HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

TRẦN HẢI ANH

NGHIÊN CỨU VÀ ỨNG DỤNG KỸ THUẬT HỌC SÂU CHO HỆ TƯ VẤN

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(*Theo định hướng ứng dụng*)

HÀ NỘI - 2024

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

TRẦN HẢI ANH

NGHIÊN CỨU VÀ ỨNG DỤNG KỸ THUẬT HỌC SÂU CHO HỆ TƯ VẤN

Chuyên ngành: **KHOA HỌC MÁY TÍNH**

Mã số: **8.48.01.018**

ĐỀ ÁN TỐT NGHIỆP THẠC SĨ KỸ THUẬT

(***Theo định hướng ứng dụng***)

NGƯỜI HƯỚNG DẪN KHOA HỌC

TS. NGUYỄN DUY PHƯƠNG

HÀ NỘI - 2024

LỜI CAM ĐOAN

Tôi xin cam đoan đề án với tiêu đề "Nghiên cứu và ứng dụng kỹ thuật học sâu cho hệ tư vấn" hoàn toàn là kết quả tìm hiểu nghiên cứu của riêng cá nhân tôi. Trong quá trình thực hiện đề án tốt nghiệp, tôi đã thực hiện nghiêm túc các quy tắc đạo đức nghiên cứu; các kết quả trình bày ở trong đề án là sản phẩm nghiên cứu khảo sát của riêng tôi; tất cả các tham khảo sử dụng trong đề án đều được trích dẫn tường minh, theo đúng quy định.

Tôi xin chịu hoàn toàn trách nghiệm về tính trung thực của đề án.

Hà Nội, ngày 20 tháng 3 năm 2024

Học viên

Trần Hải Anh

LỜI CẢM ƠN

Tôi xin chân thành cảm ơn Khoa Sau đại học, Học viện Công nghệ Bưu chính viễn thông đã tạo mọi điều kiện thuận lợi cho tôi hoàn thành đề án tốt nghiệp thạc sĩ nghành Khoa học máy tính.

Tôi xin gửi lời cảm ơn chân thành tới tập thể các Thầy Cô công tác tại Học viện Công nghệ Bưu chính viễn thông đã nhiệt tình chia sẻ và giúp đỡ tôi hoàn thành học phần này trong chương trình đào tạo thạc sĩ của nhà trường.

Tôi xin gửi lời cảm ơn chân thành và sâu sắc nhất tới người Thầy hướng dẫn của tôi là **TS. Nguyễn Duy Phương** đã luôn giúp đỡ, nhiệt tình hướng dẫn tôi hoàn thành đề án này. Bên cạnh những kiến thức chuyên sâu, uyên bác chia sẻ tới học viên, Thầy còn là niềm động viên, giúp đỡ tôi vượt qua những khó khăn để hoàn đề án.

Mặc dù đã rất cố gắng, nhưng chắc chắn đề án này không tránh khỏi những thiếu sót, rất mong nhận được sự đóng góp của các Thầy Cô để tác giả có thể hoàn thiện nghiên cứu của mình.

Xin chân thành cảm ơn!

Hà Nội, ngày 20 tháng 3 năm 2024

Học viên

Trần Hải Anh

MỤC LỤC

DANH MỤC CÁC CHỮ VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt
CF	Colaborative fittering	Lọc cộng tác
MF	Matrix factorization	Phân tích ma trận
Item - CF	Item – Item Colaborative fittering	Lọc cộng tác sản phẩm
User - CF	User – User Colaborative fittering	Lọc cộng tác người dùng
MFCF	Matrix factorization Colaborative fittering	Lọc cộng tác phân tích ma trận
User - MF	User-based Matrix factorization	Phân tích ma trận dựa trên người dùng
Item - MF	Item-based Matrix factorization	Phân tích ma trận dựa trên sản phẩm
SVD	Singular Value Decomposition	Phân tích giá trị suy biến
RMSE	Root Mean Square Error	Căn bậc hai của sai số bình phương trung bình
PR	PageRank	
DC	Degree Centrality	Hệ số trung tâm trực tiếp
CC	Closeness Centrality	Hệ số trung tâm lân cận
BC	Betweenness Centrality	Hệ số trung tâm trung gian
LC	Load Centrality	
SG	Similarity Graph	Đồ thị tương tự
GHRS	Graph-based Hybrid Recommendation System	Hệ tư vấn kết hợp dựa trên đồ thị

DANH MỤC HÌNH VẼ

DANH MỤC CÁC BẢNG

MỞ ĐẦU

1. Tính cấp thiết của đề tài

Trong cuộc sống ngày nay, chúng ta gặp phải vô vàn tình huống phải đưa ra quyết định. Buổi sáng nên mặc gì cho phù hợp? Lựa chọn thực đơn nào cho gia đình? Nhiệm vụ nào chúng ta nên thực hiện đầu tiên? Nên đăng ký học ở ngôi trường nào? Chúng ta phải trả lời hàng nghìn câu hỏi quan trọng này hàng ngày.

Trên thực tế, trước đây chúng ta thường nhờ các chuyên gia hoặc bạn bè giúp đỡ để đưa ra quyết định, nhưng trong khoảng thời gian vừa qua, với sự gia tăng chóng mặt của các sàn thương mại điện tử, nhu cầu tìm kiếm và mua bán sản phẩm một cách nhanh chóng và phù hợp với sở thích của người tiêu dùng được đặc biệt quan tâm, điều này đã thu hút được sự chú ý và quan tâm từ nhiều nhà nghiên cứu từ khắp nơi trên thế giới với mục tiêu đáp ứng được những đòi hỏi cấp thiết của thị trường. Cùng với đó là sự phát triển không ngừng trong nhiều lĩnh vực đã cung cấp nền tảng vững chắc để triển khai được các phương pháp hiện đại hơn, hoàn thiện hơn. Cũng vì thế mà hàng loạt các hệ tư vấn đã xuất hiện để đáp ứng nhu cầu này. Hệ tư vấn (Recommender System) là một hệ thống lọc thông tin dùng để dự đoán đánh giá và sở thích của một người dùng về các sản phẩm, từ đó hệ thống có thể đưa ra những tư vấn gợi ý sao cho phù hợp.

Những công trình nghiên cứu đã được kiểm chứng như phương pháp Content-Based (đề xuất dựa trên nội dung), Collborative Filtering (lọc cộng tác) đều dựa trên nền tảng học máy truyền thống hay phương pháp được phát triển gần đây như Autoencoder dựa trên kỹ thuật học sâu.

Với lý do trên, học viên đã quyết định lựa chọn đề tài "**Nghiên cứu và ứng dụng kỹ thuật học sâu cho hệ tư vấn**" để thực hiện đề án tốt nghiệp thạc sĩ.

Trong đề án tốt nghiệp lần này, học viên sẽ tìm hiểu khái niệm chung về hệ thống đề xuất, sau đó tập trung vào các thuật toán phổ biến hiện nay, thực hiện viết mã cho tất cả cá phương pháp trên để có được cái nhìn rõ ràng nhất, qua đó nắm bắt được những yếu tố cốt lõi, hiểu được ưu điểm và nhược điểm của các phương pháp

này. Sau cùng là nghiên cứu cài đặt phương pháp áp dụng đồ thị với bộ tự mã hóa rồi thực hiện so sánh hiệu suất với các phương pháp được cài đặt trước đó.

2. Đặt vấn đề

Khi người dùng truy cập vào một nền tảng xem phim nào đó thì vấn đề được đặt ra là: "Làm thế nào để nền tảng đó có thể gợi ý cho người dùng những bộ phim mà họ sẽ yêu thích?". Và câu trả lời chính là cần phải xây dựng được một hệ tư vấn đề xuất các bộ phim hiệu quả cho người dùng.

Thực tế hiện nay nhiều trang web đều đã có hệ thống đề xuất các bộ phim bằng những hiển thị quảng cáo cho người dùng. Để có thể thực hiện tác vụ trên, hệ tư vấn phải sử dụng các thuật toán phân tích đánh giá và đưa dự đoán dựa trên dữ liệu người dùng thu thập được. Nhờ đó hệ thống có thể cá nhân hóa tới người dùng và biết được mỗi người dùng có nhu cầu gì để đưa ra đề xuất thích hợp.

Một hệ tư vấn tốt ảnh hưởng rất lớn đến sự thành bại của các nền tảng và mỗi hệ thống cần tinh chỉnh một hệ tư vấn sao cho phù hợp với dữ liệu mà nền tảng thu thập được. Và trong thực tế, hầu hết các hệ tư vấn đều có thể đạt kết quả rất tốt nếu như sở hữu đủ dữ liệu nhưng sẽ là kém hiệu quả nếu dữ liệu quá ít, điều này khiến những nền tảng vừa và nhỏ sẽ không thể nào tận dụng được những ích lợi mà hệ tư vấn đem lại. Bài toán này là một trong những mục tiêu cần được giải quyết hàng đầu mà nhiều phòng nghiên cứu trên khắp thế giới đang thực hiện.

3. Mục tiêu đề ra

Ngày nay có rất nhiều công trình nghiên cứu về các hệ tư vấn cho người dùng. Nhiều mô hình mới, đa dạng được áp dụng vào thực tế và chất lượng của các mô hình này cũng ngày càng được cải thiện theo thời gian. Tuy nhiên, những phương pháp khác nhau đưa lại những ưu nhược điểm khác nhau. Trong đề án này, học viên sẽ đưa ra hai mục tiêu sau:

1. Nghiên cứu các phương pháp phổ biến đã được xây dựng trước đây và thực hiện cài đặt.

2. Xây dựng mô hình mạng học sâu kết hợp với đồ thị và giải thuật K-means, tiến hành cài đặt và so sánh hiệu suất với các phương pháp phổ biến.

4. Đối tượng và phạm vi nghiên cứu

Trong đề án này, ngoài việc trình bày cơ sở lý thuyết về hệ tư vấn và các phương pháp học máy truyền thống như đề xuất dựa trên nội dung, lọc cộng tác dựa trên người dùng hoặc sản phẩm kèm với đó các kỹ thuật khác như Matrix Factorization, SVD, Autoencoder. Đề án sẽ đi sâu về kỹ thuật đồ thị (Graph-Based) kết hợp với Autoencoder và thuật toán phân cụm K-means để xây dựng mô hình GHRS [21]. Bộ dữ liệu sẽ được sử dụng xuyên suốt đề án này là Movielens-100k.

5. Phương pháp nghiên cứu

Trong quá trình nghiên cứu và thực nghiệm, học viên sẽ kết hợp các công cụ của giải tích, giải thuật phân cụm, lý thuyết đồ thị và kiến trúc mạng cho các phương pháp xây dựng hệ tư vấn kèm với các thư viện của python cho quá trình viết mã.

6. Bố cục của báo cáo

Báo cáo được chia thành ba chương, trong đó:

Chương 1: Tổng quan về hệ tư vấn

Nội dung chính của chương này là trình bày những nghiên cứu cơ bản về hệ tư vấn, các phương pháp tiếp cận phổ biến nhất hiện nay. Trên cơ sở đó trình bày cụ thể một số phương pháp phổ biến hiện nay để có cái nhìn tổng quan khi so sánh với phương pháp được trình bày tại chương 2.

Chương 2: Mô hình dựa trên đồ thị và học sâu

Trình bày cụ thể phương pháp xây dựng mô hình GHRS cũng như cơ sở thực nghiệm sẽ được sử dụng cho việc cài đặt các phương pháp đã trình bày ở cả chương 1 và chương 2.

Chương 3: Kết quả thực nghiệm

Trên cùng một môi trường và tập thử nghiệm, so sánh đầu ra của từng phương pháp kết hợp với kiểm định RMSE và lập bảng so sánh.

Cuối cùng là kết luận và hướng nghiên cứu tiếp theo.

CHƯƠNG I: TỔNG QUAN VỀ HỆ TƯ VẤN

Mục tiêu chính của chương này là trình bày các vấn đề tổng quan của hệ tư vấn, các phương pháp tiếp cận phổ biến trong xây dựng hệ tư vấn và xu hướng mới hiện nay, phân tích rõ những hạn chế của từng phương pháp. Sau đó trình bày chi tiết cơ sở lý thuyết về các phương pháp phổ biến nhất.

1.1. Khái niệm hệ tư vấn

Hệ tư vấn (hệ thống gợi ý hay còn gọi là hệ thống khuyên dùng), tiếng anh là Recommender System hoặc Recommendation System, là một lớp con của hệ thống lọc thông tin, tìm kiếm dự đoán "đánh giá" hoặc "ưa thích" của người dùng với một sản phẩm hoặc đối tượng nào đó. Hệ thống gợi ý chủ yếu dùng trong các ứng dụng thương mại [22]. Dựa theo [2,23], Hệ tư vấn là các công cụ và kỹ thuật phần mềm cung cấp đề xuất các đối tượng có thể hữu ích với người dùng, là công cụ hiệu quả để lọc thông tin trực tuyến được phổ biến rộng rãi do thói quen thay đổi của người dùng máy tính, xu hướng cá nhân hóa và khả năng truy cập Internet mới nổi.

Hệ tư vấn thường hướng tới cá nhân người dùng, dựa trên sở thích của người dùng để đưa ra tư vấn cho một mục mà họ quan tâm. Ví dụ như việc tư vấn một bộ phim cho người dùng. Một trong số những trang web nổi tiếng sử dụng hệ tư vấn để đưa các gợi ý cho người dùng là Netflix.com hình 1.1. Hệ tư vấn sẽ dựa trên lịch sử hoạt động của bạn trên trang web và đưa ra gợi ý mà bạn sẽ quan tâm. Những người dùng có sở thích khác nhau sẽ được đưa ra những tư vấn khác nhau.

Hình 1.1: Giao diện hệ tư vấn của Netflix

(Nguồn: NETFLIX system design)

1.2. Các lĩnh vực ứng dụng của hệ tư vấn

System	Recommended Items
Amazon.com	Books and other products
Netflix	Streaming Videos
GroupLens	News
GoogleNews	News
Youtube	Online videos
TripAdvisor	Travel products
IMDB	Movies

Hình 1.2: Các hệ thống thực tế của một số nền tảng

Hình 1.2 đưa ra một số ứng dụng phổ biến của hệ tư vấn và mục tiêu của chúng. Nhiều mục tiêu trong số này đều thuộc lĩnh vực thương mại điện tử. Tuy nhiên, hệ tư vấn đã phát triển xa hơn chỉ là trong lĩnh vực gợi ý sản phẩm cụ thể. Để thúc đẩy sự phát triển của mạng xã hội, các nền tảng mạng xã hội trực tuyến thường đề xuất các liên kết với khách hàng của họ.

1.3. Phát biểu bài toán cho hệ tư vấn

Trước khi trình bày về các quy trình và hướng tiếp cận, cần làm rõ 2 thuật ngữ sẽ được sử dụng: Người dùng (user) và sản phẩm (item). Thứ nhất, khái niệm người dùng ở đây là người sử dụng hệ thống để thực hiện các thao tác xem, đánh giá, bình luận, … Thứ hai, khái niệm sản phẩm là mặt hàng như các video, bộ phim, bản nhạc, bài báo, … riêng trong đề án này thì item là các bộ phim. Trong hầu hết các hệ tư vấn, dữ liệu được cung cấp dưới dạng đánh giá của người dùng về sản phẩm.

Bảng 1.1: Xếp hạng của người dùng cho bộ phim

	User 1	**User 2**	**User 3**	**...**	**User N**
Item 1	**3**	?	**4**	...	?
Item 2	?	**2**	**1**	...	**2**
Item 3	?	**5**	?	...	**4**
⋮	⋮	⋮	⋮	⋱	⋮
Item M	**5**	?	?	...	**3**

Bài toán được đặt ra như sau: Cho tập hợp hữu hạn gồm N người dùng $\boldsymbol{U} = \{u_1, u_2, \dots, u_n, \dots, u_N\}$, M sản phẩm $\boldsymbol{I} = \{i_1, i_2, \dots, i_m, \dots, i_M\}$ và tập dữ liệu $\boldsymbol{Y} =$

$u, i, r_{u,i}$ trong đó $u \in \boldsymbol{U}$, $i \in \boldsymbol{I}$ và $r_{u,i}$ là đánh giá của người dùng u cho sản phẩm i. Cần dự đoán đánh giá (hay xếp hạng) chưa biết của một người dùng thứ n nào đó u_n cho sản phẩm i_m. (Mọi vectơ đều được biểu diễn dưới dạng cột).

Mỗi người dùng $u_n \in \boldsymbol{U}$ (với $n = 1, 2, \dots, N$) được biểu diễn thông qua tập thông tin cá nhân (biodata) $\boldsymbol{W} = \{w_1, w_2, \dots, w_q, \dots, w_Q\}$. Các $w_q \in \boldsymbol{W}$ là đặc điểm của mỗi người dùng. Ví dụ $\boldsymbol{W}$ có thể bao gồm: nghề nghiệp, giới tính, tuổi, học vấn...

Mỗi sản phẩm $i_m \in \boldsymbol{I}$ (với $m = 1, 2, \dots, M$) được biểu diễn thông qua tập đặc trưng (feature) $\boldsymbol{X} = \{x_1, x_2, \dots, x_g, \dots, x_G\}$. Các $x_g \in \boldsymbol{X}$ là thông tin chi tiết của mỗi sản phẩm. Ví dụ $\boldsymbol{X}$ có thể bao gồm: hãng, thể loại, đạo diễn…

Biểu diễn mối quan hệ giữa người dùng $\boldsymbol{U}$ và sản phẩm $\boldsymbol{I}$ được biểu diễn thông qua ma trận tiện ích (utility matrix) $\boldsymbol{Y} = [r_{n,m}]$ với $n = 1, 2, \dots, N$ và $m = 1, 2, \dots, M$.

