

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM



BÁO CÁO

ĐỒ ÁN 1

ĐỀ TÀI

TÌM HIỂU RECOMMENDER SYSTEM VÀ XÂY DỰNG
ỨNG DỤNG MINH HOẠ

Giảng viên hướng dẫn: ThS. Huỳnh Hồ Thị Mộng Trinh

Lớp: SE121.P11

Sinh viên thực hiện

Phạm Hoàng Duy 22520339

Hà Phú Thịnh 22521405

TP. Hồ Chí Minh, tháng 12 năm 2024

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

....., ngày tháng năm 2024

Người nhận xét
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Đầu tiên, nhóm xin gửi lời cảm ơn chân thành đến quý Thầy Cô Trường Đại học Công nghệ thông tin - Đại học quốc gia TP.HCM và quý Thầy Cô khoa Công nghệ phần mềm đã giúp cho nhóm có kiến thức cơ bản làm nền tảng để thực hiện đề tài này.

Đặc biệt, nhóm xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất tới Cô Huỳnh Hồ Thị Mộng Trinh (Giảng viên hướng dẫn Đồ án 1). Cô đã trực tiếp hướng dẫn tận tình, sửa chữa và đóng góp nhiều ý kiến quý báu giúp nhóm hoàn thành tốt báo cáo môn học của mình.

Trong thời gian một học kỳ thực hiện đề tài, nhóm tác giả đã vận dụng những kiến thức nền tảng đã tích lũy đồng thời kết hợp với việc học hỏi và nghiên cứu những kiến thức mới. Từ đó, nhóm vận dụng tối đa những gì đã tiếp thu được để hoàn thành một báo cáo đồ án tốt nhất. Tuy nhiên, trong quá trình thực hiện, nhóm không tránh khỏi những thiếu sót. Chính vì vậy, nhóm rất mong được sự góp ý từ phía Thầy cô nhằm hoàn thiện những kiến thức mà nhóm đã học tập và là hành trang để nhóm thực hiện tiếp các đề tài trong tương lai.

Xin chân thành cảm ơn quý Thầy cô!

Nhóm sinh viên thực hiện

MỤC LỤC

1. GIỚI THIỆU.....	8
1.1. Lý do chọn đề tài	8
1.2. Mục tiêu nghiên cứu.....	8
1.3. Đối tượng và phạm vi nghiên cứu.....	9
1.3.1. Đối tượng nghiên cứu	9
1.3.2. Phạm vi nghiên cứu	10
2. CƠ SỞ LÝ THUYẾT	11
2.1. Giới thiệu về recommender system	11
2.1.1. Tổng quan	11
2.1.2. Phân loại recommender system	13
2.1.3. Phân loại rating	14
2.2. Các phương pháp đánh giá recommender system.....	16
2.3. Content based filtering	21
2.3.1. Giới thiệu	21
2.3.2. Ý tưởng.....	22
2.3.3. Các thuật toán và phương pháp	22
2.3.4. Ưu điểm và nhược điểm	26
2.4. Collaborative filtering.....	26

2.5. Neighborhood based collaborative filtering	29
2.5.1. Giới thiệu	29
2.5.2. Ý tưởng.....	29
2.5.3. Thuật toán.....	30
2.5.4. Ưu điểm và nhược điểm	39
2.5.5. Case study	40
2.6. Matrix factorization method	40
2.6.1. Giới thiệu	40
2.6.2. Ý tưởng.....	41
2.6.3. Thuật toán.....	43
2.6.4. SVD (Singular Value Decomposition).....	47
2.6.5. Ưu và nhược điểm	49
2.6.6. Case study	50
2.7. Deep learning method	51
2.7.1. Giới thiệu	51
2.7.2. Giới thiệu	52
2.7.3. Autoencoder based recommender system	55
2.7.4. RNN for Clickstream recommender system	57
2.7.5. GRU4Rec	60
2.7.6. Ưu, nhược điểm của Deep Learning	61
2.7.7. Case Study	62
2.8. Hybrid method.....	62

