(i)} \text{sim}(i,j) \cdot r_{u,j}}{\sum\limits_{j \in N(i)} |\text{sim}(i,j)|} \tag{2.5.2}$$

Trong đó:

- $N(i)$: Tập các sản phẩm tương tự với sản phẩm i
- $\text{sim}(i,j)$: Độ tương tự giữa sản phẩm i và sản phẩm j
- $r_{u,j}$: Đánh giá của người dùng u cho sản phẩm j

Phương pháp này thường hiệu quả hơn trong môi trường có số lượng sản phẩm ít hơn số người dùng, hoặc khi dữ liệu tương tác giữa người dùng và sản phẩm thưa.

2.5.1.3 Lọc cộng tác theo mô hình (Model-based CF)

Ngoài hai phương pháp truyền thống nêu trên, CF còn có thể được triển khai theo hướng xây dựng mô hình học máy. Một trong những phương pháp phổ biến là Phân rã ma trận (Matrix Factorization), trong đó ma trận tương tác giữa người dùng và sản phẩm được phân rã thành hai ma trận đại diện cho vector đặc trưng của người dùng và sản phẩm [12]. Công thức phân rã như sau:

$$R \approx U \cdot V^\top \tag{2.5.3}$$

Trong đó:

- R: Ma trận tương tác ban đầu (người dùng $\times$ sản phẩm)
- U: Ma trận đặc trưng của người dùng
- V: Ma trận đặc trưng của sản phẩm

Giá trị dự đoán đánh giá $\hat{r}_{ui}$ được tính bằng:

$$\hat{r}_{ui} = \mathbf{u}_u^\top \cdot \mathbf{v}_i \tag{2.5.4}$$

Phương pháp này hiệu quả cao, khả năng mở rộng tốt, và đã trở thành nền tảng cho nhiều hệ thống gợi ý quy mô lớn.

2.5.2 Lọc dựa trên nội dung (Content-Based Filtering)

Lọc dựa trên nội dung (Content-Based Filtering – CBF) [13] là một trong những phương pháp gợi ý phổ biến, hoạt động dựa trên giả định rằng: nếu một người dùng ưa thích một số đối tượng có chung đặc điểm, thì họ cũng sẽ có xu hướng yêu thích các đối tượng khác có cùng đặc trưng.

Khác với phương pháp lọc cộng tác, vốn dựa vào hành vi của các người dùng khác, lọc dựa trên nội dung khai thác các đặc tính nội tại của sản phẩm (ví dụ: thể loại phim, từ khóa, đạo diễn...) để đưa ra đề xuất. Phương pháp này xây dựng một hồ sơ người dùng phản ánh sở thích cá nhân, sau đó so sánh hồ sơ này với đặc trưng của các sản phẩm chưa tương tác để tìm ra các đối tượng phù hợp nhất.

2.5.2.1 Mô hình biểu diễn đặc trưng

Trong Content-Based Filtering, mỗi sản phẩm được biểu diễn dưới dạng một vector đặc trưng $\mathbf{x}_i \in \mathbb{R}^d$, trong đó d là số chiều của không gian đặc trưng (ví dụ: số lượng từ khóa, thể loại, tính năng...).

Giả sử người dùng u đã tương tác hoặc đánh giá một tập hợp các sản phẩm I_u, hồ sơ người dùng $\mathbf{p}_u \in \mathbb{R}^d$ có thể được xây dựng bằng cách tính trung bình có trọng số các vector đặc trưng:

$$\mathbf{p}_u = \frac{1}{|I_u|} \sum_{i \in I_u} r_{u,i} \cdot \mathbf{x}_i \tag{2.5.5}$$

Trong đó:

- $r_{u,i}$: mức độ đánh giá của người dùng u đối với sản phẩm i
- $\mathbf{x}_i$: vector đặc trưng của sản phẩm i
- $|I_u|$: số lượng sản phẩm mà người dùng đã tương tác

Hồ sơ người dùng phản ánh mức độ ưu tiên của họ đối với từng đặc tính sản phẩm, từ đó hệ thống có thể cá nhân hóa gợi ý.

2.5.2.2 Dự đoán và xếp hạng

Sau khi xây dựng được hồ sơ người dùng $\mathbf{p}_u$, hệ thống tính mức độ tương đồng giữa hồ sơ này và các sản phẩm mà người dùng chưa từng tương tác để đưa ra dự đoán.

Phổ biến nhất là sử dụng **cosine similarity** để đo độ tương đồng giữa hồ sơ người dùng và vector đặc trưng của sản phẩm:

$$\text{sim}(u, i) = \cos(\mathbf{p}_u, \mathbf{x}_i) = \frac{\mathbf{p}_u^\top \mathbf{x}_i}{\|\mathbf{p}_u\| \cdot \|\mathbf{x}_i\|} \tag{2.5.6}$$

Dựa vào điểm số tương đồng $\text{sim}(u, i)$, hệ thống sẽ xếp hạng và lựa chọn các sản phẩm có giá trị cao nhất để gợi ý cho người dùng u.

2.5.2.3 Các kỹ thuật thường dùng trong Content-Based Filtering

- *TF-IDF*: Áp dụng khi sản phẩm được mô tả bằng văn bản, dùng để trích xuất trọng số từ khóa nổi bật trong mô tả sản phẩm.
- *One-hot encoding*: Mã hóa các thuộc tính phân loại như thể loại phim, quốc gia, loại sản phẩm.
- *Embedding*: Sử dụng mạng học sâu để học biểu diễn đặc trưng phức tạp hơn.
- *Mô hình học máy*: Dùng các thuật toán như Naive Bayes, SVM hoặc mạng neural để dự đoán mức độ yêu thích dựa trên đặc trưng sản phẩm.

2.5.3 Hybrid Recommendation

Hybrid Recommendation là sự kết hợp giữa hai hoặc nhiều kỹ thuật gợi ý nhằm khai thác ưu điểm của từng phương pháp riêng lẻ và khắc phục các nhược điểm thường gặp. Thông thường, các hệ thống lai được xây dựng từ sự kết hợp giữa lọc cộng tác (Collaborative Filtering – CF) và lọc dựa trên nội dung (Content-Based Filtering – CBF) [14]. Sự kết hợp này giúp hệ thống nâng cao chất lượng dự đoán, cải thiện khả năng cá nhân hóa, và đặc biệt là giảm thiểu các vấn đề phổ biến như cold-start (người dùng hoặc sản phẩm mới), dữ liệu thưa (sparsity), và over-specialization.

2.5.3.1 Các phương pháp kết hợp

a) Kết hợp tuyến tính theo trọng số (Weighted Hybrid)
Đây là phương pháp phổ biến và trực quan nhất trong hệ thống gợi ý lai. Trong đó, điểm đánh giá dự đoán của người dùng được tính bằng cách kết hợp hai kết quả dự đoán từ mô hình Content-Based Filtering (CBF) và Collaborative Filtering (CF), theo trung bình có trọng số.

$$\hat{r}_{u,i} = \alpha \cdot \hat{r}_{u,i}^{CBF} + (1 - \alpha) \cdot \hat{r}_{u,i}^{CF}, \quad 0 \leq \alpha \leq 1 \tag{2.5.7}$$

Trong đó:

- $\hat{r}_{u,i}$: điểm đánh giá dự đoán của người dùng u với sản phẩm i.
- $\hat{r}_{u,i}^{CBF}$: điểm dự đoán từ mô hình Content-Based Filtering.
- $\hat{r}_{u,i}^{CF}$: điểm dự đoán từ mô hình Collaborative Filtering.
- α: trọng số điều chỉnh mức ưu tiên giữa hai mô hình, được chọn trong khoảng $[0, 1]$.

Việc điều chỉnh α cho phép hệ thống linh hoạt trong việc cân bằng giữa đặc điểm cá nhân hóa theo nội dung và sự đồng thuận từ cộng đồng người dùng khác.

b) Kết hợp chuyển mạch (Switching Hybrid)

Phương pháp này sử dụng một mô hình hoặc một chiến lược dự đoán tại mỗi thời điểm, tùy vào đặc điểm ngữ cảnh hoặc điều kiện dữ liệu. Công thức thực hiện như sau:

$$\hat{r}_{u,i} = \begin{cases} \hat{r}_{u,i}^{CBF}, & \text{nếu } |\mathcal{I}_u| < \tau \\ \hat{r}_{u,i}^{CF}, & \text{nếu } |\mathcal{I}_u| \geq \tau \end{cases} \tag{2.5.8}$$

Trong đó:

- $|\mathcal{I}_u|$: số lượng sản phẩm mà người dùng u đã đánh giá hoặc tương tác.
- τ: ngưỡng để quyết định chuyển đổi giữa hai mô hình.
- $\hat{r}_{u,i}^{CBF}$: điểm dự đoán từ mô hình Content-Based Filtering.
- $\hat{r}_{u,i}^{CF}$: điểm dự đoán từ mô hình Collaborative Filtering.

Phương pháp này lựa chọn mô hình phù hợp dựa trên mức độ hoạt động của người dùng: nếu người dùng tương tác ít thì ưu tiên CBF, nếu tương tác nhiều thì sử dụng CF.

c) Kết hợp phân tầng (Cascade Hybrid)

Phương pháp này thực hiện lọc hai bước. Mô hình đầu tiên được dùng để tạo danh sách ứng viên, mô hình thứ hai dùng để tái xếp hạng hoặc lọc chi tiết.

- Mô hình Content-Based Filtering (CBF) tạo danh sách ứng viên ban đầu: $\hat{r}_{u,i}^{CBF} \Rightarrow$ Danh sách top-k sản phẩm
- Mô hình Collaborative Filtering (CF) đánh giá lại các sản phẩm này: $\hat{r}_{u,i}^{CF} \Rightarrow$ Xếp hạng cuối cùng

$$\text{Top-}N_u = \text{argsort}_{i \in \mathcal{I}} \left(\hat{r}_{u,i}^{CF} \mid i \in \text{Top-}k_{CBF} \right) \tag{2.5.9}$$

Trong đó:

- $\hat{r}_{u,i}^{CBF}$: điểm dự đoán từ mô hình CBF.
- $\hat{r}_{u,i}^{CF}$: điểm dự đoán từ mô hình CF.
- Top-k_{CBF}: danh sách k sản phẩm được chọn từ CBF.
- Top-N_u: danh sách N sản phẩm gợi ý cuối cùng cho người dùng u.