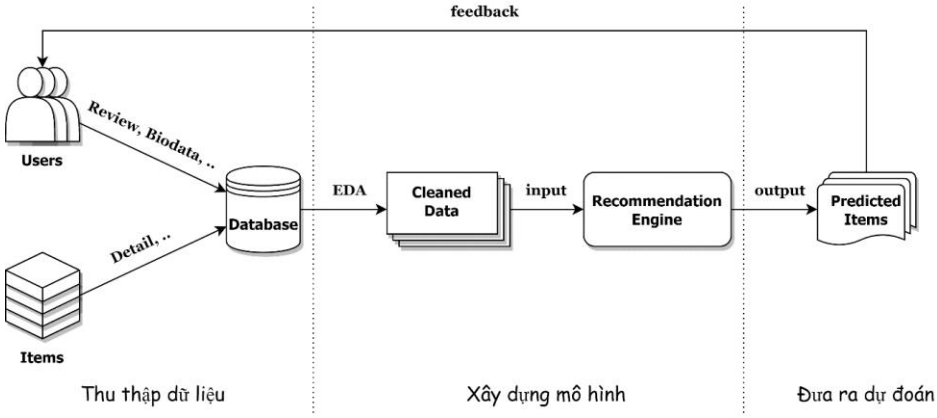
Bảng 1.2: Ví dụ về ma trận tiện ích

Sản phẩm \ Người dùng	**1**	**2**	...	***n***	...	***N***
1	4	?	...	1	...	?
2	1	5	...	3	...	3
⋮	⋮	⋮	...	⋮	...	⋮
m	5	?	...	2	...	1
⋮	⋮	⋮	...	⋮	⋱	⋮
M	?	4	...	?	...	3

Thông thường giá trị của $r_{n,m}$ nằm trong tập số tự nhiên $\{1, 2, 3, \dots\}$ được thu thập bằng cách lấy đánh giá trực tiếp của người dùng hoặc gián tiếp thông qua hệ thống phản hồi đánh giá của khách khàng. Những giá trị $r_{n,m} = ?$ được hiểu là người dùng u_n không có đánh giá nào đối với sản phẩm i_m. Đây cũng là giá trị cần đưa ra dự đoán, sau đó đưa ra danh sách các sản phẩm phù hợp với người dùng đó. Ví dụ với người dùng u_θ, hệ tư vấn sẽ chọn ra $\boldsymbol{z}$ sản phẩm phù hợp với người dùng u_θ nhất để gợi ý. Và để giải quyết bài toán hệ tư vấn này, thông thường sẽ được thực hiện theo quy trình xây dựng ở mục tiếp theo.

1.4. Quy trình xây dựng hệ tư vấn

Quy trình thực hiện thông thường bao gồm 3 bước chính như sau:



Hình 1.3: Quy trình xây dựng hệ tư vấn

- **Bước 1: Thu thập dữ liệu**

Tại giai đoạn đầu tiên, những thông tin mà các hệ thống hay thu thập như:

- Sản phẩm (Item): được mô tả thông qua tập các đặc trưng do NSX cung cấp và nhờ đó các lập trình viên có thể xây dựng dữ liệu thô cho sản phẩm đó.
- Người dùng (User): được mô tả qua thông tin cá nhân mà khách hàng cung cấp và nhờ vậy lập trình viên có thể xây dựng dữ liệu thô cho từng sản phẩm.
- Đánh giá (Rating): được mô tả dưới dạng giá trị mà người dùng xếp loại sản phẩm, sau đó được lưu trong ma trận tiện tích.

- **Bước 2: Xây dựng mô hình**

Bước này có thể thực hiện bằng nhiều hướng khác nhau nhằm đánh giá mối liên hệ giữa các thông tin thu thập được ở Bước 1. Một số hướng tiếp cận được biết đến như: thống kê, học máy, mô hình học sâu, … [3][9]. Mỗi hướng sẽ khai thác dữ liệu đầu vào theo những cách khác nhau, tiếp đó hình thành các phương pháp khác nhau. Nội dung chi tiết sẽ được trình bày cụ thể hơn tại mục 1.5 của đề án.

- **Bước 3: Đưa ra dự đoán**

Kết quả đầu ra của Bước 3 sẽ được dùng để dự đoán các đánh giá xếp loại của người dùng với sản phẩm chưa có đánh giá trước đó và chọn ra $\boldsymbol{z}$ sản phẩm mới phù hợp nhất đối với người dùng hiện thời để đưa ra gợi ý cho họ.

1.5. Các hướng tiếp cận xây dựng hệ tư vấn

Có nhiều cách phân loại các phương pháp xây dựng hệ tư vấn tùy theo quan điểm của mỗi nhà nghiên cứu. Dựa theo bài báo của Cui và cộng sự [4] cùng với nhiều nghiên cứu khác sau này [3], việc phân nhóm được đưa ra có sự chồng chéo lẫn nhau nhưng tổng thể được gom lại thành một số loại được trình bày dưới đây:

1.5.1. Content-based Filtering

Các hệ tư vấn dựa trên nội dung bắt đầu từ việc nghiên cứu truy xuất thông tin và lọc thông tin [5]. Các hệ tư vấn này sẽ tư vấn các mục tương tự như mục mà người dùng đã thích trong quá khứ. Các hệ tư vấn dựa trên nội dung chủ yếu tập trung vào tư vấn các mục có thông tin văn bản như sách, phim và tài liệu. Nội dung trong các hệ thống này được mô tả bằng các sản phẩm và mức độ tin cậy của các sản phẩm đó đối với người dùng thường được đo bằng trọng số TF-IDF. Các phương pháp tiếp cận cho lọc theo nội dung được chia thành hai nhóm chính: Lọc nội dung dựa vào bộ nhớ (Memory-based) và Lọc nội dung dựa vào mô hình (Model-based).

Những vấn đề gặp phải: *Người dùng mới*: Lọc nội dung chỉ hiệu quả khi người dùng đánh giá một lượng sản phẩm đủ lớn. Với người dùng mới, hệ thống không có bất kỳ đánh giá nào nên không thể đưa ra đề xuất thích hợp cho người dùng đó; *Trích chọn đặc trưng*: Phương pháp này chủ yếu dựa vào việc trích chọn đặc trưng trong lĩnh vực truy xuất thông tin. Để có một tập các đặc trưng đầy đủ, nội dung phải được biểu diễn sao cho máy tính có thể tự động phân tích, tính toán các trọng số. Tuy nhiên sẽ khó triển khai nếu dữ liệu phức tạp, tối nghĩa. Ví dụ: dữ liệu hình ảnh, âm thanh.

1.5.2. Collaborative Filtering

Lọc cộng tác (CF) là một kỹ thuật phổ biến nhất để xây dựng hệ tư vấn, khai thác những khía cạnh liên quan đến thói quen sử dụng sản phẩm của cộng đồn người dùng có cùng sở thích trong quá khứ để đưa ra dự đoán các sản phẩm phù hợp nhất. Giả định rằng nếu người dùng đã đồng tình với nhau trong quá khứ thì họ có nhiều khả năng sẽ đồng tình trong tương lai hơn là đồng tình với những người dùng thuộc nhóm khác. Các phương pháp tiếp cận cho CF nói chung cũng chia thành hai nhóm giống như lọc nội dung: CF dựa vào bộ nhớ và CF dựa vào mô hình.

Những vấn đề gặp phải: *Người dùng mới*: Trong trường hợp người dùng mới, họ không có đánh giá cho bất kỳ sản phẩm nào, khi đó CF không thể đưa ra đề xuất chính xác cho những khách hàng này; *Sở thích thay đổi theo thời gian*: Theo tuổi tác tăng trưởng, hoàn cảnh thay đổi theo mùa thì để đưa ra được đề xuất chính xác sẽ gặp khó khăn rất nhiều; *Dữ liệu thưa*: Trên thực tế, lượng sản phẩm lẫn người dùng đều rất lớn nên những đánh giá thu được chỉ là một phần rất nhỏ so với những đánh giá cần dự đoán.

1.5.3. Hybrid Filtering

Lọc kết hợp hay còn lại hệ thống lai là phương pháp kết hợp giữa lọc nội dung và lọc cộng tác nhằm tận dụng những ưu điểm của cả hai phương pháp này. Với lọc nội dung là việc khai thác các khía cạnh liên quan tới đặc điểm trong thông tin đi kèm với từng đối tượng mà không quan tâm tới những người dùng khác. Ngược lại, lọc cộng tác quan tâm đến thói quen người dùng của mỗi khách hàng và độ tương đồng của họ. Mỗi phương pháp đều có những ưu và nhược riêng đã thúc đẩy các nhà nghiên cứu tìm kiếm các phương pháp tận dụng được các ưu điểm đó.

Những vấn đề gặp phải: *Phức tạp trong triển khai*: Hệ tư vấn lai thường khó triển khai thực tế hơn các phương pháp khác do kiến trúc phức tạp của chúng; *Khó trong việc hiểu và giải thích*: Các đặc trưng tiềm ẩn chứa nhiều thứ phức tạp, không thể mô tả theo cách thông thường; Và không phải lúc nào kết hợp đặc tính của CF với lọc nội dung cũng thích hợp, khi bao gồm nhiều đặc trưng thì dữ liệu sẽ chứa nhiều biến dư thừa hơn dẫn đến hiện tượng đa cộng tuyến có thể xảy ra [5].

1.5.4. Other Approaches

Ngoài các phương pháp được đề cập ở trong Phần 1.5.1, 1.5.2 và 1.5.3, còn có một số phương pháp khác được phát triển và đã đạt được nhiều kết quả khả quan như: Phương pháp Knowledge-based sẽ gợi ý sản phẩm dựa trên các suy luận về nhu cầu và sở thích của người dùng. Phương pháp Context-aware, một hệ thống đề xuất dựa trên ngữ cảnh sẽ tích hợp thêm ngoài thông tin người dùng và sản phẩm, hệ thống còn quan tâm tới những yếu tố ngữ cảnh khi người dùng đánh giá một sản phẩm. Thông tin ngữ cảnh bao gồm thời gian, địa điểm hoặc dữ liệu xã hội. Phương pháp

Time-sensitive dựa trên vấn đề thay đổi theo thời gian như đã đề cập trước đó cũng mở đầu đầu cho một phương pháp tiếp cận mới có thể xử lý được bài toán này. Phương pháp Location-based xây dựng hệ thống đề xuất dựa trên vị trí [6]. Hay hệ thống Social-based [7] hoàn toàn dựa trên các khía cạnh của cấu trúc để đề xuất các nút và cạnh liên kết trong mạng xã hội. Mặt khác, hệ thống có thể giới thiệu các sản phẩm khác nhau bằng tín hiệu xã hội (Social Cues) [5]. Demography-based [8], một hệ tư vấn dựa trên nhân khẩu học, thông tin về người dùng được tận dụng để tìm hiểu, phân loại và ánh xạ tới việc đánh giá sản phẩm hoặc xu hướng mua sắm [5]. Trên thực tế, hệ tư vấn dựa trên nhân khẩu học không được phổ biến do các mối lo ngại về bảo mật và quyền riêng tư.

1.6. Phương pháp đánh giá hệ tư vấn

Trong đề án này, để đánh giá độ hiệu quả của hệ tư vấn đưa ra cần dựa trên sai số giữa giá trị dự đoán và giá trị thực tế, cụ thể qua giá trị MSE và RMSE [9].

1.6.1. Mean squared error

Sai số bình phương trung bình (MSE) là một phép toán ước lượng trung bình của bình phương các sai số, tức là sự khác biệt giữa các ước tính và những gì được đánh giá, cụ thể là làm phóng đại các sai số dự báo có giá trị tuyệt đối lớn, do đó chú trọng tới các quan sát đặc biệt (vượt trội) trong mẫu.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1.1}$$

trong đó: n là tổng số mẫu trong tập kiểm tra; y_i là giá trị thực tế tại mẫu i; $\hat{y}_i$ là giá trị dự đoán tại mẫu i.

1.6.2. Root mean squared error

Căn bậc hai của sai số bình phương trung bình (RMSE) hay đơn giản chỉ là MSE lấy căn bậc hai.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{1.2}$$

Giá trị này được sử dụng khi cần chú trọng độ chính xác của các giá trị sẽ được hiển thị thực tế, không bao hàm tính phóng đại sai số như MSE. Kết quả ta mong đợi rằng MSE và RMSE càng nhỏ càng tốt.

1.7. Cơ sở lý thuyết cho các phương pháp phổ biến

1.7.1. Hệ tư vấn sử dụng lọc nội dung

Ý tưởng chính của phương pháp này là gợi ý sản phẩm mới căn cứ theo những sản phẩm mà người dùng đã thích trước đó trong quá khứ. Sự tương đồng giữa sản phẩm được gợi ý và sản phẩm đã được người dùng yêu thích trước đó không nhất thiết phải có mối tương quan trực tiếp mà là dựa trên thuộc tính của các sản phẩm đó. Không giống như các hệ thống CF tận dụng các đánh giá của những người dùng khác, các hệ thống lọc nội dung chủ yếu tập trung vào đánh giá xếp hạng của chính người dùng mục tiêu. Do đó, những người dùng khác có độ quan trọng thấp [6].

- Hệ thống này được xây dựng dựa trên 3 bước [10] chính như sau:

- **Bước 1: Xây dựng thông tin sản phẩm**

Trong các hệ thống dựa trên nội dung, chúng ta cần xây dựng thông tin cho mỗi sản phẩm. Thông tin này được biểu diễn dưới dạng một vectơ đặc trưng. Trong những trường hợp đơn giản, vectơ này được trực tiếp trích xuất từ sản phẩm [11]. Để có thể sử dụng được những đặc trưng này, ta dùng phương pháp ước lượng trọng số của các đặc trưng (TF-IDF) [12]. Phương pháp được triển khai như sau:

Gọi $f_{g,m}$ là tần số (số lần đặc trưng x_g xuất hiện trong sản phẩm i_m). Khi đó tần suất $TF_{g,m}$ của đặc trưng x_g trong sản phẩm i_m được tính theo công thức sau:

$$TF_{g,m} = \frac{f_{g,m}}{\max_k f_{k,m}} \tag{1.3}$$

Nghĩa là tần số của đặc trưng x_d trong sản phẩm i_m chia cho tổng số lần xuất hiện của mọi đặc trưng $\{x_1, x_2, \dots, x_g, \dots, x_G\}$ trong sản phẩm i_m.

$$\max_k f_{k,m} = \sum_{k=1}^{G} f_{k,m} \tag{1.4}$$

Tuy nhiên, những đặc trưng xuất hiện trong nhiều sản phẩm sẽ không được dùng đến do chúng quá phổ biến, việc giữ lại những đặc trưng này khiến việc tính

toán trở nên khó khan hơn mà không đem lại được ý nghĩa về mặt phân loại các sản phẩm. Vì vậy, IDF được sử dụng để giải quyết được vấn đề này.

$$IDF_g = \log\left(\frac{M}{\mathrm{df}_g}\right) \quad \Big| \quad IDF_g = \log\left(\frac{M+1}{\mathrm{df}_g+1}\right)+1 \tag{1.5}$$

Trong đó: df_g là đặc trưng x_g xuất hiện trong bao nhiêu sản phẩm; M là tổng số sản phẩm. Trong đề án này, công thức bên phải sẽ được áp dụng thay thế cho công thức thông thường vì code sử dụng thư viện sklearn với $smooth\_idf = True$.

Kết hợp công thức (1.3) và (1.5) ta có phương trình tổng quát sau:

$$tf.idf_{g,m} = TF_{g,m} \times IDF_g \tag{1.6}$$

- **Bước 2: Xây dựng hồ sơ người dùng**

Đặt số lượng người dùng là N, số lượng sản phẩm là M. Đối với mỗi người dùng $u_n \in \boldsymbol{U}$, sẽ có ma trận $\ddot{\boldsymbol{X}}_n$ là ma trận con của ma trận thông tin - sản phẩm $\boldsymbol{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m, \dots, \mathbf{x}_M\} \in \mathbb{R}^{G\times M}$ ($\mathbf{x}_m$ là vectơ cột). $\{x_1, x_2, \dots, x_g, \dots, x_G\}$ là vectơ trọng số các đặc trưng sản phẩm cho người dùng u_n được tính toán từ $\ddot{\boldsymbol{X}}_n$. Vectơ của sản phẩm u_n có thể tính bằng nhiều kỹ thuật khác nhau nhưng trong phần này sẽ sử dụng mô hình hồi quy Ridge [6][11] để tính vectơ trọng số.

Để thuận tiện, ta gọi ma trận người dùng - sản phẩm là ma trận tiện ích (utility matrix) $\boldsymbol{Y} \in \mathbb{R}^{M\times N}$ chứa tất cả đánh giá dạng số, bao gồm cả những giá trị cần dự đoán và dữ liệu đánh giá được cung cấp trước đó.

Giả sử rằng tìm được mô hình cho mỗi người dùng, được minh họa bằng một vectơ cột hệ số $\mathbf{w}_n \in \mathbb{R}^G$ và hệ số bias b_n sao cho mức độ quan tâm của một người dùng tới một sản phẩm được tính bằng một hàm tuyến tính:

$$\hat{\mathrm{y}}_{m,n} = \mathbf{w}_n^T \mathbf{x}_m + b_n \tag{1.7}$$

Xét người dùng thứ n, nếu coi tập huấn luyện là tập hợp các thành phần đã biết của $\mathbf{y}_n$ (cột thứ n của ma trận $\boldsymbol{Y}$), ta rút gọn nó bằng đặt $\ddot{\boldsymbol{y}}_n = \{y_{1,n}, y_{2,n}, \dots, y_{s_n,n}\} \in \mathbb{R}^{s_n}$ là vectơ con của $\mathbf{y}_n$ được xây dựng bằng cách trích các thành phần đã biết có tổng số là s_n tại cột thứ n của ma trận $\boldsymbol{Y}$. Đồng thời có $\ddot{\boldsymbol{X}}_n \in \mathbb{R}^{G\times s_n}$ là ma trận con của $\boldsymbol{X}$, thu được bằng cách trích các cột tương ứng với sản phẩm

đã được đánh giá bởi người dùng thứ n. Tạo thêm vectơ cột $\mathbf{e}_n$ với tất cả thành phần bằng 1, ta có thể xây dựng hàm mất mát cho người dùng thứ n như sau:

$$\mathcal{L}_n(\mathbf{w}_n, b_n) = \frac{1}{2s_n}\sum_{m=1}^{s_n}\left(\hat{y}_{m,n} - y_{m,n}\right)^2 + \frac{\lambda}{2s_n}\|\mathbf{w}_n\|_2^2$$
$$= \frac{1}{2s_n}\left\|\ddot{X}_n^T\mathbf{w}_n + b_n\mathbf{e}_n - \ddot{\mathbf{y}}_n\right\|_2^2 + \frac{\lambda}{2s_n}\|\mathbf{w}_n\|_2^2 \quad (1.8)$$

Cặp nghiệm $\mathbf{w}_n$ và b_n sẽ được tính bằng gradient descent hoặc có thể giải nghiệm trực tiếp vì regularization đã đảm bảo ma trận X^TX là khả nghịch với M gần giống ma trận đơn vị $\boldsymbol{I}$ với duy nhất một hàng (cột) đầu tiên là 0 thông qua công thức sau:

$$\theta = (X^TX + \lambda[M])^{-1}X^T\boldsymbol{y} \quad (1.9)$$

- **Bước 3: Tính giá trị chưa biết cho ma trận tiện ích**

Tùy thuộc vào mỗi phương pháp mà hệ thống sẽ tính toán khác nhau. Có thể sử dụng độ tương tự cosine giữa vectơ trọng số $\mathbf{w}_n$ của người dùng với từng trọng số $\mathbf{x}_?$ của mỗi sản phẩm mà người dùng chưa đánh giá để dự đoán mức độ phù hợp của người dùng n với những sản phẩm chưa đánh giá là bao nhiêu [12].

$$\text{cosine}(\mathbf{w}_n, \mathbf{x}_?) = \frac{\mathbf{w}_n^T \cdot \mathbf{x}_?}{\|\mathbf{w}_n\|_2\|\mathbf{x}_?\|_2} = \frac{\sum_{i=0}^{G-1} w_{i,n}x_{i,?}}{\sqrt{\sum_{i=0}^{G-1} w_{i,n}^2}\sqrt{\sum_{i=0}^{G-1} x_{i,?}^2}} \quad (1.10)$$

Trong phần này, do sử dụng mô hình hồi quy nên chỉ cần nhân tích vô hướng giữa $\mathbf{w}_n$ với $\mathbf{x}_?$ và cộng với hệ số b_n sẽ cho ra giá trị dự đoán cần tìm như công thức (1.7). Cuối cùng ta có được ma trận tiện ích dự đoán $\widehat{\mathbf{Y}}$ bằng công thức tổng quát sau:

$$\widehat{\mathbf{Y}} = \boldsymbol{X}^T \cdot \boldsymbol{W} + \mathbf{b} \quad (1.11)$$

Sau khi hoàn thiện ma trận tiện ích, ta chỉ cần chọn z sản phẩm có giá trị cao nhất trong cột $\widehat{\mathbf{Y}}_n$ để đề xuất cho người dùng thứ n.