2.8.1. Giới thiệu	62
2.8.2. Case Study	63
2.9. Một số thư viện xây dựng Recommender System.....	64
2.10.Scaling recommender system	68
2.10.1. Giới thiệu về Apache Spark.....	69
2.10.2. Kiến trúc của Spark	71
2.10.3. Spark SQL	72
2.10.4. Spark Streaming.....	72
2.10.5. GraphX.....	73
2.11.Case Study.....	74
2.11.1. Amazon.....	74
2.11.2. Netflix	75
2.12.Các vấn đề khi xây dựng recommender system.....	77
2.12.1. Data Sparsity	77
2.12.2. Cold-Start Problem	78
2.12.3. Privacy	79
2.13.Các phương pháp khuyến nghị hay nhưng chưa phổ biến	
80	
2.13.1. DeepFM	80
2.13.2. NCF.....	81
3. NGHIÊN CỨU THỰC NGHIỆM	82

3.1. Thực nghiệm các phương pháp khuyến nghị.....	82
3.1.1. Thực nghiệm Content based filtering	90
3.1.2. Thực nghiệm Neighborhood based collaborative filtering	95
3.1.3. Thực nghiệm Matrix factorization method	99
3.1.4. Thực nghiệm Deep learning method	108
3.1.5. Thực nghiệm ALS với Spark	126
3.2. Xây dựng ứng dụng minh họa	137
3.2.1. Ứng dụng mua bán khóa học online	137
3.2.2. Ứng dụng blog cá nhân.....	146
3.2.3. Ứng dụng thương mại điện tử.....	154
4. KẾT LUẬN	159
4.1. Kết quả đạt được	159
4.2. Hạn chế	159
4.3. Hướng phát triển	159
5. TÀI LIỆU THAM KHẢO.....	160

1. GIỚI THIỆU

1.1. Lý do chọn đề tài

Trong bối cảnh xu hướng chuyển đổi số và khai thác dữ liệu ngày càng tăng, các hệ thống gợi ý (recommender system) đóng vai trò quan trọng trong nhiều lĩnh vực như thương mại điện tử, giải trí, giáo dục, và y tế, ... Thế nên việc cung cấp những gợi ý phù hợp không chỉ giúp tăng tỷ lệ hài lòng của người dùng, mà còn nâng cao hiệu quả kinh doanh và gia tăng lợi nhuận cho các doanh nghiệp.

Hướng nghiên cứu này đáng chú ý ở chỗ khả năng áp dụng rộng rãi và tính thực tiễn cao. Vì thế nên trong Đồ Án 1, nhóm quyết định lựa chọn đề tài "Tìm hiểu về Recommender System và xây dựng ứng dụng minh họa" với mong muốn:

- Hiểu rõ nguyên lý vận hành của các hệ thống gợi ý, bao gồm các phương pháp phổ biến như *Collaborative filtering*, *Content-based filtering* và *Hybrid methods*.
- Đánh giá được tác động thực tế của hệ thống gợi ý trong các ngành nghề khác nhau, từ đó rút ra bài học và đề xuất cải tiến.
- Tìm hiểu và áp dụng các công cụ, thuật toán tiên tiến trong lĩnh vực khoa học dữ liệu và machine learning để phát triển một hệ thống gợi ý.

Với những lý do nêu trên, đề tài này không chỉ mang lại kiến thức chuyên môn đáng giá mà còn là bước đệm quan trọng trong việc định hướng nghiên cứu và áp dụng công nghệ vào thực tiễn sau này.

1.2. Mục tiêu nghiên cứu

Ở đề tài này, nhóm đã đặt ra các mục tiêu nghiên cứu như sau:

- Khảo sát các phương pháp và thuật toán chính được sử dụng trong các hệ thống gợi ý, từ đó xây dựng nền tảng kiến thức vững chắc cho việc phát triển mô hình.

- Đánh giá hiệu quả và vai trò của các hệ thống gợi ý trong các lĩnh vực thực tiễn như thương mại điện tử, giáo dục,
- Thiết kế và triển khai một hệ thống gợi ý đơn giản để minh họa cách thức hoạt động của các thuật toán đã nghiên cứu, đồng thời đánh giá hiệu suất của hệ thống dựa trên các chỉ số như độ chính xác và mức độ hài lòng của người dùng.
- Đưa ra các ý tưởng cải tiến hoặc ứng dụng mới dựa trên kết quả nghiên cứu và thử nghiệm, nhằm tối ưu hóa hiệu quả của hệ thống gợi ý trong các tình huống cụ thể.