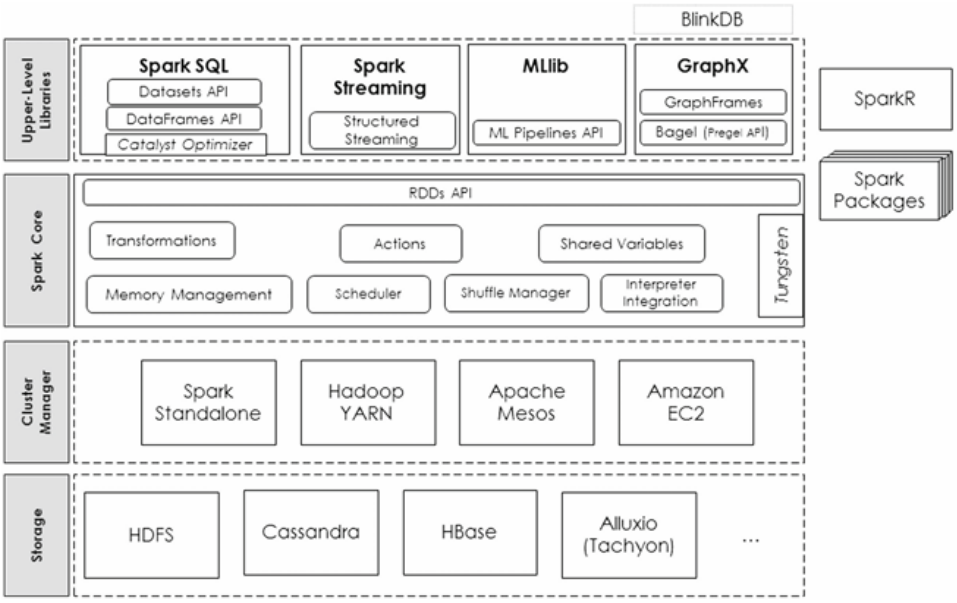
2.6 Apache Spark: Kiến trúc và tính năng

Apache Spark [1,15] là một hệ thống tính toán phân tán mã nguồn mở được thiết kế nhằm tăng tốc xử lý dữ liệu lớn, đặc biệt trong các tác vụ có tính lặp lại cao hoặc đòi hỏi tính tương tác. Spark được phát triển tại AMPLab của Đại học California, Berkeley, với mục tiêu khắc phục các hạn chế về hiệu năng của Hadoop MapReduce bằng cách hỗ trợ xử lý dữ liệu trong bộ nhớ (in-memory processing) và cung cấp một mô hình lập trình linh hoạt hơn.

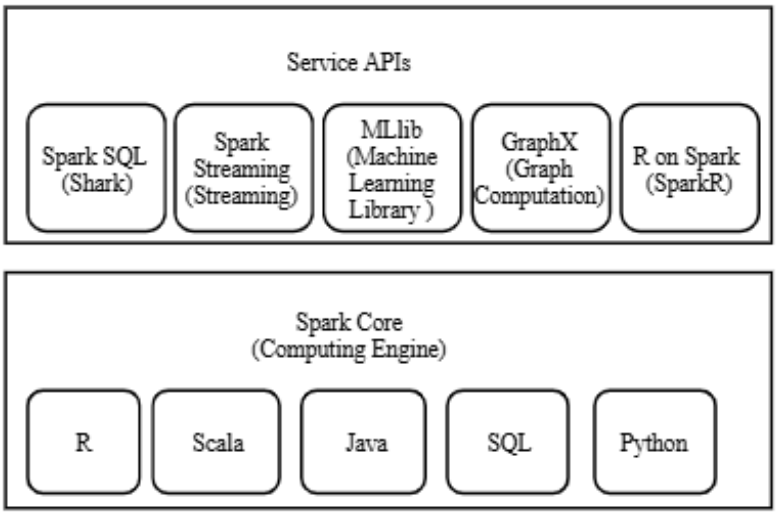
1) Mô hình xử lý theo lô (Batch Processing) của Spark: Ưu điểm lớn nhất của Spark so với MapReduce là khả năng tính toán trong bộ nhớ (in-memory computation). Spark chỉ tương tác với ổ đĩa trong hai trường hợp: tải dữ liệu ban đầu vào bộ nhớ và lưu kết quả cuối cùng trở lại bộ nhớ. Tất cả các bước xử lý trung gian đều được thực hiện trong bộ nhớ. Việc xử lý trong bộ nhớ này giúp Spark nhanh hơn đáng kể so với các hệ thống xử lý theo lô khác như Hadoop. Ngoài ra, Spark còn sử dụng phương pháp tối ưu hóa toàn diện (holistic optimization), cho phép phân tích toàn bộ chuỗi tác vụ từ trước. Điều này được thực hiện bằng cách tạo ra các đồ thị có hướng không chu trình (Directed Acyclic Graphs – DAGs) để biểu diễn tất cả các thao tác, dữ liệu và mối quan hệ giữa chúng [16]. Để hỗ trợ tính năng xử lý trong bộ nhớ, Spark sử dụng RDD (Resilient Distributed Dataset). RDD là một cấu trúc dữ liệu chỉ đọc, được lưu trữ trong bộ nhớ, giúp Spark có khả năng chịu lỗi mà không cần ghi dữ liệu ra ổ đĩa sau mỗi thao tác.

2) Mô hình xử lý luồng (Stream Processing) của Spark: Bên cạnh xử lý theo lô, Spark còn hỗ trợ khả năng xử lý luồng thông qua kỹ thuật vi-lô (micro-batching). Trong micro-batching, luồng dữ liệu được chia thành các lô rất nhỏ, và mỗi lô nhỏ này được xử lý như một tác vụ thông thường của Spark [16]. Mặc dù kỹ thuật vi-lô này hoạt động khá hiệu quả, nhưng vẫn có thể dẫn đến

sự khác biệt về hiệu năng so với các hệ thống xử lý luồng thực sự (true stream processing frameworks).



Hình 2.1: Kiến trúc cấp cao của hệ thống Apache Spark [1]



Hình 2.2: Apache Spark Ecosystem [2]

Hệ sinh thái Apache Spark bao gồm các thành phần chính sau đây:

- *Spark SQL*: Trước đây được biết đến với tên gọi Shark. Spark SQL là một khung phân tán (distributed framework) hỗ trợ xử lý dữ liệu có cấu trúc và bán cấu trúc. Nó tạo điều kiện cho việc xây dựng các ứng dụng phân tích và tương tác trên cả dữ liệu luồng thời gian thực lẫn dữ liệu lịch sử,

với khả năng truy xuất từ nhiều nguồn khác nhau như JSON, Parquet và bảng Hive.

- *Spark Streaming*: Cho phép người dùng xử lý luồng dữ liệu trong thời gian thực. Để thực hiện phân tích dữ liệu luồng, Spark Streaming cải tiến khả năng lập lịch nhanh của Apache Spark bằng cách đưa dữ liệu vào các vi-lô (mini batches). Sau đó, một phép biến đổi (transformation) được áp dụng lên các vi-lô này, vốn dễ dàng được thu nhận từ các luồng trực tiếp và nguồn dữ liệu như Twitter, Apache Kafka, cảm biến IoT và Amazon Kinesis.

- *MLlib*: Cung cấp các thuật toán chất lượng cao với tốc độ xử lý nhanh, giúp việc áp dụng và mở rộng học máy trở nên dễ dàng hơn. MLlib tích hợp nhiều thuật toán học máy như hồi quy, phân loại, phân cụm và đại số tuyến tính. Thư viện này cũng bao gồm các thành phần cấp thấp cho học máy, như thuật toán tối ưu hóa descent gradient tổng quát. Ngoài ra, MLlib còn hỗ trợ các chức năng khác như đánh giá mô hình và nhập dữ liệu. Người dùng có thể sử dụng MLlib với các ngôn ngữ như Java, Scala và Python.

- *GraphX*: Là một công cụ tính toán đồ thị, cho phép xây dựng, thao tác, biến đổi và thực thi các dữ liệu có cấu trúc đồ thị ở quy mô lớn. GraphX bao gồm nhiều API RDD của Spark, hỗ trợ việc tạo các đồ thị có hướng.

- *Spark Core*: Là nền tảng chính mà các chức năng khác của Apache Spark được xây dựng trên đó. Spark Core cung cấp một loạt các API và ứng dụng cho các ngôn ngữ lập trình như Scala, Java và Python, giúp quá trình phát triển trở nên dễ dàng hơn. Tính toán trong bộ nhớ (in-memory computation) được triển khai trong Spark Core nhằm tăng tốc độ xử lý và giải quyết các hạn chế của MapReduce.

- *SparkR*: Là một gói mở rộng cho ngôn ngữ R, cho phép các nhà khoa học dữ liệu khai thác sức mạnh của Spark trực tiếp từ giao diện R shell. Trong R, DataFrame là cấu trúc dữ liệu cơ bản để xử lý dữ liệu, và tương tự, SparkR sử dụng SparkR DataFrame làm đơn vị xử lý chính. SparkR có thể thực hiện nhiều thao tác trên tập dữ liệu lớn như lựa chọn, lọc và tổng hợp

2.7 Thuật toán ALS (Alternating Least Squares)

Thuật toán ALS (Alternating Least Squares) là một kỹ thuật phân rã ma trận (Matrix Factorization) phổ biến được sử dụng trong hệ thống gợi ý, đặc biệt là lọc cộng tác dựa trên user–item. Mục tiêu chính của ALS là dự đoán những giá trị bị thiếu trong ma trận tương tác (ví dụ: điểm đánh giá của người dùng cho sản phẩm).