1.7.2. Hệ tư vấn sử dụng lọc cộng tác

Ý tưởng của lọc cộng tác là xác định mức độ quan tâm của một người dùng tới một sản phẩm dựa trên những người dùng có hành vi tương tự. Việc xác định sự tương tự giữa người dùng có thể được xác định thông qua mức độ quan tâm của họ tới sản phẩm khác mà hệ thống đã biết [11].

1.7.2.1. Lọc cộng tác theo người dùng

Thuật toán cốt lõi của User - CF là tìm những người dùng có hành vi đánh giá trong quá khứ tương tự với người dùng cần dự đoán và sử dụng đánh giá của những người dùng tương tự đó để dự đoán cái mà người dùng cần dự đoán sẽ thích. Việc cần làm là xác định độ tương tự (similarity) giữa hai người dùng. Giả sử thông tin duy nhất ta có là ma trận tiện ích $\boldsymbol{Y}$ mà không dùng dữ liệu bên ngoài. Độ tương tự sẽ được xác định dựa trên các cột tương ứng của họ trong ma trận.

Trải qua nhiều thập niên nghiên cứu và phát triển, đã có rất nhiều công thức tính độ tương tự được đề xuất và một vài trong số đó đã được thử nghiệm thực tế và tổng hợp bởi Fethi Fkih [13]. Tuy nhiên, do giới hạn của đề án nên chỉ sử dụng công thức tính độ tương tự thông dụng nhất là cosine với $u_{\alpha,\beta}$ là các vectơ người dùng (vectơ cột) tương ứng trong ma trận tiện ích chuẩn hóa $\overline{\mathrm{Y}}$.

$$\text{cosine}(u_\alpha, u_\beta) = \frac{\mathbf{u}_\alpha^T \cdot \mathbf{u}_\beta}{\|\mathbf{u}_\alpha\|_2 \|\mathbf{u}_\beta\|_2} = \frac{\sum_{i=0}^{M-1} \bar{y}_{i,\alpha} \bar{y}_{i,\beta}}{\sqrt{\sum_{i=0}^{M-1} \bar{y}_{i,\alpha}^2} \sqrt{\sum_{i=0}^{M-1} \bar{y}_{i,\beta}^2}} \tag{1.12}$$

Để có thể đo được độ tương tự giữa hai người dùng, cách thường làm là xây dựng vectơ đặc trưng cho mỗi người dùng rồi áp dụng độ tương tự cosine giữa hai vectơ. Các vectơ đặc trưng được xây dựng dựa trên ma trận tiện ích $\boldsymbol{Y}$, tuy nhiên khó khăn đặt ra vì ma trận này thường là một ma trận thưa (sparse matrix) bao gồm nhiều giá trị bị khuyết vì người dùng thường chỉ đánh giá một lượng rất nhỏ các sản phẩm. Giải pháp đơn giản nhất là điền vào những phần trống này một giá trị ước lượng. Những giá trị này chỉ phục vụ cho phần tính độ tương tự chứ không phải kết quả cuối cùng mà hệ thống cần dự đoán. Giả sử ta có ma trận tiện ích $\boldsymbol{Y}$ sau:

Bảng 1.3: Ma trận tiện ích ban đầu $\boldsymbol{Y}$ [11]

		Người dùng						
		u_0	u_1	u_2	u_3	u_4	u_5	u_6
Sản phẩm	i_0	5	5	2	0	1	?	?
	i_1	4	?	?	0	?	2	?
	i_2	?	4	1	?	?	1	1
	i_3	2	2	3	4	4	?	4
	i_4	2	0	4	?	?	?	5

	↓	↓	↓	↓	↓	↓	↓
$\tilde{u}_j$	3.25	2.75	2.5	1.33	2.5	1.5	3.33

Việc tính giá trị ước lượng cho các ô trống (?) cần một phương pháp để tránh những trường hợp người dùng là khó tính có thói quen đánh giá thấp sản phẩm và người dùng dễ tính thì dù sản phẩm không tốt vẫn cho mức đánh giá cao. Nếu ta chỉ điền bằng những giá trị 0 thì có khả năng bị nhầm với đánh giá thấp. Một phương pháp khá ổn để xử lý trường hợp này là điền các giá trị khuyết bằng 0, còn những giá trị được biết đến sẽ trừ đi giá trị trung bình mà người dùng đã đánh giá.

$$\tilde{u}_j = \frac{\sum_{l:y_{?,j}\neq ?} y_{l,j}}{\text{count}\left(y_{?,j} \neq ?\right)} \tag{1.13}$$

Bảng 1.4: Ma trận tiện ích chuẩn hóa $\overline{\mathbf{Y}}$ [11]

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	1.75	2.25	−0.5	−1.33	−1.5	**0**	**0**
i_1	0.75	**0**	**0**	−1.33	**0**	0.5	**0**
i_2	**0**	1.25	−1.5	**0**	**0**	−0.5	−2.33
i_3	−1.25	−0.75	0.5	2.67	1.5	**0**	0.67
i_4	−1.25	−2.75	1.5	**0**	**0**	**0**	1.67

Sau khi dữ liệu đã được chuẩn hóa, hàm tương tự cosine (1.12) được sử dụng để tính ma trận tương tự người dùng **S** với số hàng bằng số người dùng N.

Bảng 1.5: Ma trận tương tự người dùng S [11]

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
u_0	1	0.83	−0.58	−0.79	−0.82	0.2	−0.38
u_1	0.83	1	−0.87	−0.4	−0.55	−0.23	−0.71
u_2	−0.58	−0.87	1	0.27	0.32	0.47	0.96
u_3	−0.79	−0.4	0.27	1	0.87	−0.29	0.18
u_4	−0.82	−0.55	0.32	0.87	1	0	0.16
u_5	0.2	−0.23	0.47	−0.29	0	1	0.56
u_6	−0.38	−0.71	0.96	0.18	0.16	0.56	1

Tương tự với KNN, thuật toán User - CF cũng sử dụng k_users lân cận để dự đoán. Sử dụng ma trận tương tự người dùng **S** với mỗi người dùng u_n cần dự đoán sản phẩm chưa được đánh giá i_m, ta thu được k người dùng có độ tương tự gần nhất

với u_n: $\aleph(u_n, i_m) = \{u_\kappa | \kappa \in k\_\text{users}\}$. Sau đó để dự đoán độ quan tâm của người dùng bằng cách kết hợp với ma trận tiện ích chuẩn hóa $\overline{\mathbf{Y}}$ qua công thức sau.

$$\hat{y}_{u_n,i_m} = \frac{\sum_{u_j \in \aleph(u_n,i_m)} [\bar{y}_{i_m,j} \cdot \text{cosine}(u_n, u_j)]}{\sum_{u_j \in \aleph(u_n,i_m)} |\text{cosine}(u_n, u_j)|} \quad (1.14)$$

Bảng 1.6: Ma trận tiện ích chuẩn hóa sau hoàn thiện [11]

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	1.75	2.25	−0.5	−1.33	−1.5	**0.18**	**−0.63**
i_1	0.75	**0.48**	**−0.17**	−1.33	**−1.33**	0.5	**0.05**
i_2	**0.91**	1.25	−1.5	**−1.84**	**−1.78**	−0.5	−2.33
i_3	−1.25	−0.75	0.5	2.67	1.5	**0.59**	0.67
i_4	−1.25	−2.75	1.5	**1.57**	**1.56**	**1.59**	1.67

Thực hiện tính giá trị cần dự đoán cho toàn bộ ma trận tiện ích chuẩn hóa $\overline{\text{Y}}$, ta cần đưa giá trị về thang đánh giá cũ để có thể thực hiện đề xuất và tính giá trị RMSE bằng cách cộng lại với giá trị $\tilde{u}_j$ tương ứng.

Bảng 1.7: Ma trận tiện ích sau hoàn thiện $\widehat{\mathbf{Y}}$ [11]

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	5	5	2	0	1	**1.68**	**2.7**
i_1	4	**3.23**	**2.33**	0	**1.67**	2	**3.38**
i_2	**4.15**	4	1	**−0.5**	**0.71**	1	1
i_3	2	2	3	4	4	**2.1**	4
i_4	2	0	4	**2.9**	**4.06**	**3.1**	5

Sau khi thu được ma trận $\widehat{\mathbf{Y}}$, ta chỉ cần chọn z sản phẩm có giá trị cao nhất trong cột $\widehat{\mathbf{Y}}_n$ để đề xuất cho người dùng thứ n.

1.7.2.2. Lọc cộng tác theo sản phẩm

User – CF đạt được nhiều thành công trong quá khứ nhưng cũng gặp phải một số hạn chế khi được sử dụng rộng rãi như: *Sự thưa thớt*: Thực tế ngay cả với người dùng tích cực nhất cũng chỉ có thể mua được số sản phẩm chiếm tỷ lệ rất thấp trong tổng số sản phẩm. Do đó, hệ tư vấn User – CF có thể không đưa ra bất kỳ gợi ý nào; *Khả năng mở rộng*: Các thuật toán CF luôn yêu cầu tính toán tăng dần theo lượng người dùng và sản phẩm. Với lượng người dùng quá lớn sẽ là gánh nặng về mặt tính toán, hơn nữa ma trận tiện ích $\boldsymbol{Y}$ thường rất thưa vì người dùng không có thói quen

đánh giá nhiều sản phẩm nên khi một đánh giá bị thay đổi cũng kéo theo ma trận tương tự người dùng **S** cần thực hiện lại; *Yêu cầu khả năng lưu trữ*: Khi lượng người dùng lớn hơn số lượng sản phẩm (thực tế điều này luôn xảy ra, đặc biệt là các sàn thương mại điện tử với cơ sở người dùng lớn), mỗi chiều của ma trận tương tự bằng với số lượng người dùng M nên việc lưu trữ ma trận tương tự là không khả thi.

Một cách tiếp cận khác là lọc cộng tác sản phẩm (Item - CF), được đề xuất bởi Sarwar cùng cộng sự [14] và được Amazon sử dụng cho hệ tư vấn của họ [15]. Cách thức tính toán thay vì tìm sự tương tự giữa các người dùng, ta có thể tìm sự tương tự giữa các sản phẩm. Từ đó nếu một người dùng thích một sản phẩm thì hệ thống nên gợi ý các sản phẩm tương tự với sản phẩm đó. Và nếu lượng sản phẩm nhỏ hơn số lượng người dùng, mô hình này sẽ có những ưu điểm như tính toán ít hơn do ma trận tiện ích có số hàng ít hơn số cột nên ảnh hưởng bởi đánh giá của một người dùng sẽ ít ảnh hưởng đến giá trị trung bình của tổng các đánh giá của mọi người dùng tới sản phẩm đó. Như vậy ma trận tương tự sản phẩm **S** sẽ không cần cập nhật quá thường xuyên. Thêm nữa là ma trận tương tự sản phẩm **S** có kích thước nhỏ hơn với số hàng bằng số sản phẩm M nên giúp lưu trữ và tính toán ở những bước sau hiệu quả hơn.

Bảng 1.8: Ma trận tiện ích ban đầu Y [11]

Người dùng

Sản phẩm	u_0	u_1	u_2	u_3	u_4	u_5	u_6		$\tilde{\imath}_j$
i_0	5	5	2	0	1	?	?	→	2.6
i_1	4	?	?	0	?	2	?	→	2
i_2	?	4	1	?	?	1	1	→	1.75
i_3	2	2	3	4	4	?	4	→	3.17
i_4	2	0	4	?	?	?	5	→	2.75

Bảng 1.9: Ma trận tiện ích chuẩn hóa $\overline{Y}$ [11]

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	2.4	2.4	−0.6	−2.6	−1.6	**0**	**0**
i_1	2	**0**	**0**	−2	**0**	0	**0**
i_2	**0**	2.25	−0.75	**0**	**0**	−0.75	−0.75
i_3	−1.17	−1.17	−0.17	0.83	0.83	**0**	0.83
i_4	−0.75	−2.75	1.25	**0**	**0**	**0**	2.25

Độ tương tự sản phẩm được tính giống với công thức (1.12):

$$\text{cosine}(i_\alpha, i_\beta) = \frac{\mathbf{i}_\alpha^T \cdot \mathbf{i}_\beta}{\|\mathbf{i}_\alpha\|_2 \|\mathbf{i}_\beta\|_2} = \frac{\sum_{u=0}^{N-1} \bar{y}_{\alpha,u} \bar{y}_{\beta,u}}{\sqrt{\sum_{u=0}^{N-1} \bar{y}_{\alpha,u}^2} \sqrt{\sum_{u=0}^{N-1} \bar{y}_{\beta,u}^2}} \quad (1.15)$$

Bảng 1.10: Ma trận tương tự sản phẩm S [11]

	i_0	i_1	i_2	i_3	i_4
i_0	1	0.77	0.49	−0.89	−0.52
i_1	0.77	1	0	−0.64	−0.14
i_2	0.49	0	1	−0.55	−0.88
i_3	−0.89	−0.64	−0.55	1	0.68
i_4	−0.52	−0.14	−0.88	0.68	1

Từ ma trận **S** này, ta nhận thấy có hai nhóm sản phẩm khác nhau nhờ các giá trị không âm, nhóm 1 $\{i_0, i_1, i_2\}$ và nhóm 2 $\{i_3, i_4\}$. Điều này giúp dự đoán chính xác hơn vì khi này những sản phẩm trong một nhóm rất tương đồng về tính chất bị ẩn nào đó, nếu ta đề xuất một sản phẩm tương tự thì người dùng sẽ có tỷ lệ hài lòng cao hơn.

Bảng 1.11: Ma trận tiện ích chuẩn hóa sau hoàn thiện [11]

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	2.4	2.4	−0.6	−2.6	−1.6	**−0.29**	**−1.52**
i_1	2	**2.4**	**−0.6**	−2	**−1.25**	0	**−2.25**
i_2	**2.4**	2.25	−0.75	**−2.6**	**−1.2**	−0.75	−0.75
i_3	−1.17	−1.17	−0.17	0.83	0.83	**0.344**	0.83
i_4	−0.75	−2.75	1.25	**1.03**	**1.16**	**0.65**	2.25

Thực hiện tính giá trị cần dự đoán cho toàn bộ ma trận tiện ích chuẩn hóa $\overline{\mathbf{Y}}$, ta cần đưa giá trị về thang đánh giá cũ để có thể thực hiện đề xuất và tính giá trị RMSE bằng cách cộng lại với giá trị $\tilde{\imath}_j$ tương ứng.

Bảng 1.12: Ma trận tiện ích sau hoàn thiện $\widehat{\mathbf{Y}}$

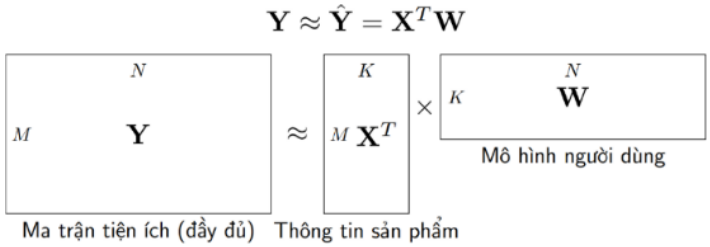
	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	5	5	2	0	1	**2.31**	**1.08**
i_1	4	**4.4**	**1.4**	0	**0.75**	2	**−0.25**
i_2	**4.15**	4	1	**−0.85**	**0.55**	1	1
i_3	2	2	3	4	4	**3.51**	4
i_4	2	0	4	**3.78**	**3.91**	**3.4**	5

Sau khi thu được ma trận $\widehat{\mathbf{Y}}$, ta chỉ cần chọn z sản phẩm có giá trị cao nhất trong cột $\widehat{\mathbf{Y}}_n$ để đề xuất cho người dùng thứ n.

1.7.2.3. Lọc cộng tác phân tích ma trận

Khác với hai phương pháp trên còn được biết đền là lọc cộng tác lân cận, một phương pháp được gọi là matrix factorization (MF) cho CF được biết đến lần đầu tiên do Simon Funk đăng trên blog năm 2006 [16] kể về lý do đằng sau việc anh ấy và đồng nghiệp giành được vị trí thứ 3 trong giải thưởng Netflix. Thay vì áp dụng mô hình Singular Value Decomposition (SVD), giải pháp của Simon Funk là phân tích ma trận tiện ích thành tích hai ma trận có số chiều thấp hơn, ma trận thứ nhất có hàng cho mỗi người dùng và ma trận thứ hai có cột tương ứng với mỗi sản phẩm. Hàng và cột này được liên kết với nhau được gọi là latent feature.

Với phương pháp trên kết hợp với nghiên cứu của Koren và cộng sự [17]. Nhiệm vụ hàng đầu là cần cố gắng tính xấp xỉ ma trận tiện ích $\boldsymbol{Y} \in \mathbb{R}^{M \times N}$ bằng tích hai ma trận: ma trận thông tin sản phẩm $\mathbf{X} \in \mathbb{R}^{K \times M}$ và ma trận mô hình người dùng $\mathbf{W} \in \mathbb{R}^{K \times N}$. Giá trị K ở đây chính là tính chất tiềm ẩn và thường nhỏ hơn so với M và N, khi đó cả hai ma trận $\mathbf{X}$ và $\mathbf{W}$ đều có hạng (rank) không vượt quá K.