1.3. Đối tượng và phạm vi nghiên cứu

1.3.1. Đối tượng nghiên cứu

- Nghiên cứu các thuật toán chính như collaborative filtering (dựa trên sự hợp tác), content-based filtering (dựa trên nội dung), và hybrid methods (phương pháp lai) nhằm tìm hiểu cách mỗi phương pháp vận hành, ưu và nhược điểm trong từng trường hợp ứng dụng cụ thể.
- Tìm hiểu về các hệ thống thực tế đang sử dụng recommender systems, bao gồm các nền tảng thương mại điện tử (như Amazon, Shopee), dịch vụ giải trí trực tuyến (như Netflix, Spotify), và các nền tảng giáo dục trực tuyến (như Coursera, Udemy). Từ đó phân tích cơ chế gợi ý mà các nền tảng này sử dụng.
- Nghiên cứu các nguồn dữ liệu cần thiết để phát triển hệ thống gợi ý, bao gồm:
 - Hồ sơ người dùng, lịch sử tương tác, sở thích và hành vi.
 - Thông tin về sản phẩm hoặc dịch vụ, các đặc điểm chi tiết của từng mục gợi ý.
 - Các đánh giá, nhận xét hoặc điểm số mà người dùng đưa ra, giúp làm cơ sở để cải thiện chất lượng gợi ý.

- Phân tích nhóm người dùng mục tiêu của các hệ thống gợi ý, bao gồm thói quen sử dụng, mức độ hài lòng và kỳ vọng của người dùng đối với các gợi ý được đưa ra.
- Đánh giá tác động của các hệ thống gợi ý đối với doanh nghiệp (tăng doanh thu, tối ưu hóa quảng cáo) và đối với người dùng (tăng trải nghiệm cá nhân hóa, tiết kiệm thời gian). Nghiên cứu các thách thức liên quan đến tính công bằng, quyền riêng tư và bảo mật trong hệ thống gợi ý.

1.3.2. Phạm vi nghiên cứu

- **Phạm vi lý thuyết**
 - Tập trung vào các phương pháp phổ biến nhất trong hệ thống gợi ý như collaborative filtering, content-based filtering và hybrid methods.
 - Khảo sát các thuật toán cơ bản như matrix factorization, k-nearest neighbors (k-NN), và deep learning trong recommender systems.
 - Phân tích các ưu, nhược điểm của từng phương pháp để đưa ra sự so sánh và lựa chọn phù hợp cho từng ngữ cảnh ứng dụng.
- **Phạm vi thực tiễn**
 - Nghiên cứu các hệ thống gợi ý thực tế được áp dụng trên các nền tảng phổ biến như Amazon, Netflix, Spotify, Shopee, và các nền tảng học tập trực tuyến.
 - Đánh giá hiệu quả của các hệ thống gợi ý qua các chỉ số như MAE (Mean Average Error), RMSE (Root Mean Square Error), Hit Rate, ...
 - Phân tích các bài toán thách thức như cold start (khởi đầu lạnh), scalability (khả năng mở rộng), và xử lý dữ liệu lớn.
- **Phạm vi dữ liệu**
 - Tập trung vào dữ liệu người dùng, bao gồm lịch sử mua sắm, hành vi tương tác, và các đánh giá/điểm số từ người dùng.

- Sử dụng các bộ dữ liệu công khai như MovieLens, Amazon Reviews để thực nghiệm và kiểm chứng các thuật toán.
- Xử lý dữ liệu bằng các công cụ và thư viện phổ biến như Pandas, NumPy, Scikit-learn, ...

- **Giới hạn nghiên cứu**

- Không tập trung sâu vào các thuật toán tiên tiến và phức tạp nhất trong lĩnh vực recommender systems do giới hạn về thời gian và nguồn lực.
- Chỉ triển khai hệ thống mẫu với quy mô nhỏ, phục vụ mục đích minh họa thay vì một sản phẩm hoàn chỉnh.

2. CƠ SỞ LÝ THUYẾT

2.1. Giới thiệu về recommender system

2.1.1. Tổng quan

- Recommender System, hay còn được gọi là hệ thống đề xuất/ hệ thống khuyến nghị, là một phần mềm hoặc hệ thống thông tin được thiết kế để tự động đưa ra gợi ý cho người dùng về các mục hoặc thông tin mà họ có thể quan tâm. Sự đề xuất này dựa trên việc phân tích và đánh giá dữ liệu lịch sử của người dùng, như lịch sử mua sắm, đánh giá, hay hành vi trực tuyến. Recommender System sử dụng các thuật toán và mô hình học máy để tối ưu hóa quá trình đề xuất, tạo ra trải nghiệm cá nhân hóa cho từng người dùng.
- Nó ra đời nhằm giải quyết bài toán cơ bản trong việc kết nối người dùng với những sản phẩm, dịch vụ hoặc thông tin phù hợp, từ đó tối ưu hóa lợi ích cho cả người dùng và doanh nghiệp. Mặc dù mục tiêu tổng thể thường xoay quanh việc tăng doanh thu và sự hài lòng của người dùng, hệ thống gợi ý còn có nhiều mục đích cụ thể, được chia thành các khía cạnh sau:

- **Tăng tính liên quan của gợi ý:** Mục tiêu cốt lõi của mọi hệ thống gợi ý là đưa ra các đề xuất phù hợp và liên quan đến sở thích hoặc nhu cầu của người dùng. Người dùng thường có xu hướng tiêu thụ các sản phẩm hoặc dịch vụ mà họ cảm thấy hấp dẫn, vì vậy tính liên quan là yếu tố hàng đầu để tăng tỷ lệ chấp nhận các gợi ý.
- **Cung cấp sự mới mẻ:** Hệ thống gợi ý mang lại giá trị thực sự khi đề xuất những mục mà người dùng chưa từng biết đến hoặc trải nghiệm trước đó. Ví dụ, việc giới thiệu một bộ phim hoặc sản phẩm mới trong thể loại mà người dùng yêu thích có thể khuyến khích họ khám phá thêm và nâng cao trải nghiệm cá nhân hóa.
- **Tạo yếu tố bất ngờ:** Tạo yếu tố bất ngờ hướng tới việc gợi ý những mục mà người dùng không ngờ tới nhưng lại phù hợp với sở thích tiềm ẩn của họ. Điều này không chỉ tăng trải nghiệm thú vị mà còn mở ra cơ hội khám phá những lĩnh vực hoặc sản phẩm mới mà người dùng chưa từng nghĩ tới trước đó. Ví dụ, gợi ý một nhà hàng với món ăn độc đáo có thể khiến người dùng thay đổi thói quen tiêu dùng và mở rộng sở thích của mình.
- **Đa dạng hóa gợi ý:** Một danh sách gợi ý phong phú và đa dạng giúp tăng khả năng người dùng tìm thấy ít nhất một mục phù hợp. Điều này cũng tránh sự nhảm chán do việc lặp lại các đề xuất tương tự. Đa dạng hóa gợi ý có ý nghĩa quan trọng trong việc cải thiện trải nghiệm tổng thể và tăng cơ hội người dùng thực hiện hành động tiêu dùng.
- **Tăng sự hài lòng và lòng trung thành của người dùng:** Việc cung cấp gợi ý phù hợp không chỉ giúp người dùng nhanh chóng tìm thấy sản phẩm hoặc dịch vụ mình cần, mà còn nâng cao sự hài lòng tổng thể đối với nền tảng. Người dùng hài lòng thường có xu hướng quay lại sử dụng dịch vụ và trở thành khách hàng trung thành, từ đó tạo ra giá trị lâu dài cho doanh nghiệp.
- **Cá nhân hóa trải nghiệm người dùng:** Bằng cách thu thập và phân tích dữ liệu từ hành vi, sở thích và tương tác của người dùng, hệ thống gợi ý

có thể cung cấp các trải nghiệm được tùy chỉnh cao, giúp mỗi người dùng cảm thấy rằng dịch vụ được thiết kế riêng cho họ.

- **Hỗ trợ doanh nghiệp tối ưu hóa chiến lược:** Từ góc độ doanh nghiệp, hệ thống gợi ý không chỉ giúp tăng doanh thu mà còn cung cấp các thông tin chi tiết về hành vi và nhu cầu của người dùng. Các dữ liệu này giúp doanh nghiệp xây dựng chiến lược quảng cáo, cải thiện sản phẩm, và nâng cao trải nghiệm khách hàng.
- **Cải thiện tương tác và sử dụng nền tảng:** Trong một số trường hợp, mục đích của hệ thống gợi ý không chỉ giới hạn ở việc tăng doanh thu mà còn nhằm thúc đẩy tương tác của người dùng với nền tảng. Ví dụ, Facebook sử dụng hệ thống gợi ý để đề xuất bạn bè mới, từ đó tăng tính kết nối và giữ chân người dùng trên mạng xã hội, qua đó tối ưu hóa lợi nhuận từ quảng cáo.
- **Giải thích và xây dựng lòng tin:** Một mục tiêu quan trọng khác của hệ thống gợi ý là cung cấp lời giải thích minh bạch về lý do một mục cụ thể được đề xuất. Điều này không chỉ tăng sự tin tưởng của người dùng mà còn cải thiện mối quan hệ giữa họ và nền tảng.