2.7.1 Mô hình toán học

Giả sử có một ma trận đánh giá $\mathbf{R} \in \mathbb{R}^{m \times n}$, trong đó:

- m: số người dùng,
- n: số sản phẩm,
- R_{ui}: điểm số người dùng u đánh giá cho sản phẩm i,

Nhiều giá trị trong $\mathbf{R}$ bị thiếu (không có đánh giá).
Mục tiêu là phân rã $\mathbf{R}$ thành tích của hai ma trận có kích thước thấp hơn:

$$\mathbf{R} \approx \mathbf{U} \cdot \mathbf{V}^T \tag{2.7.1}$$

Trong đó:

- $\mathbf{U} \in \mathbb{R}^{m \times k}$: ma trận đặc trưng người dùng,
- $\mathbf{V} \in \mathbb{R}^{n \times k}$: ma trận đặc trưng sản phẩm,
- k: số chiều ẩn (latent factors), với $k \ll m, n$.

2.7.2 Hàm mục tiêu (Loss Function)

Để tìm các ma trận $\mathbf{U}$ và $\mathbf{V}$, thuật toán tối ưu hóa hàm mất mát sau:

$$\min_{\mathbf{U},\mathbf{V}} \sum_{(u,i)\in\kappa} \left(R_{ui} - \mathbf{U}_u \cdot \mathbf{V}_i^T\right)^2 + \lambda \left(\|\mathbf{U}_u\|^2 + \|\mathbf{V}_i\|^2\right) \tag{2.7.2}$$

Trong đó:

- κ: tập các cặp người dùng – sản phẩm có đánh giá thực,
- λ: hệ số điều chuẩn (regularization) giúp tránh overfitting,
- $\|\cdot\|^2$: chuẩn L2.

2.7.3 Phương pháp tối ưu hóa luân phiên

Vì hàm mất mát trên không lồi (non-convex) theo cả $\mathbf{U}$ và $\mathbf{V}$ cùng lúc, nhưng là bài toán tối ưu bình phương nhỏ nhất (least squares) nếu cố định một trong hai ma trận, nên ALS giải quyết bằng cách:

1. Giữ cố định ma trận $\mathbf{V}$, giải phương trình tối ưu cho $\mathbf{U}$:

$$\mathbf{U}^* = \arg\min_{\mathbf{U}} \left\|\mathbf{R} - \mathbf{U}\mathbf{V}^T\right\|_F^2 + \lambda \left\|\mathbf{U}\right\|_F^2 \tag{2.7.3}$$

2. Giữ cố định ma trận $\mathbf{U}$, giải phương trình tối ưu cho $\mathbf{V}$:

$$\mathbf{V}^* = \arg\min_{\mathbf{V}} \left\|\mathbf{R} - \mathbf{U}\mathbf{V}^T\right\|_F^2 + \lambda \left\|\mathbf{V}\right\|_F^2 \tag{2.7.4}$$

3. Lặp lại hai bước trên luân phiên cho đến khi hội tụ (sai số giảm đủ nhỏ hoặc đạt số vòng lặp tối đa).

2.7.4 Phiên bản ALS mở rộng cho dữ liệu ngầm định

Spark MLlib còn hỗ trợ mô hình ALS với dữ liệu *implicit feedback*, tức là khi không có đánh giá rõ ràng (explicit ratings), thay thế bằng tương tác (click, xem, mua). Khi đó, hàm mất mát được điều chỉnh như sau:

$$\min_{\mathbf{U},\mathbf{V}} \sum_{u,i} c_{ui} \left(p_{ui} - \mathbf{U}_u \cdot \mathbf{V}_i^T\right)^2 + \lambda \left(\|\mathbf{U}_u\|^2 + \|\mathbf{V}_i\|^2\right) \tag{2.7.5}$$

Trong đó:

- $p_{ui} \in \{0, 1\}$: biểu thị việc người dùng u có tương tác với sản phẩm i hay không,
- $c_{ui} = 1 + \alpha \cdot r_{ui}$: độ tin cậy vào tương tác, với r_{ui} là tần suất hoặc cường độ tương tác,
- α: hệ số khuếch đại độ tin cậy.

Thuật toán ALS sở hữu nhiều ưu điểm nổi bật, đặc biệt phù hợp với các hệ thống gợi ý quy mô lớn. Trước hết, tính song song cao là một trong những lợi thế quan trọng của ALS, bởi mỗi hàng trong ma trận đặc trưng user U hoặc ma trận đặc trưng item V có thể được tính toán một cách độc lập. Điều này giúp thuật toán dễ dàng triển khai hiệu quả trên các nền tảng xử lý dữ liệu phân tán như Apache Spark. Bên cạnh đó, ALS hoạt động hiệu quả trên tập dữ liệu lớn, đáp ứng tốt nhu cầu của các hệ thống gợi ý có hàng triệu người dùng và

sản phẩm. Đặc biệt, ALS còn hỗ trợ mở rộng cho dữ liệu ngầm định (implicit feedback) – tức các tương tác không có đánh giá rõ ràng như số lượt xem, số lần nhấn vào sản phẩm – giúp mở rộng phạm vi ứng dụng trong các hệ thống gợi ý hiện đại.

2.8 Cơ sở lý thuyết của LightGCN

Trong những năm gần đây, Graph Neural Networks (GNNs) đã trở thành một xu hướng nổi bật trong xây dựng hệ thống gợi ý, nhờ khả năng khai thác cấu trúc quan hệ giữa người dùng và sản phẩm thông qua đồ thị hai phía (user-item bipartite graph). Một trong những mô hình tiên tiến và hiệu quả trong nhóm này là LightGCN – viết tắt của Lightweight Graph Convolutional Network, được đề xuất bởi [3].

2.8.1 Động lực phát triển LightGCN

Các mô hình mạng nơ-ron đồ thị truyền thống (Graph Convolutional Networks – GCN) khi được áp dụng vào bài toán hệ thống gợi ý thường tích hợp nhiều thành phần như biến đổi đặc trưng tuyến tính (feature transformation), hàm kích hoạt phi tuyến (non-linear activation), và cơ chế tổng hợp đa tầng (multi-layer aggregation). Mặc dù các thành phần này góp phần tăng tính biểu đạt trong các bài toán như phân loại nút hoặc dự đoán liên kết, chúng lại không thực sự cần thiết trong ngữ cảnh hệ thống gợi ý, nơi mục tiêu chính là học các biểu diễn nhúng (embedding) của người dùng và sản phẩm dựa trên cấu trúc kết nối trong đồ thị hai phía. Xuất phát từ nhận định này, LightGCN được đề xuất như một phiên bản tối giản của GCN, loại bỏ hoàn toàn các thành phần phức tạp nói trên và chỉ giữ lại thành phần cốt lõi là cơ chế lan truyền và tổng hợp thông tin từ lân cận (neighbor aggregation). Cách tiếp cận này không chỉ giúp giảm thiểu chi phí tính toán mà còn nâng cao khả năng mở rộng và tính hiệu quả của mô hình trong các hệ thống gợi ý quy mô lớn.

2.8.2 Mô hình biểu diễn trong LightGCN

Giả sử đồ thị $G = (U \cup I, E)$ gồm tập người dùng U, tập sản phẩm I, và tập cạnh E thể hiện tương tác giữa hai phía. Với mỗi nút $v \in U \cup I$, LightGCN xây dựng biểu diễn nhúng (embedding) bằng cách lan truyền qua nhiều tầng (layers) theo công thức:

$$\mathbf{e}_v^{(k)} = \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{|\mathcal{N}(v)|} \cdot \sqrt{|\mathcal{N}(u)|}} \mathbf{e}_u^{(k-1)} \tag{2.8.1}$$

Trong đó:

- $\mathbf{e}_v^{(k)}$: embedding của nút v tại tầng k,
- $\mathcal{N}(v)$: tập lân cận (neighbors) của nút v,
- $\mathbf{e}_u^{(0)}$: embedding ban đầu, được khởi tạo ngẫu nhiên và học được trong quá trình huấn luyện.

Tổng embedding cuối cùng của một nút là tổ hợp của embedding qua các tầng:

$$\mathbf{e}_v = \sum_{k=0}^{K} \alpha_k \, \mathbf{e}_v^{(k)} \tag{2.8.2}$$

Trong thực tế, hệ số α_k thường được đặt bằng nhau (ví dụ $\alpha_k = \frac{1}{K+1}$) hoặc được điều chỉnh thông qua học (learnable weights).

2.8.3 Hàm mục tiêu (Loss Function)

LightGCN được huấn luyện thông qua hàm mất mát **Bayesian Personalized Ranking (BPR)**, nhằm tối ưu sự xếp hạng giữa các item có tương tác và không tương tác:

$$\mathcal{L} = - \sum_{(u,i,j) \in \mathcal{D}} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|\Theta\|^2 \tag{2.8.3}$$

Trong đó:

- $\hat{y}_{ui} = \mathbf{e}_u^\top \mathbf{e}_i$: điểm dự đoán tương tác giữa người dùng u và item i (thông qua tích vô hướng của embedding),
- (u, i, j): bộ ba huấn luyện gồm người dùng u, item tương tác i (positive) và item không tương tác j (negative),
- λ: hệ số điều chuẩn (regularization),
- Θ: tập tham số của mô hình (các embedding của user và item).

LightGCN là một mô hình hiệu quả và tinh gọn trong nhóm Graph-based Recommender Systems. Với cách tiếp cận loại bỏ các thành phần phức tạp không cần thiết, mô hình giúp tập trung tối ưu hóa embedding từ cấu trúc đồ thị và đạt được kết quả ấn tượng trong thực tế mà vẫn giữ được khả năng mở rộng cho hệ thống lớn.

Chương 3.