Hình 1.4: Phân tích ma trận [11]

Những ưu điểm của phương pháp phân tích ma trận: Trong thực tế, số lượng người dùng N và số lượng sản phẩm M là vô cùng lớn, ví dụ như tập dữ liệu do Netflix cung cấp trong các cuộc thi của mình. Thay vì phải tính ma trận tương tự người dùng hay sản phẩm luôn yêu cầu về bộ nhớ rất lớn thì việc huấn luyện để tối ưu một trong hai ma trận $\mathbf{X}$ hoặc $\mathbf{W}$ và cố định ma trận còn lại có vẻ phức tạp hơn nhưng khi thực hiện tính giá trị dự đoán đơn giản hơn rất nhiều vì chỉ cần tính tích vô hướng hai ma trận để tìm $\hat{\mathbf{Y}}$. Hơn nữa nếu đặt K càng nhỏ thì việc tính toán sẽ càng nhanh; Một vấn đề khác của CF là ma trận $\hat{\mathbf{Y}}$ yêu cầu bộ nhớ rất lớn, vấn đề có thể giải quyết bằng việc chỉ cần lưu hai ma trận $\mathbf{X}$ và $\mathbf{W}$ có tổng kích thước nhỏ hơn rất nhiều. Cụ thể với

$(K \ll M, N)$ bộ nhớ yêu cầu cho phương pháp MF: $K(M+N)$ phần tử để lưu hai ma trận $\mathbf{X}$ và $\mathbf{W}$; phương pháp thông thường: $M \times N$ phần tử cho $\widehat{\mathbf{Y}}$, M^2 hoặc N^2 phần tử cho ma trận tương tự $\mathbf{S}$.

- **Bước 1: Xây dựng hàm mất mát**

Để dự đoán được xếp hạng của người dùng thứ n cho sản phẩm m có thể được tính bằng công thức $\hat{y}_{m,n} = \mathbf{x}_m^T \mathbf{w}_n$ tương tự với công thức (1.7). Để tăng độ chính xác ta cho thêm hệ số bias vào công thức này và thực hiện tối ưu nó.

$$y_{m,n} = \hat{y}_{m,n} = \mathbf{x}_m^T \mathbf{w}_n + b_m + d_n \tag{1.16}$$

Trong đó b_m và d_n lần lượt là các hệ số điều chỉnh tương ứng với sản phẩm m và người dùng n. Vectơ $\mathbf{b} = [b_1, b_2, \dots, b_M]^T$ là vectơ bias cho các sản phẩm, vectơ $\mathbf{d} = [d_1, d_2, \dots, d_N]^T$ là vectơ bias cho các người dùng. Kết hợp cả hai hệ số bias để xây dựng hàm mất mát (loss function) cho MFCF [17]:

$$\mathcal{L}(\mathbf{X}, \mathbf{W}, \mathbf{b}, \mathbf{d}) = \sum_{n=1}^{N} \left[\frac{1}{2s_n} \sum_{m=1}^{s_n} (\mathbf{x}_m^T \mathbf{w}_n + b_m + d_n - y_{m,n})^2 \right] + \frac{\lambda}{2} (\|\mathbf{X}\|_F^2 + \|\mathbf{W}\|_F^2) \tag{1.17}$$

Tương tự như mô hình hồi quy cho lọc nội dung (1.8) kết hợp với Frobenius Norm:

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{K} |x_{i,j}|^2} \quad \Big| \quad \|\mathbf{W}\|_F = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{K} |w_{i,j}|^2} \tag{1.18}$$

Tuy nhiên việc tối ưu đồng thời cả $\mathbf{X}, \mathbf{W}, \mathbf{b}, \mathbf{d}$ là rất phức tạp. Chính vì vậy ta tối ưu theo từng cặp $(\mathbf{X}, \mathbf{b})$, $(\mathbf{W}, \mathbf{d})$ trong khi cố định cặp còn lại. Thực hiện tính toán lặp đi lặp lại cho đến khi hàm loss có giá trị tối ưu.

- **Bước 2: Tối ưu hàm mất mát**

- Khi cố định cặp $(\mathbf{X}, \mathbf{b})$, tối ưu cặp $(\mathbf{W}, \mathbf{d})$:

$$\mathcal{L}(\mathbf{W}) = \sum_{n=1}^{N} \left[\frac{1}{2s_n} \sum_{m=1}^{s_n} (\mathbf{x}_m^T \mathbf{w}_n + b_m + d_n - y_{m,n})^2 \right] + \frac{\lambda}{2} \|\mathbf{W}\|_F^2 \tag{1.19}$$

Bài toán tối ưu cặp $(\mathbf{W}, \mathbf{d})$ tách thành N bài toán nhỏ:

$$\begin{aligned} \mathcal{L}_1(\mathbf{w}_n, d_n) &= \frac{1}{2s_n} \sum_{m=1}^{s_n} (\mathbf{x}_m^T \mathbf{w}_n + b_m + d_n - y_{m,n})^2 + \frac{\lambda}{2} \|\mathbf{w}_n\|_2^2 \\ &= \frac{1}{2s_n} \left\| \widehat{\mathbf{X}}_n^T \mathbf{w}_n + \hat{\mathbf{b}}_n + d_n \mathbf{e}_n - \hat{\mathbf{y}}_n \right\|_2^2 + \frac{\lambda}{2} \|\mathbf{w}_n\|_2^2 \end{aligned} \tag{1.20}$$

Trong đó $\widehat{\mathbf{X}}_n \in \mathbb{R}^{K \times s_n}$ là ma trận con được tạo bởi các cột của $\mathbf{X}$ tương ứng với các sản phẩm đã được người dùng thứ n (s_n tổng các sản phẩm đã được người dùng thứ n đánh giá). Tạo thêm vectơ $\mathbf{e}_n$ với tất cả thành phần bằng 1 với kích thước phù hợp. $\hat{\mathbf{b}}_n$ là vector bias tương ứng và $\hat{\mathbf{y}}_n$ là các đánh giá tương ứng.

Tính gradient:

$$\nabla_{\mathbf{w}_n} \mathcal{L}_1 = \frac{1}{s_n} \widehat{\mathbf{X}}_n (\widehat{\mathbf{X}}_n^T \mathbf{w}_n + \hat{\mathbf{b}}_n + d_n \mathbf{e}_n - \hat{\mathbf{y}}_n) + \lambda \mathbf{w}_n \tag{1.21}$$

$$\nabla_{d_n} \mathcal{L}_1 = \frac{1}{s_n} \mathbf{e}_n^T (\widehat{\mathbf{X}}_n^T \mathbf{w}_n + \hat{\mathbf{b}}_n + d_n \mathbf{e}_n - \hat{\mathbf{y}}_n) \tag{1.22}$$

Cập nhật trọng số $\mathbf{w}_n$ và bias d_n:

$$\mathbf{w}_n \leftarrow \mathbf{w}_n - \mu \left[\frac{1}{s_n} \widehat{\mathbf{X}}_n (\widehat{\mathbf{X}}_n^T \mathbf{w}_n + \hat{\mathbf{b}}_n + d_n \mathbf{e}_n - \hat{\mathbf{y}}_n) + \lambda \mathbf{w}_n \right] \tag{1.23}$$

$$d_n \leftarrow d_n - \mu \left[\frac{1}{s_n} \mathbf{e}_n^T (\widehat{\mathbf{X}}_n^T \mathbf{w}_n + \hat{\mathbf{b}}_n + d_n \mathbf{e}_n - \hat{\mathbf{y}}_n) \right] \tag{1.24}$$

- Khi cố định cặp $(\mathbf{W}, \mathbf{d})$, tối ưu cặp $(\mathbf{X}, \mathbf{b})$:

$$\mathcal{L}(\mathbf{X}) = \sum_{n=1}^{N} \left[\frac{1}{2s_m} \sum_{n=1}^{s_m} (\mathbf{w}_n^T \mathbf{x}_m + d_n + b_m - y_{m,n})^2 \right] + \frac{\lambda}{2} \|\mathbf{X}\|_F^2 \tag{1.25}$$

Bài toán tối ưu cặp $(\mathbf{X}, \mathbf{b})$ tách thành M bài toán nhỏ:

$$\begin{aligned} \mathcal{L}_2(\mathbf{x}_m, b_m) &= \frac{1}{2s_m} \sum_{n=1}^{s_m} (\mathbf{w}_n^T \mathbf{x}_m + d_n + b_m - \hat{\mathbf{y}}_m)^2 + \frac{\lambda}{2} \|\mathbf{x}_m\|_2^2 \\ &= \frac{1}{2s_m} \left\| \widehat{\mathbf{W}}_m^T \mathbf{x}_m + \hat{\mathbf{d}}_m + b_m \mathbf{e}_m - \hat{\mathbf{y}}_m \right\|_2^2 + \frac{\lambda}{2} \|\mathbf{x}_m\|_2^2 \end{aligned} \tag{1.26}$$

Trong đó $\widehat{\mathbf{W}}_m \in \mathbb{R}^{K \times s_m}$ là ma trận con được tạo bởi các cột của $\mathbf{W}$ tương ứng với các người dùng đã đánh giá sản phẩm thứ m (s_m tổng các người dùng đã đánh giá sản phẩm thứ m). Tạo thêm vectơ $\mathbf{e}_m$ với tất cả thành phần bằng 1 với kích thước phù hợp. $\hat{\mathbf{d}}_m$ là vector bias tương ứng và $\hat{\mathbf{y}}_m$ là các đánh giá tương ứng.

Tính gradient:

$$\nabla_{\mathbf{x}_m} \mathcal{L}_2 = \frac{1}{s_m} \widehat{\mathbf{W}}_m (\widehat{\mathbf{W}}_m^T \mathbf{x}_m + \hat{\mathbf{d}}_m + b_m \mathbf{e}_m - \hat{\mathbf{y}}_m) + \lambda \mathbf{x}_m \tag{1.27}$$

$$\nabla_{b_m} \mathcal{L}_2 = \frac{1}{s_m} \mathbf{e}_m^T (\widehat{\mathbf{W}}_m^T \mathbf{x}_m + \hat{\mathbf{d}}_m + b_m \mathbf{e}_m - \hat{\mathbf{y}}_m) \tag{1.28}$$

Cập nhật trọng số $\mathbf{w}_n$ và bias d_n:

$$\mathbf{x}_m \leftarrow \mathbf{x}_m - \mu \left[\frac{1}{s_m} \widehat{\mathbf{W}}_m (\widehat{\mathbf{W}}_m^T \mathbf{x}_m + \hat{\mathbf{d}}_m + b_m \mathbf{e}_m - \hat{\mathbf{y}}_m) + \lambda \mathbf{x}_m \right] \tag{1.29}$$

$$b_m \leftarrow b_m - \mu\left[\frac{1}{s_m}\mathbf{e}_m^T(\widehat{\mathbf{W}}_m^T\mathbf{x}_m + \hat{\mathbf{d}}_m + b_m\mathbf{e}_m - \hat{\mathbf{y}}_m)\right] \quad (1.30)$$

1.7.2.4. Lọc cộng tác dựa trên bộ tự mã hóa

Trong những năm gần đây, mạng nơ-ron học sâu hay tên gọi tiếng anh là Deep neural networks đã được sử dụng phổ biến trong các hệ thống đề xuất. Một số công trình đã xuất hiện để cải thiện hệ thống đề xuất với kỹ thuật điển hình là Autoencoder [20]. Các nghiên cứu tiêu biểu phải kể đến như: Diana Ferreira và các công sự 2020 [18], AutoRec do Ting-Hsiang Wang và các cộng sự [19].

Tổng kết lại có 2 cách chính để áp dụng Autoencoder vào hệ thống tư vấn:

- Trực tiếp ước tính các giá trị bị thiếu trong ma trận tiện ích bằng cách sử dụng lớp tái xây dựng (reconstruction layer).
- Được sử dụng để biểu diễn đặc trưng kết hợp giảm số chiều của chúng thông qua lớp bottleneck và tái tạo đầu ra với số chiều nhỏ hơn đầu vào.

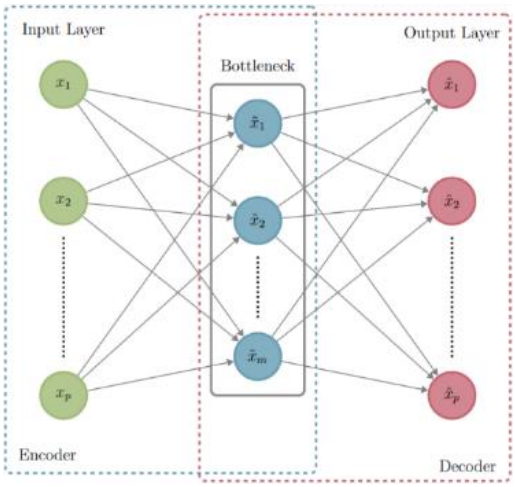
❖ Kiến trúc mạng Autoencoder cổ điển bao gồm 3 phần chính:

1. **Encoder**: Module có nhiệm vụ nén dữ liệu đầu vòa thành một biểu diễn được mã hóa, thường nhỏ hơn một vài bậc so với dữ liệu đầu vào.
2. **Bottleneck**: Module chứa các biểu diễn tri thức được nén (chính là đầu ra của Encoder), đây là phần quan trọng nhất của mạng bới nó mang đặc trưng của đầu vào, có thể dùng để tái tạo lại dữ liệu, lấy đặc trưng của dữ liệu.
3. **Decoder**: Module giúp mạng giải nén các biểu diễn tri thức và tái cấu trúc lại dữ liệu từ dạng mã hóa của nó, mô hình học dựa trên việc so sánh đầu ra của Decoder với đầu vào ban đầu (Input của Encoder).

❖ Cách thức hoạt động:

Cả bộ mã hóa và bộ giải mã đều là các mạng nơ ron chuyển tiếp được kết nối đầy đủ, về cơ bản là các ANN. Bottleneck là một lớp duy nhất với kích thước khá nhỏ. Số nút trong các lớp (kích thước Encoder và Decoder) là một siêu tham số được đặt trước khi huấn luyện cho mô hình Autoencoder. Cụ thể hơn, đầu tiên đầu vào đi qua bộ mã hóa, là một ANN được kết nối đầy đủ dùng để tạo mã. Bộ giải mã, có cấu trúc ANN tương tự, sau đó tạo đầu ra chỉ bằng cách đảo ngược bộ mã hóa. Mục tiêu

là để có được một đầu ra giống với đầu vào. Nhưng tùy bài toán cụ thể mà input và output sẽ giống về nội dung chứ giá trị thì không hẳn, ví dụ như làm mờ ảnh.



Hình 1.5: Shallow Autoencoder

Kiến trúc đơn giản nhất với chỉ một lớp ẩn được mô tả trong Hình 1.5, trình bày sơ đồ của một bộ mã hóa tự động với một lớp ẩn duy nhất (shallow autoencoder).

Tập dữ liệu không gán nhãn $D = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}$, giả sử rằng chiều của bottleneck m là đủ nhỏ so với số chiều của đầu vào và đầu ra p ($m < p$). Và nếu autoencoder đã được đào tạo mang lại $\mathbf{x} = \hat{\mathbf{x}}$ thì điều đó có nghĩa là ta cũng đã có hàm giảm chiều dữ liệu. Ví dụ trong thực nghiệm của đề án này ta có $p = 1682$ và $m = 20$. Với encoder được huấn luyện, ta có thể chuyển đổi mỗi user với vectơ chứa 1682 đặc trưng thành vectơ nhỏ hơn nhiều với kích thước 20. Với decoder sau khi huấn luyện, chỉ cần chuyển đổi ngược lại và nhận được giá trị gần đúng với giá trị gốc. Lựa chọn $m = 20$ ngụ ý rằng hệ số nén vào khoảng 84.

Có thể coi autoencoder như một hàm $f_\theta: \mathbb{R}^p \to \mathbb{R}^p$, trong đó θ là tham số có thể huấn luyện. Các tham số θ có ảnh hưởng tới hoạt động của hàm như sau: $f_\theta(\mathbf{x}^*) \approx \mathbf{x}^*$ trong đó $\mathbf{x}^*$ là một dữ liệu bất kỳ như dữ liệu trong tập huấn luyện (seen data) hoặc dữ liệu trong tập test (unseen data).

Xây dựng hàm mất mát dựa trên bình phương khoảng cách Euclid:

$$\text{Vì} \quad L_i(\theta) = \left\|\mathbf{x}^{(i)} - f_\theta(\mathbf{x}^{(i)})\right\|_2^2 \Longrightarrow L(\theta; D) = \frac{1}{n}\sum_{i=1}^{n} \left\|\mathbf{x}^{(i)} - f_\theta(\mathbf{x}^{(i)})\right\|_2^2 \qquad (1.31)$$

trong đó $L_i(\theta)$ là thước đo sai số của đầu vào $\mathbf{x}^{(i)}$ và đầu ra $\hat{\mathbf{x}}^{(i)} = f_\theta(\mathbf{x}^{(i)})$.

Có thể liên hệ với học có giám sát trong đó chúng ta có thể so sánh nhãn huấn luyện với nhãn dự đoán, thì autoencoder tuy thuộc nhóm học không giám sát nhưng ta có thể coi đầu vào như là nhãn $y^{(i)}$ của học có giám sát.