2.1.2. Phân loại recommender system

Hệ thống gợi ý thường được phân loại dựa trên cách thức sử dụng dữ liệu và phương pháp tiếp cận. Dưới đây là các loại cơ bản của hệ thống gợi ý:

- **Collaborative Filtering (Gợi ý dựa trên sự hợp tác):** Collaborative filtering sử dụng dữ liệu tương tác giữa người dùng và sản phẩm, chẳng hạn như đánh giá hoặc hành vi mua sắm, để đưa ra gợi ý. Phương pháp này dựa trên ý tưởng rằng những người dùng có sở thích giống nhau trong quá khứ có khả năng tiếp tục thích những sản phẩm giống nhau trong tương lai. Collaborative filtering được chia thành hai loại:
 - **User-based Collaborative Filtering:** Gợi ý dựa trên sự tương đồng giữa các người dùng.

- **Item-based Collaborative Filtering:** Gợi ý dựa trên sự tương đồng giữa các sản phẩm.
- **Content-Based Filtering (Gợi ý dựa trên nội dung):** Phương pháp này dựa trên các đặc điểm của sản phẩm và lịch sử của người dùng để xây dựng mô hình dự đoán. Ví dụ, nếu một người dùng đánh giá cao một bộ phim thuộc thể loại khoa học viễn tưởng, hệ thống có thể gợi ý các bộ phim tương tự dựa trên mô tả nội dung của chúng. Phương pháp này đặc biệt hữu ích trong trường hợp sản phẩm mới chưa có nhiều đánh giá.
- **Knowledge-Based Systems (Hệ thống dựa trên tri thức):** Hệ thống này không phụ thuộc vào dữ liệu lịch sử mà sử dụng các yêu cầu được định nghĩa rõ ràng của người dùng, kết hợp với cơ sở tri thức bên ngoài để đưa ra gợi ý. Chúng thường được sử dụng trong các lĩnh vực như bất động sản, du lịch, hoặc sản phẩm xa xỉ, nơi dữ liệu lịch sử hạn chế và các thuộc tính sản phẩm rất đa dạng.
- **Demographic Recommender Systems (Hệ thống gợi ý dựa trên thông tin nhân khẩu học):** Loại hệ thống này sử dụng thông tin nhân khẩu học của người dùng, chẳng hạn như độ tuổi, giới tính, hoặc nghề nghiệp, để xây dựng mô hình dự đoán. Mặc dù không mạnh mẽ khi sử dụng độc lập, nhưng chúng thường được tích hợp vào các hệ thống lai để tăng cường hiệu quả.
- **Hybrid Systems (Hệ thống lai):** Đây là sự kết hợp của nhiều loại hệ thống gợi ý để tận dụng lợi thế của từng phương pháp. Ví dụ, kết hợp Collaborative Filtering và Content-Based Filtering có thể giúp cải thiện độ chính xác và giảm thiểu các hạn chế của từng phương pháp riêng lẻ.

2.1.3. Phân loại rating

Dữ liệu đánh giá trong hệ thống gợi ý có thể được chia thành hai nhóm chính dựa trên cách thu thập thông tin từ người dùng:

2.1.3.1. Explicit Ratings (Đánh giá rõ ràng)

Đây là loại đánh giá mà người dùng cung cấp một cách trực tiếp, thường thông qua các biểu mẫu, bảng đánh giá, hoặc hệ thống sao. Người dùng tự ý chọn mức độ yêu thích hoặc không thích cho sản phẩm/dịch vụ. Các loại đánh giá thuộc nhóm này bao gồm:

- **Continuous Ratings** (Đánh giá liên tục): Giá trị đánh giá nằm trên một thang đo liên tục, ví dụ từ -10 đến 10. Loại này ít phổ biến vì yêu cầu người dùng phải chọn chính xác một giá trị trên một dải rộng.

Ví dụ: Một hệ thống yêu cầu người dùng đánh giá từ -10 đến 10 cho các truyện cười (Jester Joke Dataset).