PHƯƠNG PHÁP THỰC HIỆN

Chương này trình bày chi tiết quy trình thực hiện nghiên cứu, bao gồm các bước từ mô tả tập dữ liệu, tiền xử lý, phân tích và trực quan dữ liệu, đến xây dựng và triển khai các mô hình gợi ý. Dữ liệu được sử dụng trong nghiên cứu là tập *MovieLens 1M*, một bộ dữ liệu chuẩn trong lĩnh vực hệ thống gợi ý, chứa hơn một triệu lượt đánh giá từ người dùng đối với các bộ phim khác nhau. Trước tiên, dữ liệu được mô tả và tiền xử lý nhằm chuẩn hóa định dạng, loại bỏ các giá trị không hợp lệ và chuyển đổi thành cấu trúc phù hợp với các mô hình học máy.

Tiếp theo, quá trình phân tích và trực quan dữ liệu được thực hiện để khám phá đặc trưng tổng quan của tập dữ liệu, như phân phối đánh giá, tần suất tương tác của người dùng và mức độ phổ biến của các bộ phim. Những phân tích này không chỉ hỗ trợ hiểu rõ hơn về hành vi người dùng mà còn giúp định hướng lựa chọn mô hình và chiến lược huấn luyện phù hợp.

Về mặt mô hình hóa, nghiên cứu triển khai hai thuật toán gợi ý nổi bật: *Alternating Least Squares (ALS)* và *Light Graph Convolutional Network (LightGCN)*. Trong đó, ALS đại diện cho phương pháp phân rã ma trận tuyến tính truyền thống, còn LightGCN là một mô hình học sâu dựa trên đồ thị hiện đại, tập trung vào việc khai thác cấu trúc kết nối giữa user và item. Cuối cùng, một mô hình kết hợp hai thuật toán này được xây dựng nhằm tận dụng ưu điểm của cả hai hướng tiếp cận – ALS với khả năng hội tụ nhanh và LightGCN với năng lực biểu diễn mạnh mẽ từ cấu trúc đồ thị – để cải thiện chất lượng gợi ý và nâng cao độ chính xác dự đoán.

3.1 Mô tả dữ liệu

Trong nghiên cứu này, chúng tôi sử dụng tập dữ liệu MovieLens 1M[1], được phát hành bởi GroupLens Research vào tháng 2 năm 2003. Tập dữ liệu này bao gồm 1.000.209 lượt đánh giá từ 6.040 người dùng đối với khoảng 3.900 bộ phim, với thang điểm đánh giá từ 1 đến 5. MovieLens 1M được tổ chức thành ba tệp dữ liệu chính:

[1] https://grouplens.org/datasets/movielens/1m/

- `users.dat`: chứa thông tin người dùng như *ID*, giới tính, tuổi, nghề nghiệp và mã vùng.
- `movies.dat`: chứa thông tin phim bao gồm *ID*, tên phim và thể loại.
- `ratings.dat`: lưu trữ thông tin đánh giá, gồm *ID người dùng*, *ID phim*, điểm đánh giá và thời gian đánh giá.

Dữ liệu ban đầu được lưu trữ ở định dạng `.dat`, với các trường phân tách bằng ký tự `"::"`. Để sử dụng trong phân tích và xây dựng mô hình, dữ liệu cần được chuyển đổi về định dạng bảng chuẩn (`DataFrame`) và gộp (*merge*) theo các khóa chung.

3.2 Tiền xử lý dữ liệu

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

- *Chuyển đổi định dạng:* Các tệp `.dat` được đọc và chuyển đổi thành các bảng dữ liệu dạng cấu trúc (DataFrame), sử dụng thư viện `pandas` trong `Python`. Mỗi dòng dữ liệu được tách thành các cột tương ứng dựa trên ký tự phân tách.
- *Gộp dữ liệu:* Ba bảng `users`, `movies` và `ratings` được nối với nhau thông qua các khóa `UserID` và `MovieID` để tạo ra một bảng tổng hợp duy nhất chứa đầy đủ thông tin người dùng, phim và lượt đánh giá.
- *Chuẩn hóa dữ liệu:* Các trường dữ liệu dạng phân loại (categorical) như giới tính, nghề nghiệp và thể loại phim được mã hóa dưới dạng số (encoding) nếu cần thiết, nhằm phục vụ cho các mô hình học máy.
- *Lọc dữ liệu không hợp lệ:* Các bản ghi có giá trị thiếu (missing values) hoặc không hợp lệ (ví dụ: đánh giá ngoài khoảng 1-5) được loại bỏ khỏi tập dữ liệu.
- *Tạo ma trận tương tác:* Từ bảng dữ liệu tổng hợp, một ma trận tương tác người dùng - phim được xây dựng, trong đó mỗi hàng là một người dùng và mỗi cột là một bộ phim, với giá trị tương ứng là điểm đánh giá. Ma trận này là đầu vào chính cho các mô hình gợi ý như ALS và LightGCN.
- *Chia tập dữ liệu:* Tập dữ liệu được chia thành hai phần: tập huấn luyện (*training set*), tập xác thực (*validation set*) và tập kiểm thử (*test set*) theo tỷ lệ 70:20:10, nhằm đánh giá hiệu quả tổng quát của các mô hình trên dữ liệu chưa thấy.

Thông qua quá trình tiền xử lý nêu trên, dữ liệu được chuyển từ định dạng thô sang dạng chuẩn hóa, phù hợp cho việc huấn luyện các mô hình gợi ý dựa trên học máy và học sâu.

3.3 Phương pháp triển khai mô hình ALS

Trong nghiên cứu này, mô hình Alternating Least Squares (ALS) được sử dụng như một phương pháp gợi ý dựa trên phân rã ma trận, nhằm dự đoán mức độ yêu thích của người dùng đối với các sản phẩm chưa từng tương tác. Quy trình triển khai mô hình ALS trên tập dữ liệu MovieLens 1M được thực hiện qua các bước chính như sau:

Bước 1: Biểu diễn dữ liệu dưới dạng ma trận user – item

Tập dữ liệu MovieLens 1M bao gồm các đánh giá rời rạc theo thang điểm từ 1 đến 5 giữa người dùng và bộ phim. Dữ liệu được tổ chức thành một ma trận đánh giá: $\mathbf{R} \in \mathbb{R}^{m \times n}$, trong đó: m: số lượng người dùng, n: số lượng phim, R_{ui}: là điểm số mà người dùng u dành cho phim i, hoặc để trống nếu chưa đánh giá.

Bước 2: Mô hình hóa bằng phân rã ma trận

Mục tiêu của thuật toán ALS là tìm hai ma trận ẩn:

- $\mathbf{U} \in \mathbb{R}^{m \times k}$: biểu diễn đặc trưng người dùng,
- $\mathbf{V} \in \mathbb{R}^{n \times k}$: biểu diễn đặc trưng sản phẩm.

sao cho tích $\hat{R} = UV^T$ tiệm cận với ma trận đánh giá ban đầu R, trong đó k là số chiều ẩn.

Bước 3: Tối ưu hóa luân phiên

Quá trình học của ALS được thực hiện thông qua tối ưu hóa luân phiên:

$$\min_{U,V} \sum_{(u,i) \in K} \left(R_{ui} - U_u^T V_i\right)^2 + \lambda \left(\|U_u\|^2 + \|V_i\|^2\right) \tag{3.3.1}$$

Trong đó:

- K: tập các đánh giá đã biết,
- λ: hệ số điều chuẩn giúp tránh overfitting.

Quy trình tối ưu hóa bao gồm hai bước lặp lại:

1. Giữ cố định V, cập nhật từng hàng của U bằng cách giải bài toán bình phương nhỏ nhất.

2. Giữ cố định U, cập nhật từng hàng của V theo cách tương tự.

Quá trình được lặp lại cho đến khi hội tụ hoặc đạt đến số vòng lặp xác định trước.

Bước 4: Cấu hình huấn luyện mô hình trên Spark MLlib

Mô hình được triển khai bằng thư viện `pyspark.ml.recommendation.ALS`, với các tham số chính bao gồm:

- `rank`: số chiều tiềm ẩn k,
- `maxIter`: số vòng lặp tối đa,
- `regParam`: hệ số điều chuẩn λ,
- `userCol`, `itemCol`, `ratingCol`: tên các cột trong tập dữ liệu,
- `coldStartStrategy="drop"`: loại bỏ các dự đoán không có đủ thông tin trong tập huấn luyện.

Bước 5: Dự đoán và đánh giá

Sau khi mô hình ALS được huấn luyện trên tập huấn luyện, quá trình đánh giá hiệu quả mô hình được thực hiện trên tập kiểm thử. Việc đánh giá được tiến hành dựa trên hai nhóm tiêu chí: (1) tiêu chí hồi quy và (2) tiêu chí đánh giá gợi ý top-N.

(1) Các tiêu chí hồi quy (dự đoán điểm đánh giá):

Root Mean Squared Error (RMSE): Đo độ lệch trung bình bình phương giữa điểm dự đoán và điểm thực tế. RMSE càng nhỏ, mô hình càng chính xác.

$$\text{RMSE} = \sqrt{\frac{1}{|T|} \sum_{(u,i) \in T} \left(R_{ui} - \hat{R}_{ui} \right)^2} \tag{3.3.2}$$

Mean Absolute Error (MAE): Đo độ lệch tuyệt đối trung bình giữa dự đoán và thực tế. MAE phản ánh sai số một cách tuyến tính.

$$\text{MAE} = \frac{1}{|T|} \sum_{(u,i) \in T} \left| R_{ui} - \hat{R}_{ui} \right| \tag{3.3.3}$$

Hệ số xác định (R^2): Đo lường mức độ giải thích phương sai của biến phụ thuộc bởi mô hình. Giá trị $R^2 = 1$ biểu thị mô hình dự đoán hoàn hảo, còn $R^2 = 0$ cho thấy mô hình không giải thích được phương sai.

$$R^2 = 1 - \frac{\sum_{(u,i)\in T} \left(R_{ui} - \hat{R}_{ui}\right)^2}{\sum_{(u,i)\in T} \left(R_{ui} - \bar{R}\right)^2} \tag{3.3.4}$$

trong đó $\bar{R}$ là trung bình các giá trị thực R_{ui}.