$f_\theta(\cdot)$: hàm encoder được ký hiệu là $f^{[1]}_{\theta^{[1]}}(\cdot)$ và hàm decoder được ký hiệu là $f^{[2]}_{\theta^{[2]}}(\cdot)$, trong đó $\theta^{[1]}$ là các tham số của encoder và $\theta^{[2]}$ là các tham số của decoder:

$$\hat{\mathbf{x}} = f_\theta(\mathbf{x}) = \left(f^{[1]}_{\theta^{[1]}} \circ f^{[2]}_{\theta^{[2]}}\right)(\mathbf{x}) = f^{[2]}_{\theta^{[2]}}\left(f^{[1]}_{\theta^{[1]}}(\mathbf{x})\right) \tag{1.32}$$

Trong trường hợp bài toán chỉ với một lớp ẩn như hình 1.5 ta có thể xác định công thức cụ thể như sau:

$$\begin{aligned} f^{[1]}_{\theta^{[1]}}(\mathbf{u}) &= \mathbf{S}^{[1]}\left(\mathbf{b}^{[1]} + \mathbf{W}^{[1]}\mathbf{u}\right) \quad \text{với} \quad \mathbf{u} \in \mathbb{R}^p \quad \text{(Encoder)} \\ f^{[2]}_{\theta^{[2]}}(\mathbf{u}) &= \mathbf{S}^{[2]}\left(\mathbf{b}^{[2]} + \mathbf{W}^{[2]}\mathbf{u}\right) \quad \text{với} \quad \mathbf{u} \in \mathbb{R}^m \quad \text{(Decoder)} \end{aligned} \tag{1.33}$$

trong đó: Nhóm encoder có các tham số $\theta^{[1]}$ đã bao gồm vectơ bias $\mathbf{b}^{[1]} \in \mathbb{R}^m$ và ma trận trọng số $\mathbf{W}^{[1]} \in \mathbb{R}^{m\times p}$, $\mathbf{S}^{[1]}(\cdot)$ là vectơ các hàm kích hoạt với $\mathbf{S}^{[1]}: \mathbb{R}^m \to \mathbb{R}^m$. Nhóm decoder có các tham số $\theta^{[2]}$ đã bao gồm vectơ bias $\mathbf{b}^{[2]} \in \mathbb{R}^p$ và ma trận trọng số $\mathbf{W}^{[2]} \in \mathbb{R}^{p\times m}$, $\mathbf{S}^{[2]}(\cdot)$ là vectơ các hàm kích hoạt với $\mathbf{S}^{[2]}: \mathbb{R}^p \to \mathbb{R}^p$. Vì vậy, danh sách toàn bộ các tham số cho mạng autoencoder như sau:

$$\theta = \left(\mathbf{b}^{[1]}, \mathbf{W}^{[1]}, \mathbf{b}^{[2]}, \mathbf{W}^{[2]}\right) \tag{1.34}$$

Làm rõ vectơ các hàm kích hoạt $\mathbf{S}^{[1]}(\cdot)$ và $\mathbf{S}^{[2]}(\cdot)$, ta khái quát thành $\mathbf{S}^{[l]}(\cdot)$ được xây dựng bởi các hàm kích hoạt vô hướng $\sigma^{[l]}: \mathbb{R} \to \mathbb{R}$ với $l = 1{,}2$ chẳng hạn như hàm sigmoid, ReLU. Có $\mathbf{S}^{[l]}(\mathbf{z})$ được xác định bởi $\sigma^{[l]}(\cdot)$ trên mỗi phần tử của $\mathbf{z}$ bằng phép toán element-wise:

$$\mathbf{S}^{[l]}(\mathbf{z}) = \begin{bmatrix} \sigma^{[l]}(z_1) \\ \vdots \\ \sigma^{[l]}(z_r) \end{bmatrix} \tag{1.35}$$

Tính giá trị cho các nơ-ron bottleneck $\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_m$ và nơ-ron đầu ra $\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_p$ với $\mathbf{a}_k = \hat{\mathbf{x}}_k$, $\mathbf{a}$ đại diện cho giá trị đã có hàm kích hoạt.

$$\begin{aligned} \tilde{\mathbf{x}}_i &= \sigma^{[1]}\left(b^{[1]}_i + \sum\nolimits_{k=1}^{p} \mathbf{W}^{[1]}_{ik}\mathbf{x}_k\right) \quad \text{với} \quad i = 1, \dots, m \\ \hat{\mathbf{x}}_j &= \sigma^{[2]}\left(b^{[2]}_j + \sum\nolimits_{k=1}^{m} \mathbf{W}^{[2]}_{jk}\mathbf{a}_k\right) \quad \text{với} \quad j = 1, \dots, p \end{aligned} \tag{1.36}$$

Viết lại hàm kích hoạt (1.31):

$$L(\theta; D) = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}^{(i)} - \underbrace{\mathbf{S}^{[2]}(W^{[2]}\mathbf{S}^{[1]}(\mathbf{W}^{[1]}\mathbf{x}^{(i)} + \mathbf{b}^{[1]}) + \mathbf{b}^{[2]})}_{f_\theta(\mathbf{x}^{(i)})}\|_2^2 \quad (1.37)$$

1.7.2.5. Lọc cộng tác phân tích giá trị suy biến

Trong phần này, hệ tư vấn được xây dựng dựa trên phương pháp phân tích giá trị suy biến [11] còn được viết tắt là SVD thuộc nhóm các phương pháp matrix factorization. Đây là một phương pháp cổ điển từ đại số tuyến tính đã trở nên phổ biến trong lĩnh vực khoa học dữ liệu và học máy. SVD là một kỹ thuật phân tích ma trận thành tích của nhiều ma trận giúp giảm chiều dữ liệu mà không yêu cầu là ma trận vuông như lý thuyết chéo hóa ma trận (Diagonalizable matrix) đã đề cập.

Cho ma trận $\mathbf{A}$ kích thước $m \times n$, SVD của $\mathbf{A}$ được tính theo công thức sau:

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}\mathbf{V}^T_{n\times n} \quad (1.38)$$

Có hai điểm quan trọng cần đảm bảo cho phương trình (1.38):

- Thứ nhất là số chiều của mỗi ma trận: $\mathbf{U} \in \mathbb{R}^{m\times m}$, $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$, $\mathbf{V} \in \mathbb{R}^{m\times n}$. Theo lý thuyết phân rã trị riêng (Eigendecomposition) các ma trận vế phải đều là các ma trận vuông có kích thước giống với ma trận ta muốn phân rã. Tuy nhiên trong SVD vì ma trận mục tiêu có thể là ma trận không vuông nên ma trận tạo bởi vế phải có cùng kích cỡ với ma trận mục tiêu.
- Thứ hai là ma trận $\mathbf{U}$ và $\mathbf{V}$ là ma trận trực giao (Orthogonal matrix), $\mathbf{\Sigma}$ là ma trận chéo hình chữ nhật (Rectangular diagonal matrix) cùng kích cỡ ma trận $\mathbf{A}$. Ma trận $\mathbf{\Sigma}$ với tất cả các thành phần có tọa độ $i \neq j$ đều bằng 0, các giá trị trên đường chéo chính $i = j$ được sắp xếp theo thứ tự giảm dần $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0 = 0 = \cdots = 0$. Số lượng các phần tử khác 0 trong $\mathbf{\Sigma}$ chính là hạng của ma trận $\mathbf{A}$ với $r = rank(\mathbf{A})$. Ta có thể biểu diễn lại phương trình (1.38) như sau: $\mathbf{A}_{m\times n} = \mathbf{U}_{m\times r}\mathbf{\Sigma}_{r\times r}\mathbf{V}^T_{r\times n}$.

Có thể thấy rằng sự phân rã này giống như phân rã trị riêng, bằng cách viết lại phương trình (1.38) thông qua phân rã trị riêng $\mathbf{A}^T\mathbf{A}$ và $\mathbf{A}\mathbf{A}^T$. Cụ thể như sau:

- Đối với trường hợp $\mathbf{A}^T\mathbf{A}$:

$$\mathbf{A}^T\mathbf{A} = (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T) = (\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T)(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T) = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T \quad (1.39)$$

Đẳng thức cuối cùng vì là nghịch đảo của ma trận trực giao bằng với chuyển vị của nó. Ta cũng thay thế $\mathbf{\Sigma}^T\mathbf{\Sigma} = \mathbf{\Lambda}$ để phương trình (1.39) đơn giản hóa thành:

$$\mathbf{A}^T\mathbf{A} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1} \quad (1.40)$$

Kết quả thu được sau khi phân rã trị riêng một ma trận các vectơ độc lập tuyến tính bằng hạng với ma trận ban đầu, một ma trận chéo và ma trận nghịch đảo của nó. Thật vậy, dễ dàng nhận thấy phương trình (1.40) là phân tích trị riêng của ma trận $\mathbf{A}^T\mathbf{A}$. Vì ma trận $\mathbf{A}$ không nhất thiết phải vuông nên ta tính $\mathbf{A}^T\mathbf{A}$ cho ra một ma trận vuông, hơn nữa còn là một ma trận đối xứng và hiển nhiên là một ma trận bán xác định dương vì thỏa mãn $\mathbf{x}^T\mathbf{A}\mathbf{A}^T\mathbf{x} \geq 0$ (với $\mathbf{x}$ là một vectơ cột bất kỳ). Hệ quả là khi thực hiện phép phân tích trị riêng sẽ thu về các giá trị riêng không âm ($\lambda \geq 0$).

- Đối với trường hợp $\mathbf{A}\mathbf{A}^T$:

$$\mathbf{A}\mathbf{A}^T = (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T = (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)(\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T) = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T \quad (1.41)$$

Nhận thấy sự tương đồng giữa cả hai trường hợp, giống như ma trận $\mathbf{V}$ ma trận $\mathbf{U}$ cũng là một ma trận trực giao nên chứa các vectơ riêng của ma trận $\mathbf{A}\mathbf{A}^T$. Chỉ có duy nhất sự khác biệt giữa hai ma trận trên là chiều của chúng. Trong $\mathbf{U}$ là ma trận kích cỡ $m \times m$ thì $\mathbf{V}$ là ma trận kích cỡ $n \times n$, sự khác biệt này gây ra bởi ma trận $\mathbf{A}$ không vuông nên ma trận $\mathbf{A}\mathbf{A}^T$ và $\mathbf{A}^T\mathbf{A}$ là không giống nhau. Ma trận tạo bởi $\mathbf{\Sigma}\mathbf{\Sigma}^T$ tuy có kích cỡ khác so với $\mathbf{\Sigma}^T\mathbf{\Sigma}$ nhưng lại có giá trị trên đường chéo chính giống nhau và mọi giá trị khác đều bằng không. Ta cũng thay thế $\mathbf{\Sigma}\mathbf{\Sigma}^T = \Lambda$:

$$\mathbf{A}\mathbf{A}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T = \mathbf{U}\Lambda\mathbf{U}^{-1} \quad (1.42)$$

❖ Áp dụng thuật toán SVD cho bài toán hệ tư vấn:

Nhờ tính chất trên, ta có thể áp dụng cho bất kỳ ma trận nào, dù nó có vuông hay không nhưng tính chất quan trọng thực sự của nó là nén dữ liệu từ việc trích xuất thông tin từ dữ liệu đã cho. Quá trình này còn được gọi là giảm chiều dữ liệu.

Giả sử ta có một ma trận tiện ích như sau:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \quad (1.43)$$

➢ Xác định ma trận $\mathbf{A}\mathbf{A}^T$:

$$\mathbf{A}\mathbf{A}^T = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \begin{bmatrix} 1 & 3 & 4 & 5 & 0 & 0 & 0 \\ 1 & 3 & 4 & 5 & 2 & 0 & 1 \\ 1 & 3 & 4 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 5 & 2 \\ 0 & 0 & 0 & 0 & 4 & 5 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 9 & 12 & 15 & 2 & 0 & 1 \\ 9 & 27 & 36 & 45 & 6 & 0 & 3 \\ 12 & 36 & 48 & 60 & 8 & 0 & 4 \\ 15 & 45 & 60 & 75 & 10 & 0 & 5 \\ 2 & 6 & 8 & 10 & 36 & 40 & 18 \\ 0 & 0 & 0 & 0 & 40 & 50 & 20 \\ 1 & 3 & 4 & 5 & 18 & 20 & 9 \end{bmatrix} \quad (1.44)$$

Tìm giá trị riêng của ma trận bằng cách tìm nghiệm của đa thức đặc trưng:

$$\det(\mathbf{A}\mathbf{A}^T - \lambda\mathbf{I}) = \lambda^7 - 248\lambda^6 + 14530\lambda^5 - 25500\lambda^4 = 0 \quad (1.45)$$

Ta thu được $\lambda_1 \approx 155.78$, $\lambda_2 \approx 90.41$ và $\lambda_3 \approx 1.81$, suy ra ma trận chéo Λ (trong ví dụ này sử dụng $\mathbf{A}_{m\times n} = \mathbf{U}_{m\times r}\mathbf{\Sigma}_{r\times r}\mathbf{V}^T_{r\times n}$ với $r = 3$):

$$\Lambda = \begin{bmatrix} 155.78 & 0 & 0 \\ 0 & 90.51 & 0 \\ 0 & 0 & 1.81 \end{bmatrix} \quad (1.46)$$

Có thể nhận thấy rằng giá trị suy biến của $\mathbf{A}$ là căn bậc hai giá trị riêng tương đương với ma trận $\mathbf{A}\mathbf{A}^T$, $\sigma_1 \approx 12.48$, $\sigma_2 \approx 9.51$ và $\sigma_3 \approx 1.35$:

$$\mathbf{\Sigma} = \begin{bmatrix} 12.48 & 0 & 0 \\ 0 & 9.51 & 0 \\ 0 & 0 & 1.35 \end{bmatrix} \quad (1.47)$$

Dễ dàng tìm được ma trận trực giao $\mathbf{U}$:

$$\mathbf{U} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \quad (1.48)$$

- Xác định ma trận $\mathbf{A}^T\mathbf{A}$:

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} 1 & 3 & 4 & 5 & 0 & 0 & 0 \\ 1 & 3 & 4 & 5 & 2 & 0 & 1 \\ 1 & 3 & 4 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 5 & 2 \\ 0 & 0 & 0 & 0 & 4 & 5 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 51 & 51 & 51 & 0 & 0 \\ 51 & 56 & 51 & 10 & 10 \\ 51 & 51 & 51 & 0 & 0 \\ 0 & 10 & 0 & 45 & 45 \\ 0 & 10 & 0 & 45 & 45 \end{bmatrix} \quad (1.49)$$

Thực hiện tương tự như trên, ta cũng tìm được ma trận trực giao $\mathbf{V}$:

$$\mathbf{V} = \begin{bmatrix} 0.56 & 0.12 & 0.40 \\ 0.59 & -0.02 & -0.80 \\ 0.56 & 0.12 & 0.40 \\ 0.09 & -0.69 & 0.09 \\ 0.09 & -0.69 & 0.09 \end{bmatrix} \quad (1.50)$$

- SVD của ma trận $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$:

$$\mathbf{A} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \begin{bmatrix} 12.48 & 0 & 0 \\ 0 & 9.51 & 0 \\ 0 & 0 & 1.35 \end{bmatrix} \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix} \tag{1.51}$$

Có thể biểu diễn lại công thức (1.38) bằng các ma trận có $rank = 1$ như sau:

$$\mathbf{A} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_k \mathbf{u}_k \mathbf{v}_k^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T \tag{1.52}$$

Chìa khóa để giảm kích thước là một vài cột của **U**, giá trị riêng tương ứng trong **Σ** và một vài hàng tương đương của $\mathbf{V}^T$ chứa hầu hết các thông tin của ma trận **A**. Ta thấy được giá trị riêng trong **Σ** nhỏ dần nghĩa là đóng góp trong **A** cũng ít dần đi. Nói cách khác ta có thể thu được một ma trận xấp xỉ **A** bằng cách trích xuất một số cột và hàng đầu tiên của từng thành phần trong SVD.

$$\begin{aligned} \mathbf{A} \approx \mathbf{A}_{k=2} &= \begin{bmatrix} 0.13 & 0.02 \\ 0.41 & 0.07 \\ 0.55 & 0.09 \\ 0.68 & 0.11 \\ 0.15 & -0.59 \\ 0.07 & -0.73 \\ 0.07 & -0.29 \end{bmatrix} \begin{bmatrix} 12.48 & 0 \\ 0 & 9.51 \end{bmatrix} \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \end{bmatrix} \\ &= \begin{bmatrix} 0.93 & 0.95 & 0.93 & 0.01 & 0.01 \\ 2.95 & 3.01 & 2.95 & 0.00 & 0.00 \\ 3.95 & 4.03 & 3.95 & 0.03 & 0.03 \\ 4.88 & 4.99 & 4.88 & 0.04 & 0.04 \\ 0.38 & 1.22 & 0.38 & 4.04 & 4.04 \\ -0.34 & 0.65 & -0.34 & 4.87 & 4.87 \\ 0.16 & 0.57 & 0.16 & 1.98 & 1.98 \end{bmatrix} \end{aligned} \tag{1.53}$$

Giá trị **A** thu được từ việc tính từ một phần hàng và cột trong SVD cho giá trị gần đúng với giá trị **A** ban đầu. Phương trình (1.53) được gọi là phân tích giá trị suy biến cắt gọn (truncated SVD) với k giá trị suy biến ($k < r$) được chọn trong khi loại bỏ $r - k$ giá trị suy biến:

$$\mathbf{A} \approx \mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_k \mathbf{u}_k \mathbf{v}_k^T \tag{1.54}$$

Cũng từ ví dụ trên ta dễ dàng nhận thấy khi sử dụng SVD trong trường hợp $m, n \gg r$ sẽ có lợi ích lớn về mặt lưu trữ, đặc biệt với truncated SVD lại càng hiệu quả hơn. Ví dụ trên **A** cần: $7 \times 5 = 35$ ô lưu trữ trong khi $\mathbf{A}_{k=2}$ chỉ cần: $7 \times 2 + 2 \times 2 + 2 \times 5 = 28$ ô lưu trữ.

Kết luận: Chương này đã trình bày tổng quát về định nghĩa hệ tư vấn, phân loại các cách tiếp cận phổ biến và mô tả thêm một số phương pháp phổ biến hay được áp dụng trong thực tế. Hơn nữa, một số ví dụ cụ thể nhằm mô tả hoạt động của các phương pháp cũng được trình bày một cách kỹ lưỡng nhằm biểu diễn tốt cách hoạt động của chúng.

CHƯƠNG II: MÔ HÌNH DỰA TRÊN ĐỒ THỊ VÀ HỌC SÂU

Chương này trình bày quá trình áp dụng đồ thị để đo lường mối quan hệ người dùng dựa trên các độ đo và trích xuất đặc trưng, bộ tự mã hóa đảm nhiệm tác vụ giảm chiều dữ liệu và giải thuật K-means để tiến hành phân nhóm người dùng. Từ đó xây dựng kiến trúc kết hợp cả ba kỹ thuật trên nhằm tái tạo lại mô hình GHRS [21].

Trình bày cơ sở thực nghiệm bao gồm liệt kê thông tin bộ dữ liệu Movielens-100k được sử dụng và phân tích bộ dữ liệu này.

2.1. Cơ sở lý thuyết cho mô hình GHRS

2.1.1. Lựa chọn đặc trưng dựa trên đồ thị

Trong nhiều nhiệm vụ phân tích dữ liệu, chúng ta thường phải đối mặt với dữ liệu có chiều rất lớn đòi hỏi cần có những phương pháp tổng hợp để tìm được mối quan hệ, mức độ liên quan và từ đó cho ra được những đặc trưng và sự tương đồng giữa các nút trong đồ thị. Và để đo lường được sự chênh lệch giữa các nút, nhà nghiên cứu sẽ tiến hành đo lường các hệ số thể hiện tính trung tâm (centrality) của các nút trong mạng đồ thị, nút nào có giá trị càng lớn thì sẽ có vị trí càng cao và sức ảnh hưởng lên các nút khác càng mạnh.