- **Interval-Based Ratings** (Đánh giá theo khoảng): Giá trị đánh giá được chọn từ một thang đo rời rạc, chẳng hạn như 1 đến 5 sao hoặc 1 đến 10 điểm. Đây là dạng phổ biến nhất trong các hệ thống gợi ý.

Ví dụ: Hệ thống 5 sao của Netflix hoặc Amazon.

- **Ordinal Ratings** (Đánh giá theo thứ tự): Sử dụng các giá trị phân loại có thứ tự, ví dụ: "Rất không thích", "Không thích", "Trung lập", "Thích", "Rất thích". Loại này có ý nghĩa xếp hạng nhưng không giả định khoảng cách đều nhau giữa các giá trị.

Ví dụ: Khảo sát đánh giá mức độ hài lòng.

2.1.3.2. Implicit Ratings (Đánh giá suy diễn)

Đây là loại đánh giá được suy ra từ hành vi của người dùng mà không yêu cầu họ cung cấp đánh giá trực tiếp. Các loại đánh giá thuộc nhóm này bao gồm:

- **Binary Ratings** (Đánh giá nhị phân): Biểu thị hai trạng thái, thường là "Thích" hoặc "Không thích". Đây là dạng đặc biệt đơn giản và phổ biến.
- Ví dụ:** Hệ thống của Pandora Radio cho phép người dùng nhấn "Thích" hoặc "Không thích" một bài hát.
- **Unary Ratings** (Đánh giá đơn trị): Biểu thị sự yêu thích mà không cung cấp cơ chế để thể hiện sự không thích. Thông tin đánh giá được suy diễn từ hành vi như xem, mua hàng, hoặc tải xuống.

Ví dụ: Nút "Like" trên Facebook.

2.2. Các phương pháp đánh giá recommender system

Để đảm bảo hiệu quả và tính phù hợp của các hệ thống gợi ý, việc đánh giá chúng trở thành một bước không thể thiếu trong quy trình phát triển. Quá trình đánh giá giúp xác định khả năng đáp ứng các mục tiêu chính của hệ thống, bao gồm độ chính xác, tính cá nhân hóa, và hiệu quả thương mại. Phần này trình bày các phương pháp đánh giá hệ thống gợi ý, được chia thành ba nhóm chính: **User Studies**, **Online Evaluation**, và **Offline Evaluation**.

2.2.1. User Studies (Nghiên cứu người dùng)

Phương pháp này tập trung vào việc thu thập phản hồi trực tiếp từ người dùng thông qua các bài kiểm tra thực nghiệm. Người dùng được yêu cầu thực hiện các nhiệm vụ cụ thể trên hệ thống và cung cấp đánh giá dựa trên trải nghiệm của họ.

Đặc điểm chính:

- Người dùng đưa ra phản hồi thông qua bảng khảo sát, phỏng vấn, hoặc các phương pháp định lượng khác.
- Dữ liệu thu thập bao gồm: mức độ hài lòng, tính dễ sử dụng của giao diện, và chất lượng gợi ý.

Ưu điểm:

- Cho phép đánh giá các khía cạnh chủ quan như trải nghiệm người dùng và mức độ hài lòng.
- Phù hợp để kiểm tra các cải tiến về giao diện hoặc thuật toán.

Nhược điểm:

- Quá trình tuyển chọn người dùng có thể dẫn đến thiên lệch dữ liệu.
- Chi phí cao và khó thực hiện trên quy mô lớn.

Ví dụ thực tế: Một nền tảng như Coursera có thể mời một nhóm người dùng thử nghiệm các gợi ý khóa học mới và ghi nhận phản hồi về mức độ phù hợp và hữu ích.

2.2.2. Online Evaluation (Đánh giá trực tuyến)

Phương pháp này được thực hiện trên hệ thống thực tế, nơi người dùng thực hiện các hoạt động tương tác trong môi trường thực. Một phương pháp phổ biến trong đánh giá trực tuyến là **A/B testing**, trong đó hai phiên bản hệ thống được triển khai song song để so sánh hiệu quả.

Đặc điểm chính:

- Đánh giá dựa trên các hành động thực tế của người dùng, chẳng hạn như tỷ lệ nhấp chuột (CTR) hoặc tỷ lệ chuyển đổi (conversion rate).
- Không yêu cầu người dùng phải cung cấp đánh giá chủ quan.

Ưu điểm:

- Đo lường hiệu quả thực tế trong môi trường sử dụng thật.

- Cung cấp dữ liệu phong phú và liên quan đến kết quả kinh doanh.