(2) Các tiêu chí đánh giá gợi ý Top-N (Recommendation Metrics):
Trong trường hợp đánh giá khả năng gợi ý danh sách phim cho từng người dùng, hai tiêu chí phổ biến được sử dụng là:

Precision@N: Tỉ lệ item được gợi ý nằm trong danh sách đúng thực sự người dùng quan tâm.

$$\text{Precision@N} = \frac{|\text{Items đúng trong Top-N}|}{N} \tag{3.3.5}$$

Recall@N: Tỉ lệ item đúng được gợi ý trên tổng số item mà người dùng thực sự quan tâm.

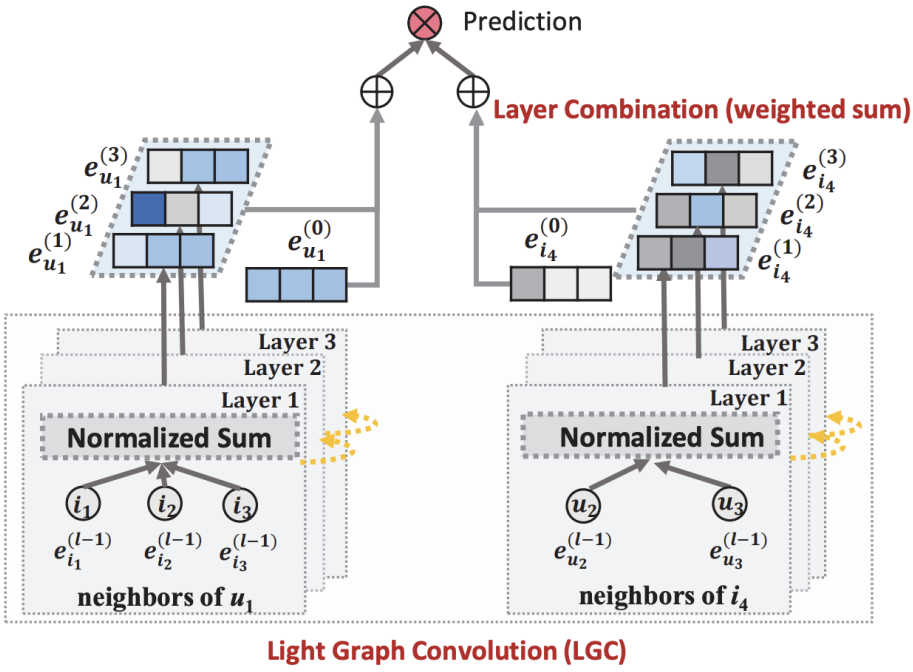
$$\text{Recall@N} = \frac{|\text{Items đúng trong Top-N}|}{|\text{Items đúng}|} \tag{3.3.6}$$

Bước 6: Sinh gợi ý Top-N

Mô hình ALS sau huấn luyện được sử dụng để sinh danh sách Top-N sản phẩm (phim) được gợi ý cho từng người dùng bằng cách lựa chọn những item có $\hat{R}_{ui}$ cao nhất mà người dùng chưa từng đánh giá.

3.4 Mô hình LightGCN

Trong phần này, chúng tôi trước hết giới thiệu mô hình Light Graph Convolutional Network (LightGCN) mà chúng tôi thiết kế, như được minh họa trong Hình 2. Tiếp theo, chúng tôi trình bày phân tích chuyên sâu để làm rõ cơ sở hợp lý đằng sau thiết kế đơn giản của LightGCN. Cuối cùng, chúng tôi mô tả cách huấn luyện mô hình trong bối cảnh hệ thống gợi ý.



Hình 3.1: Minh họa kiến trúc mô hình LightGCN [3]

3.4.1 Light Graph Convolution (LGC)

Trong LightGCN, chúng tôi sử dụng bộ tổng hợp trung bình có trọng số đơn giản và loại bỏ hoàn toàn việc biến đổi đặc trưng và hàm kích hoạt phi tuyến. Phép lan truyền trên đồ thị (hay còn gọi là phép truyền tín hiệu [**?**]) trong LightGCN được định nghĩa như sau:

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)}; \quad \mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)} \tag{3.4.1}$$

Thuật ngữ chuẩn hóa đối xứng $\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}$ được kế thừa từ thiết kế của GCN tiêu chuẩn, giúp tránh việc độ lớn của các vector nhúng (embedding) tăng lên không kiểm soát khi lan truyền qua nhiều tầng. Một số lựa chọn chuẩn hóa khác như chuẩn L_1 cũng có thể được áp dụng, tuy nhiên trên thực nghiệm, chuẩn hóa đối xứng này cho kết quả tốt hơn.

Cũng cần lưu ý rằng trong LightGCN, chúng tôi chỉ tổng hợp từ các nút lân cận được kết nối và không bao gồm chính nút đó (tức là không có self-connection). Điều này khác biệt với hầu hết các phép lan truyền đồ thị truyền thống [17], vốn mở rộng phạm vi lân cận và phải xử lý riêng biệt kết nối với chính mình. Phép kết hợp tầng (layer combination), sẽ được giới thiệu trong mục sau, có thể đạt được hiệu quả tương đương như self-connection. Do đó, trong LightGCN không cần tích hợp self-connections.

3.4.2 Layer Combination and Model Prediction

Trong LightGCN, các tham số duy nhất cần học của mô hình là các vector nhúng ở tầng thứ 0, tức là $\mathbf{e}_u^{(0)}$ cho tất cả người dùng và $\mathbf{e}_i^{(0)}$ cho tất cả các item. Khi các vector nhúng này đã được khởi tạo, các embedding ở các tầng cao hơn có thể được tính toán thông qua LGC (Light Graph Convolution) như đã định nghĩa trong phương trình (3.4.1).

Sau khi lan truyền qua K tầng LGC, các vector nhúng tại mỗi tầng sẽ được kết hợp lại để tạo ra biểu diễn cuối cùng của người dùng (hoặc item) như sau:

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}; \quad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)} \tag{3.4.2}$$

Trong đó, $\alpha_k \geq 0$ biểu thị mức độ quan trọng của embedding ở tầng thứ k trong việc tạo nên biểu diễn cuối cùng. Trọng số α_k có thể được xem là một siêu tham số (hyperparameter) do người dùng xác định, hoặc là một tham số mô hình (ví dụ: đầu ra của mạng attention để tối ưu tự động.

Trong các thực nghiệm, việc gán đều trọng số $\alpha_k = \frac{1}{K+1}$ cho tất cả các tầng cho thấy hiệu năng tổng thể tốt. Do đó, chúng tôi không thiết kế thêm thành phần đặc biệt để tối ưu α_k nhằm giữ cho LightGCN đơn giản và không phức tạp hóa mô hình một cách không cần thiết.

Lý do chúng tôi thực hiện kết hợp tầng để thu được biểu diễn cuối cùng bao gồm ba điểm chính: (1) Khi số tầng tăng, embedding có thể trở nên quá mượt, nên chỉ sử dụng tầng cuối là không tối ưu; (2) Các tầng khác nhau mang ngữ nghĩa khác nhau: tầng thứ nhất làm mượt giữa các nút có tương tác trực tiếp; tầng thứ hai làm mượt giữa các nút có chung lân cận; các tầng cao hơn bắt giữ các mối liên kết bậc cao hơn. Việc kết hợp các tầng này giúp biểu diễn trở nên toàn diện hơn; (3) Kết hợp các embedding từ nhiều tầng bằng phép cộng có trọng số giúp mô phỏng hiệu ứng của self-connection trong GCN, một kỹ thuật quan trọng.

Dự đoán của mô hình được xác định là tích vô hướng giữa biểu diễn cuối cùng của người dùng và item:

$$\hat{y}_{ui} = \mathbf{e}_u^\top \mathbf{e}_i \tag{3.4.3}$$

Giá trị $\hat{y}_{ui}$ này được sử dụng như là điểm xếp hạng để sinh ra danh sách gợi ý.

3.4.3 Huấn luyện mô hình

Các tham số cần huấn luyện của LightGCN chỉ là các vector nhúng ở tầng thứ 0, tức là:$\Theta = \{E^{(0)}\}$. Nói cách khác, độ phức tạp của mô hình tương đương

với phương pháp phân rã ma trận truyền thống (Matrix Factorization - MF).

Chúng tôi sử dụng hàm mất mát *Bayesian Personalized Ranking* (BPR)là một hàm mất mát cặp (*pairwise loss*), nhằm khuyến khích mô hình dự đoán điểm số cho một tương tác quan sát được (positive) cao hơn các tương tác chưa quan sát (negative).

Hàm mất mát BPR được định nghĩa như sau:

$$\mathcal{L}_{\text{BPR}} = -\sum_{u=1}^{M} \sum_{i \in \mathcal{N}_u} \sum_{j \notin \mathcal{N}_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|E^{(0)}\|_2^2 \tag{3.4.4}$$

Trong đó:

- $\hat{y}_{ui}$ và $\hat{y}_{uj}$ là điểm dự đoán của mô hình giữa người dùng u với item đã tương tác i và item chưa từng tương tác j.
- σ là hàm sigmoid.
- λ là hệ số điều chuẩn (*regularization coefficient*).
- $\|E^{(0)}\|_2^2$ là chuẩn L2 của embedding khởi tạo.

3.5 Mô hình kết hợp ALS + LightGCN

Dựa trên quan sát thực nghiệm và lý thuyết, mô hình Alternating Least Squares (ALS) và Light Graph Convolutional Network (LightGCN), chúng tôi đề xuất một phương pháp kết hợp hai mô hình, việc kết hợp hai phương pháp này nhằm tận dụng ưu điểm của cả hai: ALS cung cấp thông tin trực tiếp từ đánh giá số, LightGCN cung cấp thông tin cấu trúc và các mối quan hệ bậc cao giữa các nút trong đồ thị. Cụ thể với kiến trúc kết hợp như sau:

- $\hat{y}_{ui}^{\text{ALS}}$: điểm dự đoán từ mô hình ALS cho người dùng u và item i,
- $\hat{y}_{ui}^{\text{GCN}}$: điểm dự đoán từ mô hình LightGCN,
- $\hat{y}_{ui}$: điểm dự đoán cuối cùng sau khi kết hợp hai mô hình.