2.1.1.1. PageRank

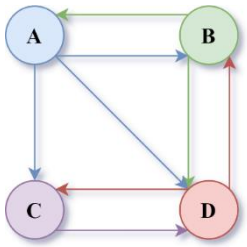
Thuật toán PageRank (PR) được phát minh bởi Serge Brin và Larry Page là công thức toán học đánh giá sự ảnh hưởng của một nút thông qua việc xem xét số lượng của các nút liên kết với nó. Được sử dụng lần đầu tiên để đánh giá tầm quan trọng tương đối của website trong toàn bộ hệ thống World Wide Web hay đơn giản là xếp hạng các trang web nhằm hỗ trợ tác vụ tìm kiếm của Google, thuật toán đã giúp nâng cao chất lượng kết quả tìm kiếm cao hơn rất nhiều so với các công cụ tìm kiếm khác.

Dựa trên bài báo của Zahra và Valipour [21], các tác giả đã xếp hạng các nút bằng việc lặp đi lặp lại việc phân phối giá trị mỗi nút theo tỷ lệ với các nút kết nối với nó. Các nút khác sau đó nhận được giá trị từ nút gốc và quá trình phân phối xếp

hạng sẽ tiếp tục diễn ra cho đến khi hội tụ. Phương pháp này cho ra kết quả tương tự như kỹ thuật Random Walk.

- Thuật toán hoạt động như sau:

  Giả sử rằng có 4 nút trong đồ thị được mô tả bởi hình dưới đây:



Hình 2.1: Đồ thị 4 nút

Ta có giá trị phân phối của mỗi nút tới các nút còn lại như sau:

$$l_A = \left(0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right), \quad l_B = \left(\frac{1}{2}, 0, 0, \frac{1}{2}\right), \quad l_C = (0,0,0,1), \quad l_D = \left(0, \frac{1}{2}, \frac{1}{2}, 0\right). \tag{2.1}$$

Giá trị nhận được của từng nút có thể biểu diễn dưới dạng ma trận truyền năng lượng L và vectơ biến $\mathbf{r}$ đại diện cho giá trị hiện tại của các nút:

$$L = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 1 & 0 \end{bmatrix} \qquad \mathbf{r} = \begin{bmatrix} r_A \\ r_B \\ r_C \\ r_D \end{bmatrix} \tag{2.2}$$

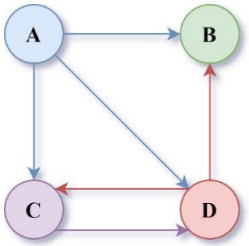
Công thức tổng quát để tính giá trị của nút i tại thời điểm vòng lặp thứ t:

$$r_i^{(t)} = \sum_{j=1}^{n} L_{i,j} \times r_j^{(t-1)} \tag{2.3}$$

Kết hợp với công thức (2.3) ta có thể biểu diễn (2.2) thông qua phương trình $\mathbf{r}^{(t)} = L\mathbf{r}^{(t-1)}$:

$$\begin{aligned} r_A^{(t)} &= \frac{1}{2} r_B^{(t-1)} \\ r_B^{(t)} &= \frac{1}{3} r_A^{(t-1)} + \frac{1}{2} r_D^{(t-1)} \\ r_C^{(t)} &= \frac{1}{3} r_A^{(t-1)} + \frac{1}{2} r_D^{(t-1)} \\ r_D^{(t)} &= \frac{1}{3} r_A^{(t-1)} + \frac{1}{2} r_B^{(t-1)} + r_C^{(t-1)} \end{aligned} \tag{2.4}$$

Tuy nhiên công thức (2.3) gặp phải một vấn đề có tên là đường cụt (dead ends), nghĩa là trong quá trình tính toán khi đồ thị đầu vào xuất hiện nút không có liên kết đi ra (links out) khỏi nút đó và giá trị bị kẹt tại đó và khiến giá trị một vài nút hoặc tất cả các nút sau khi thực hiện tính toán lặp nhiều lần tiến dần về 0.



Hình 2.2: Đồ thị 4 nút chứa đường cụt

Giả sử rằng đồ thị trong hình 2.1 với nút B không có cạnh ra được mô tả trong hình 2.2, ta có giá trị phân phối của chúng là:

$$l_A = \left(0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right), \quad l_B = (0,0,0,0), \quad l_C = (0,0,0,1), \quad l_D = \left(0, \frac{1}{2}, \frac{1}{2}, 0\right). \tag{2.5}$$

Biểu diễn ma trận truyền năng lượng L và vectơ biến $\mathbf{r}$ tại thời điểm $t = 0$:

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 0 & 1 & 0 \end{bmatrix} \qquad \mathbf{r}^{(0)} = \begin{bmatrix} r_A^{(0)} \\ r_B^{(0)} \\ r_C^{(0)} \\ r_D^{(0)} \end{bmatrix} = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix} \tag{2.6}$$

Tiếp tục thực hiện tính toán và sau lần lặp thứ k dễ dàng thu được kết quả sau:

$$\mathbf{r}^{(1)} = \begin{bmatrix} 0 \\ 5/24 \\ 5/24 \\ 1/3 \end{bmatrix}, \quad \mathbf{r}^{(2)} = \begin{bmatrix} 0 \\ 1/6 \\ 1/6 \\ 5/24 \end{bmatrix}, \quad \ldots \quad \mathbf{r}^{(k)} \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.7}$$

Để giải quyết vấn đề này, một khái niệm quan trọng trong PR gọi là “damping factor” được sử dụng trong quá trình lặp. Nói cách khác, để tổng giá trị các nút không bị biến mất khi số lần lặp tăng, một giá trị xác suất để chuyển đổi ngẫu nhiên sẽ được thêm vào. Ta viết lại công thức (2.3):

$$r_i^{(t)} = \frac{1-d}{n} + d \sum_{j=1}^{n} L_{i,j} \times r_j^{(t-1)} \tag{2.8}$$

trong đó d là hệ số damping factor nằm trong miền $[0, 1]$, trong thực tế người ta thường đặt giá trị mặc định của d là 0.85.

2.1.1.2. Degree Centrality

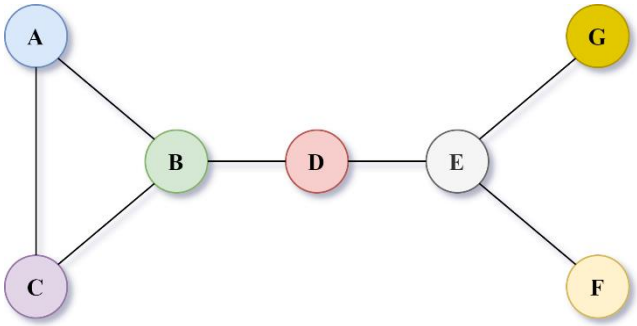
Hệ số trung tâm trực tiếp (Degree Centrality - DC) giúp đo lường được số lượng các mối quan hệ trực tiếp của một nút nào đó với các nút khác trong đồ thị. Giá trị của hệ số này chạy từ 0 đến 1 và khi giá trị tiến gần đến 1 thì tính trung tâm của nút càng lớn, tức là nút này càng có nhiều kết nối với các nút khác trong đồ thị, điều này đồng nghĩa với việc nút đó nắm giữ nhiều thông tin nhất. Ngược lại khi tiến dần về 0 thì nút trở nên phụ thuộc vào các nút khác, mang ít giá trị khai thác hơn.

Theo bài báo của Zahra và Valipour [21], phần cài đặt đã thực hiện phép tính cho đồ thị vô hướng thay vì đồ thị có hướng để phù hợp với dữ liệu đầu vào.

- Công thức tính hệ số này như sau:

$$C_D(u) = \frac{k}{n-1} \tag{2.9}$$

trong đó k là tổng số cạnh liên thuộc hay là tổng số kết nối giữa nút u tới hàng xóm trực tiếp của nó và n là tổng số đỉnh trong đồ thị với $n-1$ là bậc tối đa có thể có của một nút. Hệ số $C_D(u) \in [0, 1]$, khi giá trị càng gần 1 thì tính trung tâm trực tiếp càng lớn, tức là càng nằm ở vị trí trung tâm của đồ thị và ngược lại khi tiến gần về 0.



Hình 2.3: Đồ thị vô hướng với $n = 7$

Ví dụ có đồ thị vô hướng như hình 2.3, ta lập bảng liệt kê hệ số DC dựa trên công thức (2.9) như sau:

Bảng 2.1: Kết quả hệ số trung tâm trực tiếp

Nút (u)	k	$C_D(u)$

A	2	0.3333
B	**3**	**0.5000**
C	2	0.3333
D	2	0.3333
E	**3**	**0.5000**
F	1	0.1667
G	1	0.1667

Với kết quả như trên, dễ thấy nút B và E là hai nút có hệ số DC cao nhất với $C_D(B) = C_D(E) = 0.5$ vượt trội hơn những nút còn lại trong đồ thị. Người ta thường gọi những nút như vậy là Hub của mạng đồ thị.

2.1.1.3. Closeness Centrality

Hệ số trung tâm lân cận (Closeness Centrality - CC) là hệ số tìm các nút có mối liên hệ "gần" nhất với một nút nào đó trong mạng đồ thị hay cụ thể hơn là tìm những nút có thể chia sẻ thông tin hiệu quả. Một nút như vậy là quan trọng khi nó nằm trên đường đi ngắn nhất tới tất cả các nút còn lại. Hệ số này khắc phục điểm yếu của hệ số DC khi chỉ xét đến mối quan hệ trực tiếp mà bỏ qua tiếp xúc gián tiếp giữa các nút, một nút có hệ số DC cao không có nghĩa là nút đó là nút "gần" nhất với mọi thành viên trong mạng đồ thị.

Để tính toán hệ số này, cần phải tính tổng số bước (step) của các đoạn đường ngắn nhất (shortest path hoặc geodesic path) mà nút phải đi để đến được tất cả các nút khác trong mạng đồ thị. Phương pháp được sử dụng để tìm đường đi ngắn nhất khi sử dụng hàm *closeness_centrality()* của thư viện networkx được đặt mặc định là thuật toán Dijkstra, trong khi các thuật toán thông dụng khác như Bellman-Ford, Floyd-Warshall hay Johnson vẫn chưa được hỗ trợ.

- Công thức tính hệ số này như sau:

$$C_C(u) = \frac{n-1}{\sum_{v=1}^{n-1} d(v,u)} \tag{2.10}$$

trong đó n là tổng số nút trong mạng với $n-1$ là tổng số nút không phải nút u, $d(v,u)$ là đoạn đường ngắn nhất từ nút v tới nút u. Hệ số $C_C(u) \in [0,1]$, càng đến gần 1 thì nút càng gần với những nút còn lại trong đồ thị hay số bước cần để tới các nút khác càng ngắn.

Dựa trên đồ thị như hình 2.3, ta thu được bảng giá trị:

Bảng 2.2: Kết quả hệ số trung tâm lân cận

Nút (u)	$\sum_{v=1}^{n-1} d(v,u)$	$C_C(u)$
A	15	0.4000
B	11	0.5454
C	15	0.4000
D	**10**	**0.6000**
E	11	0.5454
F	16	0.3750
G	16	0.3750

Từ bảng 2.2, rõ ràng thấy được nút D có số bước phải đi ít nhất với chỉ 10 với hệ số CC cao nhất với $C_C(D) = 0.6$ trong khi hai nút B và E là hai nút có hệ số DC cao nhất lúc này chỉ có $C_C(B) = C_C(E) = 0.5454$.

2.1.1.4. Betweenness Centrality

Định nghĩa chính thức về hệ số trung tâm trung gian (Betweenness Centrality - BC) được đề xuất đầu tiên bởi Linton Freeman, là một trong những thước đo tính trung tâm (Centrality) quan trọng trong mạng đồ thị. Hệ số mô tả một nút bất kỳ có thể mang giá trị C_D và C_C thấp nhưng lại có ý nghĩa cầu nối giữa các nút khác trong đồ thị hay đơn giản là thông tin trao đổi cần đi qua nó.

- Công thức tính hệ số này như sau:

$$C_B(u) = \sum_{\substack{s,t \in V \\ s \neq t \neq u}} \frac{\sigma(s,t|u)}{\sigma(s,t)} \tag{2.11}$$

trong đó V là tập các nút với $u \notin V$, $\sigma(s,t)$ là số đường đi ngắn nhất từ s tới t và $\sigma(s,t|u)$ số đường đi ngắn nhất có đi qua nút u.

Trong trường hợp tốt nhất, u luôn nằm trên mọi đường đi ngắn nhất từ s tới t, vì vậy ta có $\frac{\sigma(s,t|u)}{\sigma(s,t)} = 1$. Giá trị lớn nhất mà $C_B(u)$ có thể đạt được trong đồ thị có hướng:

$$C_B(u) = \sum_{\substack{s,t \in V \\ s \neq t \neq u}} \frac{\sigma(s,t|u)}{\sigma(s,t)} = \sum_{\substack{s,t \in V \\ s \neq t \neq u}} 1 = 2 \times \binom{n-1}{2} = (n-1)(n-2) \tag{2.12}$$

Để chuẩn hóa công thức (2.11) và căn cứ vào kết quả thu được từ phương trình (2.12), bằng cách chia cho tổng số cạnh có thể có trong đồ thị bằng cách áp dụng tổ hợp, ta có thể giới hạn giá trị $C_B(u)$ trong đoạn $[0, 1]$:

- Chuẩn hóa khi đồ thị vô hướng:

$$C_B(u)_{normalized} = \frac{2}{(n-1)(n-2)} \sum_{\substack{s,t \in V \\ s \neq t \neq u}} \frac{\sigma(s,t|u)}{\sigma(s,t)} \tag{2.13}$$

- Chuẩn hóa khi đồ thị có hướng:

$$C_B(u)_{normalized} = \frac{1}{(n-1)(n-2)} \sum_{\substack{s,t \in V \\ s \neq t \neq u}} \frac{\sigma(s,t|u)}{\sigma(s,t)} \tag{2.14}$$

2.1.1.5. Load Centrality

Một thước đo khác về tính trung tâm được gọi là Load Centrality (LC), một hệ số khá tương tự như hệ số BC. Tuy nhiên, thay vì dựa vào số lượng đoạn đường ngắn nhất, LC ước tính lưu lượng trên một nút, giả sử rằng lưu lượng được phân bố công bằng giữa tất cả các đoạn đường ngắn nhất của đồ thị. Hệ số LC hội tụ về hệ số BC trong nhiều trường hợp thực tế.

- Công thức tính hệ số này như sau:

$$C_L(u) = \sum_{\substack{s,t \in V \\ s \neq t \neq u}} \theta_{s,t}(u) \tag{2.15}$$

trong đó V là tập các nút với $u \notin V$, $\theta_{s,t}$ là năng lượng truyền từ nút s tới nút t thông qua đường đi ngắn nhất giữa hai đỉnh, trong trường hợp có nhiều bước thì năng lượng được chia đều cho các nút trên đường đi. $\theta_{s,t}(u)$ là giá trị mà đỉnh u thu được từ quá trình truyền năng lượng từ nút s tới nút t.

Khi trường hợp tốt nhất xảy ra, u luôn nằm trên mọi đường đi ngắn nhất từ s tới t và là duy nhất, vì vậy ta có $\theta_{s,t}(u) = 1$. Thực hiện tương tự như hệ số BC để tìm giá trị $C_L(u)$ được chuẩn hóa.

- Chuẩn hóa khi đồ thị vô hướng:

$$C_L(u)_{normalized} = \frac{2}{(n-1)(n-2)} \sum_{\substack{s,t \in V \\ s \neq t \neq u}} \theta_{s,t}(u) \tag{2.16}$$

- Chuẩn hóa khi đồ thị có hướng:

$$C_L(u)_{normalized} = \frac{1}{(n-1)(n-2)} \sum_{\substack{s,t \in V \\ s \neq t \neq u}} \theta_{s,t}(u) \quad (2.17)$$

2.1.1.6. Average Neighbor Degree

Giá trị của mỗi nút có thể tính bằng cách lấy trung bình của tổng bậc các nút hàng xóm của nút đó.

- Công thức tính hệ số này như sau:

$$AND(u) = \frac{1}{N(u)} \sum_{v \in N(u)} k_v \quad (2.18)$$

trong đó $N(u)$ là tập hợp các hàng xóm trực tiếp của nút u và k_v là bậc của nút v với điều kiện v phải thuộc $N(u)$.

Đối với những đồ thị có trọng số (Weighted graph), một công thức tính hệ số tương tự (2.18):

$$AND^w(u) = \frac{1}{s_u} \sum_{v \in N(u)} w_{uv} k_v \quad (2.19)$$

trong đó s_u là weighted degree của nút u nghĩa là tổng trọng số các cạnh kết nối trực tiếp u với các nút hàng xóm $N(u)$, w_{uv} là trọng số của cạnh nối giữa u và v.

2.1.2. Autoencoder

2.1.2.1. Autoencoder denoising

Khác với mô hình cơ bản trong hình 1.5, mô hình này học cách loại bỏ nhiễu trong lớp tái xây dựng từ dữ liệu đầu vào bị nhiễu. Nó nhận đầu vào bị hỏng một phần và học cách khôi phục đầu vào đã khử nhiễu ban đầu. Về mặt kiến trúc, autoencoder denoising có kiến trúc tương tự autoencoder cổ điển, tuy nhiên điểm khác biệt chính là trong quá trình huấn luyện, mô hình được đào tạo trên mẫu bị nhiễu và hàm mất mát đã được sửa tương ứng.

Đầu tiên, đầu vào ban đầu **x** bị biến đổi thành $\tilde{\mathbf{x}}$ thông qua một số phép ánh xạ ngẫu nhiên ví dụ như Gussian noise, masking noise hay salt-and-pepper noise. Vì vậy công thức (1.32) được biến đổi thành:

$$\widehat{\tilde{\mathbf{x}}} = f_\theta(\tilde{\mathbf{x}}) = \left(f_{\theta^{[1]}}^{[1]} \circ f_{\theta^{[2]}}^{[2]}\right)(\tilde{\mathbf{x}}) = f_{\theta^{[2]}}^{[2]}\left(f_{\theta^{[1]}}^{[1]}(\tilde{\mathbf{x}})\right) \quad (2.20)$$

Tiếp theo cần sửa hàm mất mát (1.31), ta so sánh giữa $\mathbf{x}$ với $\widehat{\tilde{\mathbf{x}}}$ thay vì $\tilde{\mathbf{x}}$ với $\widehat{\tilde{\mathbf{x}}}$:

$$L(\theta; D) = \frac{1}{n}\sum_{i=1}^{n} \left\|\mathbf{x}^{(i)} - f_\theta(\tilde{\mathbf{x}}^{(i)})\right\|_2^2 \quad (2.21)$$

Vì vậy trong quá trình huấn luyện, ta tìm được các tham số θ để cố gắng loại bỏ nhiễu càng nhiều càng tốt.