Phương pháp kết hợp tuyến tính giữa hai mô hình được định nghĩa như sau:

$$\hat{y}_{ui} = \alpha \cdot \hat{y}_{ui}^{\text{ALS}} + (1 - \alpha) \cdot \hat{y}_{ui}^{\text{GCN}}, \quad \alpha \in [0, 1] \tag{3.5.1}$$

Trong đó, α là hệ số điều chỉnh mức ảnh hưởng giữa hai mô hình, có thể được lựa chọn thủ công (tuned as a hyperparameter) hoặc được học tự động từ tập validation. Ngoài ra, với dự đoán từ từng mô hình:

ALS: sử dụng tích vô hướng giữa embedding người dùng u_u và item v_i:

$$\hat{y}_{ui}^{\text{ALS}} = u_u^\top v_i \tag{3.5.2}$$

LightGCN: sau khi lan truyền qua K tầng và kết hợp embedding:

$$\hat{y}_{ui}^{\text{GCN}} = \left(\sum_{k=0}^{K} \alpha_k e_u^{(k)} \right)^\top \left(\sum_{k=0}^{K} \alpha_k e_i^{(k)} \right) \tag{3.5.3}$$

trong đó $\alpha_k = \frac{1}{K+1}$ là trọng số cho mỗi tầng lan truyền.

Để tận dụng thế mạnh riêng của từng mô hình, chúng được huấn luyện riêng biệt và kết quả dự đoán được hợp nhất thông qua hàm kết hợp tuyến tính được định nghĩa tại phương trình (3.5.1).

Chương 4.

KẾT QUẢ THỰC NGHIỆM

Chương này trình bày quá trình thực nghiệm huấn luyện mô hình gợi ý kết hợp trên tập dữ liệu MovieLens 1M. Các kết quả thu được từ mô hình ALS, LightGCN và mô hình lai (hybrid) được so sánh theo các tiêu chí đánh giá hiệu năng như Precision@K, Recall@K, RMSE, MAE và thời gian xử lý. Ngoài ra, chương cũng phân tích khả năng mở rộng cũng như các ưu và nhược điểm của từng mô hình.

4.1 Kết quả huấn luyện mô hình

Quá trình huấn luyện được thực hiện trên tập dữ liệu MovieLens 1M với 1 triệu đánh giá từ khoảng 6.000 người dùng và gần 4.000 bộ phim. Các mô hình được triển khai như sau:

4.1.1 Mô hình ALS

Thuật toán Alternating Least Squares được triển khai và đánh giá với các cấu hình khác nhau của siêu tham số, bao gồm: số lượng *latent factors* (`rank`), số vòng lặp huấn luyện (`maxIter`) và hệ số điều chuẩn (`regParam`).

Các mô hình được đánh giá dựa trên 5 tiêu chí: `Precision@10`, `Recall@10`, `RMSE`, `MAE` và $\texttt{R}^2$, nhằm phản ánh độ chính xác gợi ý, khả năng bao phủ, độ sai số dự đoán, và mức độ giải thích phương sai dữ liệu. Được thể hiện thông qua bảng số liệu kết quả (Bảng 4.1) như sau:

Bảng 4.1: Kết quả đánh giá mô hình ALS với các cấu hình siêu tham số khác nhau.

Rank	MaxIter	RegParam	Precision	Recall	RMSE	MAE	R^2
5	5	0.01	0.8751	0.3869	0.8845	0.6992	0.3691
5	5	0.10	0.9244	0.2099	0.9110	0.7391	0.3306
5	5	0.50	0.9324	0.0817	1.0229	0.8460	0.1561
5	10	0.01	0.8768	0.4108	0.8722	0.6876	0.3864
5	10	0.10	0.9082	0.2930	0.8819	0.7079	0.3727
5	10	0.50	0.9347	0.0655	1.0405	0.8632	0.1268
5	15	0.01	0.8768	0.4220	0.8687	0.6839	0.3914
5	15	0.10	0.9013	0.3235	0.8751	0.6997	0.3824
5	15	0.50	0.9350	0.0649	1.0413	0.8639	0.1255
10	5	0.01	0.8645	0.4393	0.8862	0.6940	0.3666
10	5	0.10	0.9273	0.2212	0.9022	0.7311	0.3435
10	5	0.50	0.9328	0.0805	1.0242	0.8472	0.1540
10	10	0.01	0.8685	0.4583	0.8763	0.6849	0.3806
10	10	0.10	0.9132	0.2988	0.8741	0.7012	0.3838
10	10	0.50	0.9346	0.0655	1.0406	0.8632	0.1267
10	15	0.01	0.8692	0.4640	0.8715	0.6808	0.3874
10	**15**	**0.10**	**0.9083**	**0.3291**	**0.8665**	**0.6926**	**0.3944**
10	15	0.50	0.9350	0.0649	1.0413	0.8639	0.1255

A. Ảnh hưởng của số vòng lặp (maxIter)

Kết quả cho thấy rằng việc tăng số vòng lặp huấn luyện từ 5 lên 15 có xu hướng cải thiện các chỉ số **RMSE**, **MAE**, và **Recall**, đặc biệt khi kết hợp với hệ số điều chuẩn nhỏ (ví dụ, `regParam = 0.01`). Cụ thể, với cấu hình `rank = 10`, `regParam = 0.01`, ta có:

- `maxIter = 5`: Recall = **0.4393**, RMSE = **0.8862**, MAE = **0.6940**
- `maxIter = 15`: Recall = **0.4640**, RMSE = **0.8715**, MAE = **0.6808**

Điều này cho thấy mô hình học được biểu diễn tốt hơn khi được huấn luyện lâu hơn, giúp tăng khả năng bao phủ các mục liên quan trong top-K gợi ý.

B. Ảnh hưởng của hệ số điều chuẩn (regParam)

Khi `regParam` tăng cao (ví dụ `0.5`), mô hình trở nên **quá điều chuẩn** (*over-regularized*), dẫn đến **giảm Recall rõ rệt** nhưng lại làm **tăng nhẹ Precision**.

Cụ thể, với cấu hình `rank = 5`, `maxIter = 15`:

- `regParam = 0.01`: Recall = **0.4220**

- `regParam = 0.5`: Recall = **0.0649**

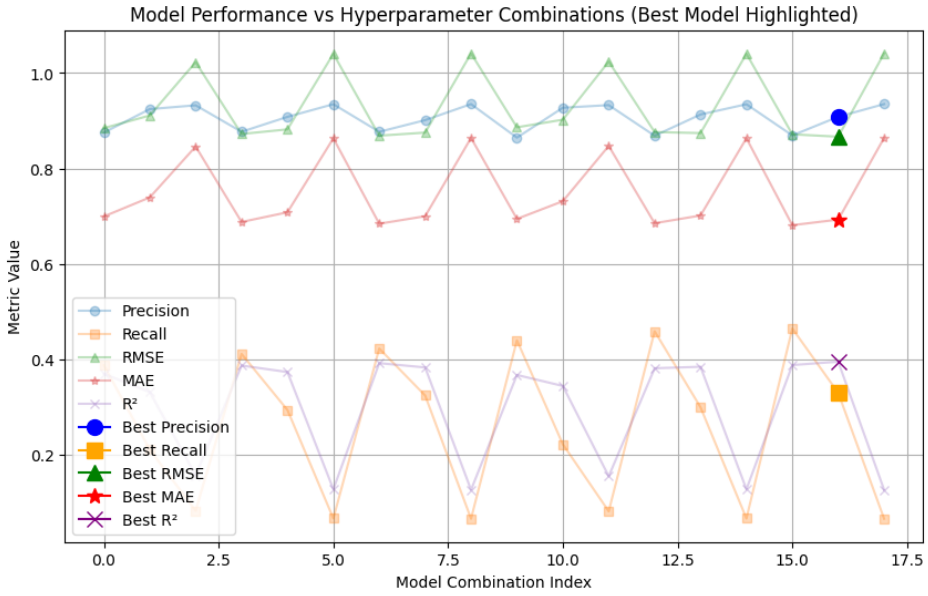
Tuy nhiên, Precision trong các trường hợp `regParam` lớn vẫn duy trì ở mức cao ($\approx$ **0.935**), cho thấy mô hình có xu hướng **ưu tiên các mục có xác suất cao**, nhưng làm mất đi **khả năng bao phủ rộng** các mục liên quan trong danh sách gợi ý.

C. Ảnh hưởng của số lượng yếu tố tiềm ẩn (rank)

So sánh giữa `rank = 5` và `rank = 10` cho thấy mô hình với `rank = 10` có khả năng học biểu diễn tốt hơn, đặc biệt trong điều kiện `regParam` nhỏ và `maxIter` cao. Ví dụ:

- Với `regParam = 0.01`, `maxIter = 15`:
  - `rank = 5`: RMSE = **0.8687**, R^2 = **0.3914**
  - `rank = 10`: RMSE = **0.8715**, R^2 = **0.3874**

Sự chênh lệch nhỏ cho thấy hiệu quả của việc tăng số chiều *embedding* là không tuyệt đối, và có thể phụ thuộc vào độ phức tạp của dữ liệu.



Hình 4.1: Biểu đồ thể hiện kết quả thực nghiệm đánh giá thuật toán ALS.

Qua quá trình thực nghiệm với nhiều cấu hình khác nhau của mô hình ALS, một trong những thiết lập nổi bật thể hiện được sự cân bằng tối ưu giữa *Precision* và *Recall* là: `rank = 10`, `maxIter = 15`, và `regParam = 0.1`. Cấu hình này đạt *Precision* = **0.9083** và *Recall* = **0.3291**, trong khi *RMSE* và *MAE* lần lượt là **0.8665** và **0.6926**, với hệ số xác định R^2 = **0.3944**. Những kết quả này cho

thấy mô hình không chỉ có độ chính xác gợi ý cao mà còn duy trì được khả năng bao phủ tốt và giải thích đáng kể phương sai trong dữ liệu người dùng.