2.1.2.2. Hồi quy ElasticNet

Một mô hình hồi quy kết hợp của hồi quy Lasso và hồi quy Ridge giúp cân bằng trong việc lựa chọn các đặc trưng (điều này giải quyết vấn đề đa cộng tuyến) nhưng cũng không làm mất tính ổn định của mô hình. Kết hợp với (2.21), ta có:

$$L(\theta; D) = \frac{1}{n}\sum_{i=1}^{n} \left\|\mathbf{x}^{(i)} - f_\theta(\tilde{\mathbf{x}}^{(i)})\right\|_2^2 + \alpha_1 \sum_{i=1}^{n} |\theta_i| + \alpha_2 \sum_{i=1}^{n} \theta_i^2 \quad (2.22)$$

trong đó α_1 và α_2 là hệ số chính quy hóa lần lượt của hồi quy Lasso và Ridge.

2.1.3. Phân cụm người dùng

Dựa theo đề xuất của Zahra và Valipour [21], mỗi người dùng sẽ thuộc một cụm nào đó và mỗi phân cụm được coi là xếp hạng ước tính cho nhóm người dùng đó. Căn cứ vào kết quả thu được sau quá trình huấn luyện thông qua mạng học sâu Autoencoder để thực hiện phân loại bằng thuật toán phân cụm K-means. Một vấn đề nữa được đặt ra là làm sao tìm được số cụm k thích hợp, ở đây hai tác giả đã sử dụng phương pháp Elbow và Average Silhouette để giải quyết vấn đề nay.

2.1.3.1. K-means

Thuật toán phân cụm K-means (K-means clustering) là một giải thuật của học không giám sát. Trong đó ta không biết được nhãn của từng điểm dữ liệu. Mục đích là làm sao để chia dữ liệu thành các cụm (cluster) khác nhau sao cho dữ liệu trong cùng một cụm có những tính chất giống nhau, hiểu theo một cách hình học là khoảng cách của các thành viên trong cụm tới tâm cụm (centroid) của chúng phải là gần hơn so với tâm của các cụm khác, kiểu phân chia này trong toán học được gọi là sơ đồ Voronoi.

Dựa trên mục 1.3 và hình 1.4, ta có ma trận đặc trưng-người dùng $\boldsymbol{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_N] \in \mathbb{R}^{d \times N}$ trong đó d là số đặc trưng hay số chiều của vectơ cột $\mathbf{w}$, N là số lượng người dùng và K là số cụm được xác định trước (sẽ được ước tính tại mục 2.1.3.2 và 2.1.3.3) với điều kiện $K < N, K \in \mathbb{N}^*$. Nhiệm vụ là tìm vị trí các tâm cụm $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_K \in \mathbb{R}^{d \times 1}$ và nhãn (label) của mỗi người dùng. Mã hóa one-hot được sử dụng để biểu diễn nhãn của người dùng thứ i: $\mathbf{w}_i$ với vectơ $\mathbf{y}_i \in \mathbb{R}^{1 \times K}$ trong đó tất cả phần tử của $\mathbf{y}_i = 0$ trừ phần tử y_{ik} là nhãn đúng của $\mathbf{w}_i$ có giá trị bằng 1 với $k \in \{1, 2, \dots, K\}$ nghĩa là $\mathbf{w}_i$ thuộc cụm thứ k có $\mathbf{m}_k$ là tâm cụm, phần tử thuộc ma trận nhãn $\mathbf{Y} \in \mathbb{R}^{N \times K}$. Có thể biểu diễn ràng buộc cho $\mathbf{y}_i$ như sau:

$$y_{ij} \in \{0, 1\} \ \& \ \sum_{j=1}^{K} y_{ij} = 1 \tag{2.20}$$

- **Bước 1: Xây dựng hàm mất mát**

Để kiểm tra sai số của vectơ $\mathbf{w}_i$ khi được phân vào cụm thứ k, khoảng cách Euclid được áp dụng với $y_{ik} = 1$ và $y_{ij} = 0, \forall j \neq k$. Công thức tính được trình bày nhau sau:

$$\|\mathbf{w}_i - \mathbf{m}_k\|_2^2 = y_{ik}\|\mathbf{w}_i - \mathbf{m}_k\|_2^2 = \sum_{j=1}^{K} y_{ij}\|\mathbf{w}_i - \mathbf{m}_j\|_2^2 \tag{2.21}$$

Hàm mất mát hay sai số trung bình cho toàn bộ dữ liệu:

$$\mathcal{L}(\mathbf{Y}, \mathbf{M}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\left\|\mathbf{w}_i - \mathbf{m}_j\right\|_2^2 \tag{2.22}$$

trong đó $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_K] \in \mathbb{R}^{d \times K}$ là ma trận tạo bởi K tâm cụm và $\mathcal{L}(\mathbf{Y}, \mathbf{M})$ phải thỏa mãn điều kiện được mô tả trong phương trình (2.20).

- **Bước 2: Tối ưu hàm mất mát**

Để tìm được lời giải tối ưu cho phương trình (2.22) là rất khó do tồn tại yêu cầu một số biến nhận giá trị số nguyên, có thể thấy bài toán này thuộc nhóm quy hoạch số nguyên hỗn hợp (Mixed integer programming – MILP hay MIP). Mặc dù hầu hết các dạng bài này đều sử dụng quy trình nhánh và cận (branch-and-bound) để tìm kiếm nghiệm nguyên, tuy nhiên chúng có độ phức tạp rất lớn. Một kỹ thuật phổ biến khác đã được áp dụng trước đó trong mô hình MF cũng phù hợp cho bài toán

này, đơn giản là thực hiện lặp đi lặp quá trình cố định một trong hai biến và giải biến còn lại cho đến khi $\mathcal{L}(\mathbf{Y}, \mathbf{M})$ hội tụ.

- Cố định ma trận tâm cụm $\mathbf{M}$, tìm ma trận nhãn $\mathbf{Y}$

Giả sử rằng đã xác định được các tâm cụm $[\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_K]$, thực hiện tìm nhãn cho mỗi điểm $\mathbf{w}_i$ sao cho hàm mất mát đạt giá trị nhỏ nhất.

Khi cố định các tâm cụm, bài toán trở thành tìm nhãn cho các vectơ $\mathbf{w}_i$ bằng cách tìm nghiệm $\mathbf{y}_i$ để $\mathbf{w}_i$ có khoảng cách tới tâm cụm được gán cho nó là nhỏ nhất:

$$\mathbf{y}_i = \underset{\mathbf{y}_i}{\operatorname{argmin}} \frac{1}{N} \sum_{j=1}^{K} y_{ij} \left\| \mathbf{w}_i - \mathbf{m}_j \right\|_2^2 \tag{2.23}$$

Đơn giản hóa phương trình (2.23) vì chỉ có một phần tử tại vị trí nhãn đúng của vectơ nhãn $\mathbf{y}_i$ mới có giá trị bằng 1 nên ta chỉ cần tìm tâm cụm gần với $\mathbf{w}_i$ nhất:

$$j = \underset{j}{\operatorname{argmin}} \left\| \mathbf{w}_i - \mathbf{m}_j \right\|_2^2 \tag{2.24}$$

Căn cứ vào khoảng cách từ vectơ $\mathbf{w}_i$ tới vectơ tâm cụm $\mathbf{m}_j$, ta có thể kết luận mỗi điểm $\mathbf{w}_i$ thuộc vào nhóm có tâm cụm gần nhất với nó. Từ đó suy ra được nhãn của $\mathbf{w}_i$ và tiếp tục thực hiện trên toàn bộ tập dữ liệu.

- Cố định ma trận nhãn $\mathbf{Y}$, tìm ma trận tâm cụm $\mathbf{M}$

Giả định rằng đã xác định được nhãn $[\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N]$ cho các điểm $\mathbf{w}_i$, thực hiện tìm các tâm cụm mới trong mỗi nhóm để hàm mất mát đại giá trị nhỏ nhất.

Khi cố định các nhãn, bài toán trở thành tìm cụm cho các vectơ $\mathbf{w}_i$ bằng cách tìm nghiệm $\mathbf{m}_j$ để $\mathbf{w}_i$ có khoảng cách tới tâm cụm mới của nó là nhỏ nhất:

$$\mathbf{m}_j = \underset{\mathbf{m}_j}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} y_{ij} \left\| \mathbf{m}_j - \mathbf{w}_i \right\|_2^2 \tag{2.25}$$

Đây là một hàm khả vi liên tục hay đơn giản là có đạo hàm tại mọi điểm nên ta có thể tìm nghiệm $\mathbf{m}_j$ bằng cách đạo hàm phương trình (2.25) và cho bằng không. Dựa trên công thức $\nabla_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = 2\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$, nếu $\mathbf{A}$ là ma trận đơn vị $\boldsymbol{I}$ thì $\nabla_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = 2(\mathbf{x} - \mathbf{b})$. Suy ra ta cần giải phương trình sau:

$$\nabla_{\mathbf{w}_i} \left[\frac{1}{N} \sum_{i=1}^{N} y_{ij} \left\| \mathbf{m}_j - \mathbf{w}_i \right\|_2^2 \right] = \frac{2}{N} \sum_{i=1}^{N} y_{ij} (\mathbf{m}_j - \mathbf{w}_i) = 0 \Leftrightarrow \frac{2}{N} \sum_{i=1}^{N} y_{ij} \mathbf{m}_j = \frac{2}{N} \sum_{i=1}^{N} y_{ij} \mathbf{w}_i$$

$$\Leftrightarrow \mathbf{m}_j \sum_{i=1}^{N} y_{ij} = \sum_{i=1}^{N} y_{ij}\mathbf{w}_i \Leftrightarrow \mathbf{m}_j = \frac{\sum_{i=1}^{N} y_{ij}\mathbf{w}_i}{\sum_{i=1}^{N} y_{ij}} \quad (2.26)$$

Dễ thấy $\sum_{i=1}^{N} y_{ij}$ là tổng số phần tử nằm trong cụm j và $\sum_{i=1}^{N} y_{ij}\mathbf{w}_i$ là tổng giá trị của các điểm thuộc cụm j được biểu diễn dưới dạng vectơ thuộc $\mathbb{R}^{d\times 1}$. Vậy giá trị $\mathbf{m}_j$ cần tìm là một vectơ trung bình cộng của các vectơ $\mathbf{w}$ trong cụm j.

2.1.3.2. Phương pháp Elbow

Tính đến hiện tại đã có nhiều biện pháp để xác định xem số cụm thích hợp để tách nhóm dữ liêu. Sự phân tách đo mức độ khác biệt của một cụm với một cụm khác. Sai số Cluster cohesion có thể được đo bằng Within Cluster Sum of Square (WCSS) và Cluster separation với Between Cluster Sum Squares (BCSS)

$$\text{WCSS} = \sum_{j}^{N_C} \sum_{\mathbf{x}\in C_j} \|\mathbf{x} - \mathbf{m}_j\|_2^2 \quad (2.27)$$

trong đó N_C là tập hợp các cụm, C_j là phân cụm thứ j thuộc tập N_C.

$$\text{BCSS} = \sum_{j}^{N_C} |C_j| \|\bar{\mathbf{m}} - \mathbf{m}_j\|_2^2 \quad (2.28)$$

trong đó $|C_j|$ là số cụm trong N_C và $\bar{\mathbf{m}}$ là tâm cụm của toàn bộ tập dữ liệu.

Trên thực tế phương pháp Elbow không có một thuật toán cụ thể nào cả bởi mục đích của phương pháp là tìm ra "điểm uốn" trên đồ thị trực quan. Đây là tiêu chí chủ quan khi mỗi người sẽ đưa ra những kết luận khác nhau trên cùng một biểu đồ. Tuy là một phương pháp chủ quan nhưng vẫn có khả năng tạo một phương pháp toán cụ thể để phát hiện điểm "khuỷu tay" bằng cách tìm tiếp tuyến với đường cong và phải song song với đường thẳng nối hai điểm cuối của đường cong. Đường cong ở đây được tạo bởi các sai số cluster cohesion thu được sau khi lặp đi lặp lại việc tính WCSS sau mỗi lần tăng số cụm K, $\lim_{K\to N} \text{WCSS} = 0$. Tuy nhiên phương pháp này vẫn chưa đủ để xác định liệu số cụm có tốt hay không, đặc biệt khi đồ thị có độ cong thấp.

2.1.3.3. Phương pháp Silhouette

Phương pháp này sử dụng hệ số silhouette là sự kết hợp của Cluster cohesion và Cluster separation. Kết quả mong muốn khi hệ số silhouette trả về càng cao càng tốt.

$$\text{Nếu WCSS} < \text{BCSS}: \quad s = 1 - \frac{\text{WCSS}}{\text{BCSS}}, \quad \text{ngược lại:} \quad s = \frac{\text{BCSS}}{\text{WCSS}} - 1 \qquad (2.29)$$

2.2. Cơ sở thực nghiệm

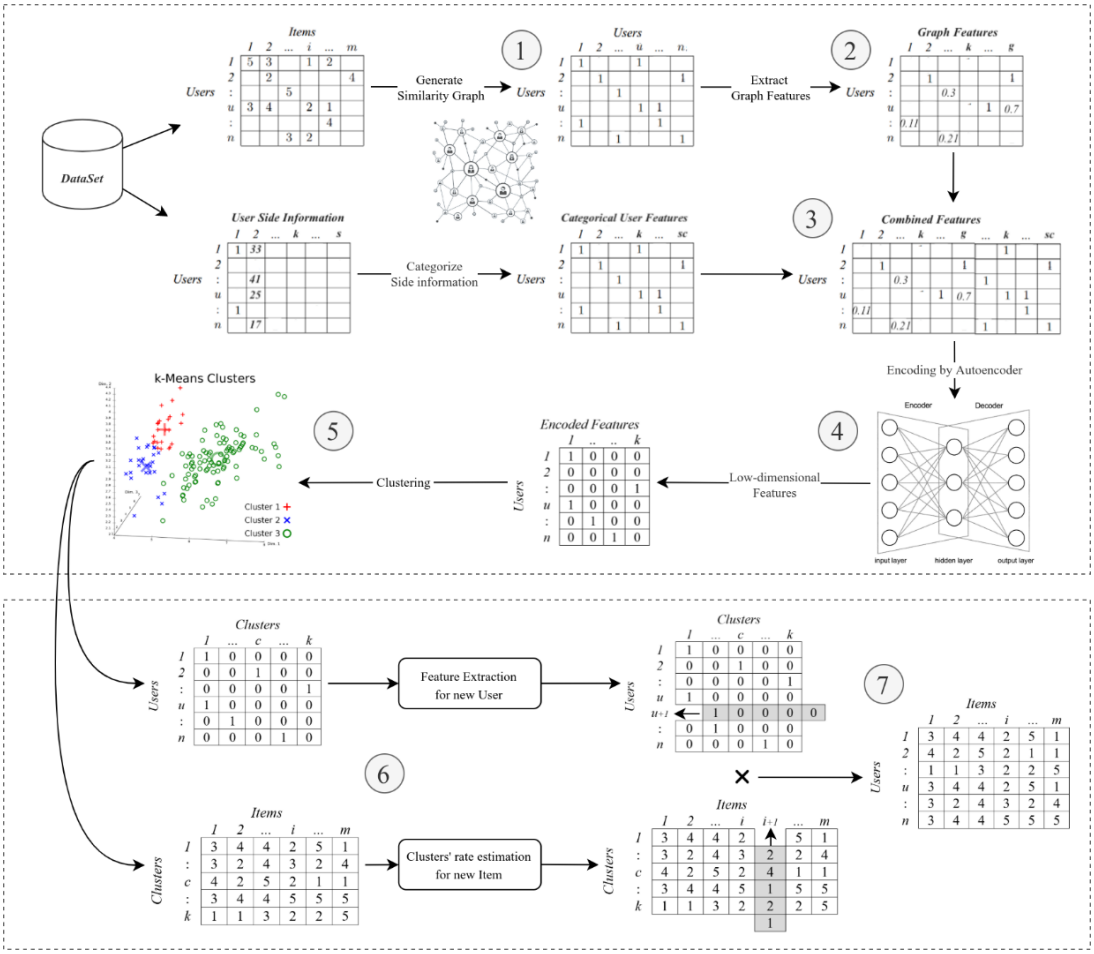
Tập dữ liệu thực nghiệm được sử dụng trong đề án là MovieLens-100k [1] đã được giới thiệu tại mục mở đầu. Bộ dữ liệu này được công bố năm 1998 bởi GroupLens. Nó bao gồm 100.000 đánh giá từ 943 người dùng cho 1682 bộ phim và hơn nữa đã được chia sẵn thành các tập huấn luyện và tập kiểm tra cho việc kiểm định mô hình, chi tiết thông tin các tệp sẽ được sử dụng:

- **u.data**: Chứa toàn bộ các đánh giá của 943 người dùng cho 1682 bộ phim. Mỗi người dùng đánh giá ít nhất 20 bộ phim.
- **ua.base** và **ua.test:** Là tập huấn luyện và tập kiểm tra được chia ra từ tập **u.data** với **ua.test** chứa ít nhất 10 đánh giá cho mỗi người dùng. Việc tập test không tuân theo nguyên tắc phân chia theo tỷ lệ giữa tập dữ liệu và tập huấn luyện sẽ gây ảnh hưởng như thế nào đến sai số bình phương trung bình của các phương pháp được trình bầy tại mục 3.2.
- **u.user**: Chứa thông tin về người dùng, bao gồm: id | age | gender | occupation | zip code. Những thông tin này có thể ảnh hưởng tới sở thích người dùng nhưng trong báo cáo chỉ sử dụng thuộc tính id.
- **u.genre**: Chứa tên của 19 thể loại phim, bao gồm: unknown | Action | Adventure | Animation | Children's | Comedy | Crime | Documentary | Drama | Fantasy | Film-Noir | Horror | Musical | Mystery | Romance | Sci-Fi | Thriller | War | Western.
- **u.item**: Thông tin thuộc tính về mỗi bộ phim.
- **u.info**: Thông tin về bộ dữ liệu

2.3. Xây dựng mô hình GHRS

- **Bước 1**: Xây dựng đồ thị với những người dùng như các nút. Hai người dùng sẽ được kết nối dựa trên những đặc điểm giống nhau theo mục 2.1.1. Cạnh kết nối là những người có sự tương đồng về xếp hạng.
- **Bước 2**: Trích xuất thông tin của người dùng từ đồ thị.