Ngược lại, cấu hình `rank = 10`, `maxIter = 15`, và `regParam = 0.5` mặc dù đạt *Precision* cao nhất là **0.9350**, nhưng lại đánh đổi quá lớn về *Recall* (**0.0649**) và R^2 (**0.1255**). Điều này phản ánh hiện tượng *quá điều chuẩn* (*over-regularization*), khiến mô hình đánh mất khả năng khai thác đầy đủ các thông tin quan trọng trong dữ liệu huấn luyện.

Từ các kết quả thực nghiệm trên, có thể rút ra các kết luận sau:

- Tăng số vòng lặp huấn luyện (`maxIter`) giúp cải thiện đáng kể *Recall* và độ chính xác tổng thể, đặc biệt khi kết hợp với các giá trị điều chuẩn phù hợp.
- Giá trị điều chuẩn thấp (`regParam` ≤ 0.1) cho phép mô hình học được các đặc trưng biểu diễn phong phú hơn mà không bị mất tính khái quát do ràng buộc quá chặt.
- `Rank = 10` được xem là giá trị tối ưu trong phạm vi thử nghiệm, vì nó đạt được sự cân bằng hiệu quả giữa độ chính xác, khả năng bao phủ, và chi phí tính toán.

Như vậy, cấu hình `rank = 10`, `maxIter = 15`, `regParam = 0.1` có thể được xem là lựa chọn khuyến nghị cho mô hình ALS trong hệ thống gợi ý dựa trên dữ liệu *MovieLens 1M*.

4.1.2 Mô hình LightGCN

Để đánh giá hiệu quả của mô hình LightGCN, các thí nghiệm được thực hiện trên tập dữ liệu gồm 302.000 mẫu huấn luyện, 302.000 mẫu kiểm tra và 302.000 mẫu kiểm định. Mô hình được huấn luyện trong 20 epoch sử dụng thuật toán tối ưu Adam với tốc độ học (learning rate) là 0.001 và không sử dụng điều chuẩn trọng số (weight decay = 0).

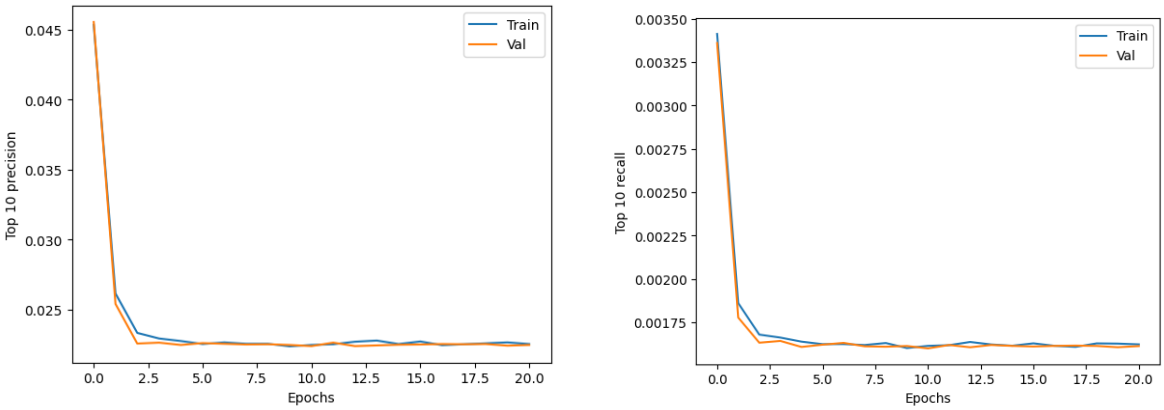
Trong suốt quá trình huấn luyện, hàm mất mát *BPR* (Bayesian Personalized Ranking) giảm đều và hội tụ về mức trung bình khoảng **0.002708** sau vài *epoch* đầu, cho thấy mô hình học ổn định. Đồng thời, hàm mất mát điều chuẩn (*regularization loss*) tiệm cận về 0, cho thấy mô hình không gặp hiện tượng quá điều chỉnh.

Về các chỉ số đánh giá *Top-K*, *Precision* và *Recall* được ghi nhận như sau:

- **Precision@K** (trên tập huấn luyện) dao động từ `0.0453` ở *epoch* đầu, giảm dần về khoảng `0.0225` từ *epoch* 5 trở đi và duy trì ổn định cho đến *epoch* 20.
- **Recall@K** giảm nhẹ theo thời gian từ `0.0034` về khoảng `0.0016`, cho thấy khả năng bao phủ gợi ý có cải thiện ban đầu và sau đó duy trì ổn định.

Trên tập kiểm định (*validation set*), các chỉ số cũng cho thấy sự hội tụ:

- **Precision@K** ổn định quanh mức `0.0224-0.0226` từ *epoch* 4 trở đi.
- **Recall@K** dao động nhẹ quanh `0.00160-0.00162`.



(a) Biểu đồ top K Precision (b) Biểu đồ top K recall

Hình 4.2: Biểu đồ kết quả thực nghiệm đánh giá thuật toán LightGCN.

Kết quả thực nghiệm chỉ ra rằng mô hình LightGCN với số lượng epoch hợp lý (≥ 5) có khả năng học hiệu quả và đạt sự ổn định sớm trong quá trình huấn luyện. Dù các chỉ số Precision và Recall không tăng mạnh sau các epoch đầu, mô hình vẫn duy trì được hiệu năng gợi ý ổn định, đồng thời tránh được hiện tượng quá khớp nhờ regularization loss gần bằng 0. Điều này cho thấy LightGCN phù hợp cho các hệ thống gợi ý lớn với dữ liệu rời rạc, đặc biệt khi ưu tiên tính đơn giản và khả năng mở rộng.

4.1.3 Mô hình kết hợp ALS + LightGCN

Quá trình đánh giá mô hình được thực hiện với các tham số được điều chỉnh bao gồm:

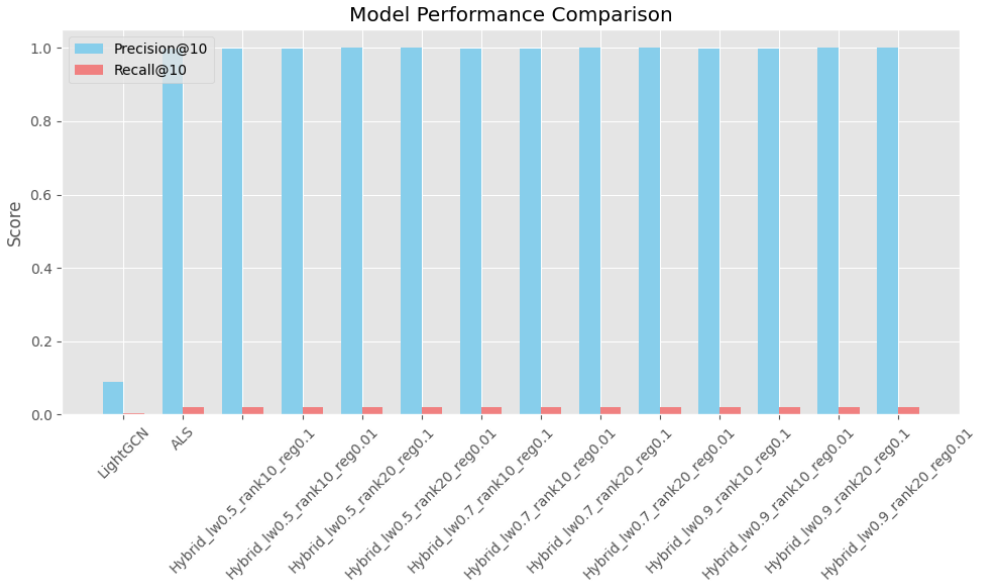
- Trọng số LightGCN $\alpha \in \{0.5, 0.7, 0.9\}$, $embedding\_size = 64$, $num\_layers = 5$.

- Hạng ma trận ALS *rank* $\in \{10, 20\}$.
- Hệ số điều chuẩn ALS *regParam* $\in \{0.01, 0.1\}$.

Khi áp dụng kết hợp, tất cả các cấu hình đều cho thấy mức `Precision@10` vượt trội, trong đó một số cấu hình thậm chí đạt độ chính xác tuyệt đối `Precision@10 = 1.0000`. Tuy nhiên, `Recall@10` không cải thiện so với mô hình ALS đơn lẻ, giữ ổn định ở mức `0.0200` trong tất cả các cấu hình (Bảng 4.2).

Bảng 4.2: Kết quả với các cấu hình Hybrid khác nhau

Weight	Rank	regParam	Precision@10	Recall@10
0.5	10	0.1	0.9985	0.0200
0.5	10	0.01	0.9995	0.0200
0.5	20	0.1	1.0000	0.0200
0.5	20	0.01	1.0000	0.0200
0.7	10	0.1	0.9985	0.0200
0.7	10	0.01	0.9995	0.0200
0.7	20	0.1	1.0000	0.0200
0.7	20	0.01	1.0000	0.0200
0.9	10	0.1	0.9985	0.0200
0.9	10	0.01	0.9995	0.0200
0.9	20	0.1	1.0000	0.0200
0.9	20	0.01	1.0000	0.0200



Hình 4.3: Biểu đồ đánh giá mô hình thực nghiệm kết hợp.

Dựa trên kết quả thực nghiệm, cấu hình kết hợp tối ưu giữa mô hình `ALS` và `LightGCN` được xác định với trọng số LightGCN bằng `0.5`, hạng ma trận ALS

bằng 20 và hệ số điều chuẩn là 0.1. Cấu hình này đạt độ chính xác tuyệt đối ở mức Precision@10 = 1.0000, đồng thời đạt Recall@10 = 0.0200, cho thấy mô hình có khả năng học rất hiệu quả các mục tiêu ưu tiên hàng đầu trong danh sách gợi ý top-10. Tuy nhiên, phạm vi bao phủ vẫn còn hạn chế, thể hiện ở mức Recall không thay đổi so với mô hình ALS đơn lẻ.

Ngoài ra, các cấu hình kết hợp khác với trọng số LightGCN lần lượt là 0.7 và 0.9 khi kết hợp với ALS rank = 20 cũng cho kết quả Precision tương đương, nhưng không mang lại sự cải thiện rõ rệt về Recall. Điều này cho thấy trong bối cảnh dữ liệu hiện tại, ALS đóng vai trò quyết định trong việc tối ưu độ chính xác của hệ thống gợi ý. Trong khi đó, LightGCN chưa phát huy được tiềm năng đáng kể, có thể do quy mô nhỏ của tập người dùng hoặc mạng đồ thị xây dựng từ dữ liệu chưa đủ thông tin liên kết để khai thác hiệu quả đặc trưng cấu trúc.

Kết quả thực nghiệm cho thấy mô hình kết hợp ALS và LightGCN là một hướng đi triển vọng, giúp duy trì hiệu suất cao của ALS đồng thời mở ra tiềm năng mở rộng mô hình với các đặc trưng từ LightGCN. Mặc dù Recall chưa được cải thiện, nhưng việc giữ ổn định Precision cao cho thấy mô hình có thể phù hợp trong các bài toán ưu tiên độ chính xác cao ở top-k gợi ý.

4.2 So sánh kết quả thực nghiệm

Kết quả thực nghiệm cho thấy sự khác biệt rõ rệt về hiệu năng giữa ba mô hình. Mô hình ALS thể hiện hiệu suất rất cao với Precision@10 đạt 0.9083 và Recall@10 là 0.3291. Điều này phản ánh khả năng xác định chính xác các mục tiêu gợi ý nằm trong top ưu tiên, mặc dù mức độ bao phủ vẫn còn hạn chế.

Ngược lại, mô hình LightGCN có Precision@10 chỉ đạt 0.0890 và Recall@10 là 0.00756, cho thấy hiệu quả khuyến nghị còn thấp. Nguyên nhân có thể đến từ đặc điểm cấu trúc đồ thị chưa đủ thông tin liên kết hoặc quy mô nhỏ của tập người dùng khiến mô hình khó khai thác quan hệ lân cận hiệu quả.

Đáng chú ý, mô hình kết hợp ALS + LightGCN đã đạt được Precision@10 lên đến 1.0000, đồng thời giữ nguyên Recall@10 = 0.0200 trong nhiều cấu hình khác nhau (đặc biệt với trọng số LightGCN = 0.5 và ALS rank = 20). Điều này cho thấy việc kết hợp giúp duy trì hoặc cải thiện độ chính xác mà không làm suy giảm khả năng bao phủ, mặc dù vẫn chưa mở rộng được tập dự đoán. Ngoài ra, LightGCN tuy chưa nâng cao Recall nhưng vẫn đóng vai trò hỗ trợ ổn định hóa quá trình học.

Bảng 4.3: Bảng tổng hợp so sánh hiệu suất các mô hình

Mô hình	**Precision@10**	**Recall@10**
ALS	0.9083	0.3291
LightGCN	0.0890	0.00756
Hybrid (ALS + LightGCN)	1.0000	0.0200

Từ bảng trên có thể kết luận rằng mô hình kết hợp ALS + LightGCN không chỉ duy trì độ chính xác cao như ALS mà còn giữ ổn định phạm vi bao phủ, nhờ đó có tiềm năng ứng dụng tốt hơn trong các hệ thống khuyến nghị thực tế. Trong khi đó, LightGCN đơn lẻ cần được cải thiện hoặc mở rộng dữ liệu để phát huy hiệu quả rõ rệt hơn.

Chương 5.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Trong đề tài này, một hệ thống khuyến nghị phim dựa trên dữ liệu người dùng từ *MovieLens* đã được xây dựng và đánh giá bằng cách ứng dụng các mô hình học máy hiện đại, bao gồm ALS, LightGCN, và một mô hình kết hợp ALS + LightGCN. Hệ thống được triển khai trên nền tảng *Apache Spark* nhằm đảm bảo khả năng xử lý dữ liệu lớn và mở rộng.

Các thực nghiệm cho thấy mô hình ALS đạt hiệu suất cao về độ chính xác với `Precision@10 = 0.9083` và `Recall@10 = 0.3291`, trong khi LightGCN có độ chính xác thấp hơn đáng kể. Tuy nhiên, khi kết hợp hai mô hình này, hệ thống đã đạt `Precision@10 = 1.0000` mà vẫn giữ nguyên Recall, chứng minh tính hiệu quả của cách tiếp cận kết hợp.

5.2 Những đóng góp chính của nghiên cứu

- Đề xuất một phương pháp kết hợp mô hình giữa `ALS` và `LightGCN` nhằm tận dụng sức mạnh của cả hai: ALS trong việc học quan hệ tuyến tính và LightGCN trong việc khai thác thông tin cấu trúc từ đồ thị.
- Triển khai hệ thống trên `Apache Spark`, cho phép xử lý hiệu quả với dữ liệu lớn.
- Thực hiện phân tích đánh giá chi tiết qua các chỉ số chuẩn như `Precision`, `Recall`, `RMSE`, `MAE` và $\texttt{R}^2$ để so sánh hiệu suất giữa các mô hình.
- Cung cấp một bộ thông số tối ưu cho mô hình kết hợp: *trọng số LightGCN* $= 0.5$, *rank ALS* $= 20$, *regParam* $= 0.1$.

5.3 Hạn chế

- Tập dữ liệu sử dụng có quy mô giới hạn, đặc biệt là số lượng người dùng còn nhỏ, dẫn đến mô hình `LightGCN` chưa khai thác triệt để thông tin liên kết đồ thị.

- Giá trị `Recall` mặc dù được duy trì nhưng vẫn còn rất thấp, cho thấy khả năng bao phủ của hệ thống còn hạn chế.
- Mô hình kết hợp mới chỉ dừng ở mức tuyến tính đơn giản (kết hợp điểm dự đoán bằng trọng số), chưa sử dụng phương pháp học sâu để tối ưu việc hợp nhất kết quả.

5.4 Định hướng phát triển trong tương lai

- Mở rộng quy mô dữ liệu, áp dụng trên tập người dùng lớn hơn và thêm các đặc trưng bổ sung như thời gian, thể loại phim, và hành vi tương tác nâng cao (click, rating time...).
- Cải tiến mô hình kết hợp bằng cách sử dụng mạng học sâu (như MLP hoặc attention-based fusion) để tự động học trọng số kết hợp giữa các mô hình thành phần.
- Nghiên cứu tích hợp thêm các kỹ thuật học tăng cường (reinforcement learning) để cập nhật mô hình theo thời gian thực, phù hợp với môi trường thay đổi liên tục.
- Triển khai hệ thống trong môi trường thực tế, kết hợp với giao diện người dùng để đánh giá trực tiếp trải nghiệm người dùng và phản hồi từ phía sử dụng.

Tóm lại, nghiên cứu đã đề xuất và triển khai thành công một hệ thống khuyến nghị phim hiệu quả bằng cách kết hợp hai kỹ thuật bổ trợ nhau: ALS và LightGCN. Thông qua việc thực nghiệm trên tập dữ liệu MovieLens và đánh giá trên các chỉ số chuẩn, kết quả thu được đã khẳng định tiềm năng của mô hình kết hợp trong việc nâng cao độ chính xác của hệ thống khuyến nghị. Mặc dù vẫn còn tồn tại một số hạn chế nhất định như khả năng bao phủ thấp và tính khái quát chưa cao, hệ thống đã đặt nền móng vững chắc cho những hướng cải tiến trong tương lai. Những đóng góp của nghiên cứu không chỉ dừng lại ở kết quả thực nghiệm, mà còn mở ra cơ hội phát triển các mô hình lai ghép có khả năng mở rộng và thích ứng với dữ liệu quy mô lớn hơn trong thực tiễn. Đây là bước tiến quan trọng trong hành trình hướng đến các hệ thống khuyến nghị thông minh, linh hoạt và chính xác hơn.

TÀI LIỆU THAM KHẢO

[1] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on apache spark," *International Journal of Data Science and Analytics*, vol. 1, no. 3, pp. 145–164, 2016.

[2] E. Shaikh, I. Mohiuddin, Y. Alufaisan, and I. Nahvi, "Apache spark: A big data processing engine," pp. 1–6, 2019.

[3] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 639–648, 2020.

[4] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender systems handbook*, pp. 1–35, Springer, 2010.

[5] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *Algorithmic Aspects in Information and Management: 4th International Conference, AAIM 2008, Shanghai, China, June 23-25, 2008. Proceedings 4*, pp. 337–348, Springer, 2008.

[6] A. Gandomi and M. Haider, "Beyond the hype: Big data concepts, methods, and analytics," *International journal of information management*, vol. 35, no. 2, pp. 137–144, 2015.

[7] D. Laney *et al.*, "3d data management: Controlling data volume, velocity and variety," *META group research note*, vol. 6, no. 70, p. 1, 2001.

[8] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issues," *Information systems*, vol. 47, pp. 98–115, 2015.

[9] P. Zikopoulos and C. Eaton, *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011.

[10] M. Schroeck, R. Shockley, J. Smart, D. Romero-Morales, and P. Tufano, "Analytics: The real-world use of big data," *IBM Global Business Services*, vol. 12, no. 2012, pp. 1–20, 2012.

[11] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The adaptive web: methods and strategies of web personalization*, pp. 291–324, Springer, 2007.

[12] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[13] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The adaptive web: methods and strategies of web personalization*, pp. 325–341, Springer, 2007.

[14] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, pp. 331–370, 2002.

[15] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, *et al.*, "Apache spark: a unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[16] J. Ellingwood, "Hadoop, storm, samza, spark, and flink: Big data frameworks compared," *Digital Ocean*, 2016.

[17] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 165–174, 2019.