- **Bước 3**: Tiến hành kết hợp thông tin phụ như giới tính và độ tuổi với các đặc trưng dựa trên đồ thị ở bước 2 làm đầu vào cho giai đoạn Autoencoder.
- **Bước 4**: Áp dụng kĩ thuật Autoencoder được mô tả trong mục 1.7.2.4 và 2.1.2 để trích xuất đặc trưng mới và giảm kích thước dữ liệu.
- **Bước 5**: Sử dụng các đặc trưng mới được mã hóa bởi Autoencoder để phân cụm người dùng, sử dụng thuật toán K-means đã trình bày trong mục 2.1.3 để tạo ra một số lượng các nhóm người dùng có sự tương đồng.
- **Bước 6**: Phân bố người dùng mới vào cụm thích hợp dựa trên các đặc trưng được mã hóa và dự đoán xếp hạng các mục mới mà người dùng đó chưa xếp hạng.
- **Bước 7**: Dự đoán xếp hạng của người dùng cho tất cả các mục theo xếp hạng trung bình của cụm và tiến hành đề xuất cho người dùng.



Hình 2.4: Framework của phương pháp GHRS [21]

Kết luận: Chương này đã trình bày một cách cụ thể quá trình xây dựng mô hình GHRS. Sáu kỹ thuật được sử dụng trong quá trình xây dựng đồ thị, phương pháp Autoencoder với nhiễu đầu nhằm tăng cường khả năng học các đặc trưng tiềm ẩn và hai phương pháp phổ biến cho việc tìm hệ số phân cụm tối ưu.

CHƯƠNG III: KẾT QUẢ THỰC NGHIỆM

3.1 Môi trường thực nghiệm

3.1.1 Môi trường thực nghiệm

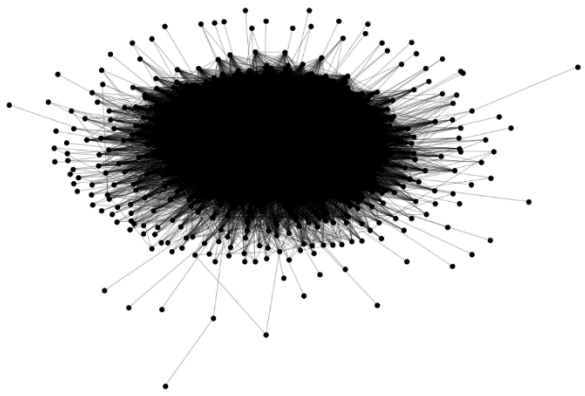
Toàn bộ quá trình cài đặt huấn luyện và kiểm thử được cài đặt trên 2 môi trường: JupyterLab với Intel(R) Core i5-4460 CPU @ 3.20GHz, bộ nhớ RAM 8GB; Google Colab với Intel(R) Xeon(R) CPU @ 2.20GHz, bộ nhớ RAM 12.7 GB.

3.1.2 Ngôn ngữ và thư viện lập trình

Xuyên suốt quá trình thực hiện đề án, tôi đã sử dụng Python 3.7.4 trên JupyterLab cho tất cả các mô hình ngoại trừ mạng Autoencoder và mô hình GHRS được triển khai trên Google Colab sử dụng Python 3.10.12. Các thư viện đi kèm bao gồm: numpy, pandas, matplotlib, sklearn, scipy, pytorch, networkx và một số thư viện hỗ trợ đọc file, tìm đường dẫn.

3.2 Thực hiện các bước xây dựng mô hình GHRS

Bước 1,2: Dựa trên lý thuyết đã trình bày tại mục 2.1.1, cũng như mô tả trong nghiên cứu [21] và hệ số $\alpha = 0.01$ được lựa chọn cho thực nghiệm này. Kết quả đồ thị tương tự (Similarity Graph) thu được như trong hình 3.1 mô tả mối liên hệ giữa 943 người dùng dựa trên đánh giá từ 1682 bộ phim:



Hình 3.1: Đồ thị tương tự của 943 người dùng

Bước 3,4: Dựa trên mô tả trong nghiên cứu [21] về thông tin người dùng kết hợp với kết quả thu được từ bước 2, ta thu được bảng mô tả đặc trưng các user được mô tả trong hình 3.2.

	age1	age2	age3	age4	age5	age6	gender1	...	PR	CD	CC	CB	LC	AND
0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.460577	0.673966	0.817829	0.043582	0.043606	0.479257
1	0.0	0.0	0.0	0.0	0.0	1.0	1.0	...	0.044933	0.014599	0.568223	0.000000	0.000000	0.660178
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4	0.0	0.0	0.0	1.0	0.0	0.0	1.0	...	0.189245	0.255474	0.653926	0.000660	0.000662	0.610428
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
938	0.0	0.0	1.0	0.0	0.0	0.0	1.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
939	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.107218	0.116788	0.618164	0.000025	0.000026	0.705393
940	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
941	0.0	0.0	0.0	0.0	1.0	0.0	1.0	...	0.057353	0.036496	0.574410	0.000000	0.000000	0.709976
942	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.260841	0.369830	0.697905	0.002970	0.002976	0.581300

943 rows × 35 columns

Hình 3.2: Các đặc trưng đầu vào cho Autoencoder

Sau khi chuẩn bị xong dữ liệu, ta xây dựng mô hình Autoencoder như hình 3.3 nhằm giảm chiều dữ liệu để có thể dễ dàng phân cụm người dùng tại bước 5.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                [-1, 16]             576
           Sigmoid-2                [-1, 16]               0
            Linear-3                 [-1, 8]             136
            Linear-4                [-1, 16]             144
           Sigmoid-5                [-1, 16]               0
            Linear-6                [-1, 35]             595
================================================================
Total params: 1,451
Trainable params: 1,451
Non-trainable params: 0
----------------------------------------------------------------
```

Hình 3.3: Thông tin mạng Autoencoder

Bước 5,6,7: Kết thúc bước 4 ta thu được ma trận đặc trưng. Và để áp dụng giải thuật K-means, lúc này cần xác định hệ số k thông qua phương pháp Elbow và Silhouette được trình bày trong mục 2.1.3. Kết quả thu được cho biết số cụm tốt nhất là 3 với số lượng người dùng của từng cụm được mô tả như hình 3.4.

```
Cluster 0: 405 users
Cluster 1: 342 users
Cluster 2: 196 users
```

Hình 3.4: Số người dùng trong mỗi cụm

Hình 3.5 dưới đây mô tả cụ thể user thuộc nhóm nào với 1 đại diện cho user đang được phân vào cụm (cột) tương ứng còn 0 là user đó không thuộc cụm (cột) người dùng đó.

	cluster1	cluster2	cluster3
1	1.0	0.0	0.0
2	1.0	0.0	0.0
3	1.0	0.0	0.0
...	...	...	...
941	0.0	0.0	1.0
942	0.0	1.0	0.0
943	1.0	0.0	0.0

943 rows × 3 columns

Hình 3.5: Ma trận phân cụm người dùng

Thực hiện tương tự cho ma trận phân cụm bộ phim, ta thu được kết quả như hình 3.6:

	1	2	3	4	5	6	...	1681	1682
cluster1	3.960591	3.202899	3.071429	3.555556	3.25	2.875	...	3.0	3.0
cluster2	3.769231	3.136364	2.769231	3.381818	3.411765	3.777778	...	3.262132	3.702399
cluster3	3.729412	3.233333	3.166667	3.727273	3.277778	3.666667	...	3.297181	3.695487

3 rows × 1682 columns

Hình 3.6: Ma trận phân cụm bộ phim

Cuối cùng ta nhân ma trận phân cụm người dùng với ma trận phân cụm bộ phim để có ma trận dự đoán của người dùng với bộ phim được mô tả trong hình 3.7.

	1	2	3	4	5	6	...	1681	1682
1	3.960591	3.202899	3.071429	3.555556	3.25	2.875	...	3.0	3.0
2	3.960591	3.202899	3.071429	3.555556	3.25	2.875	...	3.0	3.0
3	3.960591	3.202899	3.071429	3.555556	3.25	2.875	...	3.0	3.0
4	3.960591	3.202899	3.071429	3.555556	3.25	2.875	...	3.0	3.0
5	3.960591	3.202899	3.071429	3.555556	3.25	2.875	...	3.0	3.0
...	...	...	...	...	...	...	...	...	...
939	3.960591	3.202899	3.071429	3.555556	3.25	2.875	...	3.0	3.0
940	3.729412	3.233333	3.166667	3.727273	3.277778	3.666667	...	3.297181	3.695487
941	3.729412	3.233333	3.166667	3.727273	3.277778	3.666667	...	3.297181	3.695487
942	3.769231	3.136364	2.769231	3.381818	3.411765	3.777778	...	3.262132	3.702399
943	3.960591	3.202899	3.071429	3.555556	3.25	2.875	...	3.0	3.0

943 rows × 1682 columns

Hình 3.7: Ma trận dự đoán người dùng – bộ phim

3.3 Kết quả mô hình và so sánh

Trong đề án này, chúng tôi tiến hành đánh giá hiệu suất của một loạt các phương pháp dự đoán dựa trên cùng một tập huấn luyện **ua.base** và cùng một tập kiểm tra **ua.test**. Các phương pháp đã được trình bày tại chương I và chương II sẽ được đưa vào thực nghiệm bao gồm Content-based, User-based CF, Item-based CF, User-based Matrix Factorization, Item-based MF, Autoencoder-based CF, User-based SVD, Item-based SVD và GHRS.

Bảng 3.1: So sánh độ chính xác giữa các mô hình

Method	RMSE
Content-based	1.2703
User - CF	0.9767
Item - CF	0.9678
User - MF	1.0431
Item - MF	1.0421
Autoencoder - CF	0.9678
User - SVD	1.0031
Item - SVD	1.0050
GHRS	1.0518

Trong số các phương pháp này, User-based CF, Item-based CF và Autoencoder-based CF đã thể hiện sự hiệu quả cao nhất với các giá trị RMSE thấp nhất. Đặc biệt, Autoencoder-based CF cho thấy sự tiến bộ đáng kể trong việc học biểu diễn dữ liệu và dự đoán chính xác hơn về sở thích của người dùng.

Nhận xét: Từ bảng 3.1 ta thấy được mô hình Lọc nội dung có kết quả kém nhất với RMSE ≈ 1.2703, trong khi đó ba mô hình Item - CF, User - CF và Autoencoder - CF cho kết quả khả quan hơn rất nhiều với RMSE đều xấp xỉ 0.97. Mô hình GHRS đã không cho kết quả như kỳ vọng như theo bài báo [21] với RMSE ≈ 1.0518 gần như tương đương với RMSE thu được từ các phương pháp MF. Phương pháp sử dụng kỹ thuật SVD cho kết quả khá tốt với RMSE lần lượt là 1.0031 và 1.0050 . Dưới đây là kết quả đề xuất với $z = 20$ cho người dùng thứ 2 bằng ba mô hình khác nhau:

	recommend
0	Aiqing wansui (1994)
1	Anna (1996)
2	Boys, Les (1997)
3	Close Shave, A (1995)
4	Everest (1998)
5	Great Day in Harlem, A (1994)
6	Little City (1998)
7	Marlene Dietrich: Shadow and Light (1996)
8	Pather Panchali (1955)
9	Prefontaine (1997)
10	Saint of Fort Washington, The (1993)
11	Santa with Muscles (1996)
12	Schindler's List (1993)
13	Some Mother's Son (1996)
14	Someone Else's America (1995)
15	Star Kid (1997)
16	They Made Me a Criminal (1939)
17	To Kill a Mockingbird (1962)
18	Wallace & Gromit: The Best of Aardman Animatio...
19	Wrong Trousers, The (1993)

(a)

	recommend
0	39 Steps, The (1935)
1	Alphaville (1965)
2	Anne Frank Remembered (1995)
3	Close Shave, A (1995)
4	Cry, the Beloved Country (1995)
5	Faust (1994)
6	Grass Harp, The (1995)
7	Guantanamera (1994)
8	Mina Tannenbaum (1994)
9	Nico Icon (1995)
10	Schindler's List (1993)
11	Shawshank Redemption, The (1994)
12	Sling Blade (1996)
13	Someone Else's America (1995)
14	SubUrbia (1997)
15	Thin Man, The (1934)
16	Visitors, The (Visiteurs, Les) (1993)
17	Walking and Talking (1996)
18	Wallace & Gromit: The Best of Aardman Animatio...
19	Winter Guest, The (1997)

(b)

	recommend
0	Aiqing wansui (1994)
1	Anna (1996)
2	Bitter Sugar (Azucar Amargo) (1996)
3	Boys, Les (1997)
4	Brothers in Trouble (1995)
5	Butcher Boy, The (1998)
6	Butcher Boy, The (1998)
7	Faust (1994)
8	Great Day in Harlem, A (1994)
9	Kaspar Hauser (1993)
10	Little City (1998)
11	Pather Panchali (1955)
12	Saint of Fort Washington, The (1993)
13	Santa with Muscles (1996)
14	Schindler's List (1993)
15	Some Mother's Son (1996)
16	Someone Else's America (1995)
17	Spanish Prisoner, The (1997)
18	Two or Three Things I Know About Her (1966)
19	Wrong Trousers, The (1993)

(c)

Hình 3.8: Đề xuất bằng (a) Item – CF, (b) GHRS và (c) Autoencoder – CF

Qua nhận xét về kết quả thu được trong bảng 3.1 phía trên, ta có thể nhận thấy rằng việc thay đổi trong cách xây dựng tập huấn luyện và kiểm tra đã dẫn tới phương pháp được chứng minh có hiệu suất cao như GHRS lại cho ra kết quả kém hơn so với một vài phương pháp phổ biến khác. Việc tác giả bỏ qua thực nghiệm kiểm tra trên tập dữ liệu huấn luyện có phân phối không đồng nhất là một thiếu sót khiến người đọc không nhận thức được rõ ràng phạm vi mà mô hình GHRS có thể đạt được hiệu quả cao và những trường hợp GHRS không thể cho ra kết quả tốt hơn so với các phương pháp phổ biến khác.

KẾT LUẬN CHUNG

I. Kết quả đạt được

Đề án hướng tới một chủ đề quan trọng cả về mặt lý thuyết và thực tiễn trong khoa học máy tính, nó đã và đang được nhiều nhà nghiên cứu quan tâm, đó là nghiên cứu một số phương pháp xây dựng hệ tư vấn.

❖ Về mặt lý thuyết

Đề án đã trình bày tổng quan về bài toán xây dựng hệ tư vấn cũng như vai trò của bài toán trong xã hội hiện nay kèm theo là xu hướng phát triển. Những lý thuyết này về cơ bản là những lý thuyết nền tảng của học máy thống kê và học sâu.

❖ Về mặt thực tiễn

Đề án đã triển khai cài đặt được một số phương pháp nổi bật nhất trong lĩnh vực xây dựng hệ tư vấn, kết quả thực nghiệm của từng phương pháp, từ đó đánh giá được ưu nhược điểm của các phương pháp đó dựa trên tập dữ liệu Movielens-100k.

II. Hạn chế và hướng phát triển

❖ Hạn chế

Do hạn chế về mặt phạm vi và thời gian, kèm theo đó là sức mạnh phần cứng đã cản trở rất nhiều tới hiệu suất thực hiện việc cài đặt các mô hình. Phần cứng yếu đã khiến việc triển khai trên một tập dữ liệu lớn hơn như MovieLens 10M 20M [1] là không khả thi. Các vấn đề khác chưa giải quyết được như khởi động nguội hay việc dữ liệu liên tục cập nhật.

❖ Hướng phát triển

Định hướng tiếp theo của tác giả là tìm hiểu và xây dựng phương pháp mới cho hệ tư vấn trong thời điểm các mô hình ngôn ngữ lớn đã và đang trở nên phổ biến như hiện nay.

TÀI LIỆU THAM KHẢO

[1] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages.

[2] Ricci, F., Rokach, L., Shapira, B. (2022). Recommender Systems: Techniques, Applications, and Challenges. In: Ricci, F., Rokach, L., Shapira, B. (eds) Recommender Systems Handbook. Springer, New York, NY.

[3] C. Li, I. Ishak, H. Ibrahim, M. Zolkepli, F. Sidi and C. Li, "Deep Learning-Based Recommendation System: Systematic Review and Classification," in IEEE Access, vol. 11, pp. 113790-113835, 2023.

[4] Cui, P., Yin, B. & Xu, B. The application of social recommendation algorithm integrating attention model in movie recommendation. Sci Rep 13, 16938 (2023).

[5] S. Feng and T. Zhao, "Hybrid Recommendation System," CAIBDA 2022; 2nd International Conference on Artificial Intelligence, Big Data and Algorithms, Nanjing, China, 2022, pp. 1-5.

[6] Aggarwal, C. C. (2016). Recommender Systems - The Textbook. Springer. ISBN: 978-3-319-29659-3

[7] Daniel Schall. (2015). Social Network-Based Recommender Systems. ISBN: 978-3-319-22735-1, doi: https://doi.org/10.1007/978-3-319-22735-1.

[8] Al-Shamri, M.Y. (2016). User profiling approaches for demographic recommender systems. Knowl. Based Syst., 100, 175-187.

[9] Wooldridge J. M. (2020). Introductory econometrics: a modern approach (Seventh). Cengage Learning. ISBN: 978-1-337-55886-0.

[10] Ricci, Francesco & Shapira, Bracha & Rokach, Lior. (2015). Recommender systems handbook, Second edition. 10.1007/978-1-4899-7637-6.

[11] Vu Huu Tiep. (2018). Machine Learning basic. ISBN: 978-604-67-1095-0.

[12] Jurafsky, D. & Martin, J. (2020). Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition. 3rd Edition. https://web.stanford.edu/~jurafsky/slp3/.

[13] Fethi Fkih, Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 9, 2022, Pages 7645-7669, ISSN 1319-1578.

[14] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl, “Item-based collaborative filtering recommendation algorithms,” in ACM WWW ’01, pp. 285–295, ACM, 2001.

[15] G. Linden, B. Smith, and J. York, “Amazon.com recommendations: Itemto-item collaborative filtering,” IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, 2003.

[16] Funk, Simon. "Netflix Update: Try This at Home". https://sifter.org/~simon/journal/20061211.html.

[17] Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," in Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009.

[18] Ferreira D, Silva S, Abelha A, Machado J. Recommendation System Using Autoencoders. Applied Sciences. 2020; 10(16):5510.

[19] Wang, T., Song, Q., Han, X., Liu, Z., Jin, H., & Hu, X. (2020). AutoRec: An Automated Recommender System. Proceedings of the 14th ACM Conference on Recommender Systems.

[20] Kramer, M.A. (1991). Nonlinear principal component analysis using autoassociative neural networks. Aiche Journal, 37, 233-243.

[21] Zamanzadeh Draban, Zahra & Valipour, M. Hadi. (2022). GHRS: Graph-based hybrid recommendation system with application to movie recommendation. Expert Systems with Applications. 200.

[22] Ricci, F., Rokach, L., Shapira, B. (2011). Introduction to Recommender Systems Handbook. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds) Recommender Systems Handbook. Springer, Boston, MA.

[23] Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. J Big Data 9, 59 (2022).