Nhược điểm:

- Yêu cầu cơ sở hạ tầng lớn và phức tạp.
- Có thể gặp khó khăn trong việc triển khai đồng thời nhiều phiên bản hệ thống.

Ví dụ thực tế: Netflix sử dụng A/B testing để kiểm tra các thuật toán gợi ý mới nhằm xác định liệu chúng có làm tăng tỷ lệ người xem hay không.

2.2.3. Offline Evaluation (Đánh giá ngoại tuyến)

Đây là phương pháp phổ biến nhất trong các nghiên cứu học thuật, sử dụng dữ liệu lịch sử để đánh giá hiệu quả của hệ thống mà không cần sự tham gia của người dùng thực tế.

Đặc điểm chính:

- Dữ liệu lịch sử được chia thành tập huấn luyện và tập kiểm tra.
- Hệ thống được xây dựng trên tập huấn luyện và kiểm tra hiệu suất trên tập kiểm tra.

Ưu điểm:

- Đễ dàng thực hiện trên nhiều bộ dữ liệu chuẩn như MovieLens hoặc Amazon Reviews.
- Không yêu cầu người dùng thực tế, giảm chi phí và thời gian.

Nhược điểm:

- Không thể đo lường phản ứng của người dùng với các gợi ý mới.
- Không phản ánh được các yếu tố như bất ngờ (**Serendipity**) hoặc mới mẻ (**Novelty**).

Ví dụ thực tế: Sử dụng tập dữ liệu MovieLens để kiểm tra khả năng dự đoán xếp hạng của người dùng đối với các bộ phim chưa từng xem.

2.2.4. Các chỉ số đánh giá hệ thống gợi ý

- **Độ chính xác (Accuracy):**

Độ chính xác là chỉ số phổ biến nhất, tập trung vào việc đo lường khả năng hệ thống gợi ý đúng những mục mà người dùng yêu thích hoặc đã tương tác.

- **Mean Absolute Error (MAE):** MAE đo lường mức độ sai số trung bình tuyệt đối giữa giá trị dự đoán và giá trị thực tế. MAE càng thấp, dự đoán càng chính xác.

Công thức:

$$\text{MAE} = \frac{\sum_{i=1}^n |y_i - x_i|}{n}$$

với:

y_i là giá trị dự đoán

x_i là giá trị thực tế

n là tổng số đánh giá

- **Root Mean Squared Error (RMSE):** RMSE phóng đại ảnh hưởng của các sai số lớn hơn, nhờ đó nhấn mạnh vào các trường hợp hệ thống đưa ra dự đoán sai nghiêm trọng.

Công thức:

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

với:

\hat{y}_i là giá trị dự đoán

y_i là giá trị thực tế

n là tổng số đánh giá

- **Độ bao phủ (Coverage):**

Độ bao phủ đo lường tỷ lệ các mục hoặc người dùng được phục vụ bởi hệ thống gợi ý. Chỉ số này giúp đánh giá mức độ hệ thống có thể đáp ứng các sở thích đa dạng hay không.

- **User Coverage:** Tỷ lệ người dùng nhận được ít nhất một gợi ý phù hợp.

Công thức:

$$\text{User Coverage} = \frac{\text{Số người dùng có gợi ý phù hợp}}{\text{Tổng số người dùng}}$$

- **Item Coverage:** Tỷ lệ mục được đưa vào danh sách gợi ý so với tổng số mục trong hệ thống.

$$\text{Item Coverage} = \frac{\text{Số mục được gợi ý}}{\text{Tổng số mục trong hệ thống}}$$

- **Tính bất ngờ (Serendipity):**

Tính bất ngờ đo lường khả năng hệ thống cung cấp các gợi ý mà người dùng không ngờ tới nhưng vẫn phù hợp với sở thích tiềm ẩn của họ. Chỉ số này giúp thúc đẩy sự khám phá và mở rộng sở thích người dùng.

Ví dụ: Một người thường xuyên nghe nhạc pop được gợi ý một bài hát jazz mà họ thực sự yêu thích.

- **Tính mới mẻ (Novelty):**

Tính mới mẻ đo lường khả năng hệ thống gợi ý những mục mà người dùng chưa từng biết đến. Điều này giúp tăng trải nghiệm và sự hấp dẫn của hệ thống.

Ví dụ: Gợi ý một bộ phim mới ra mắt mà người dùng chưa từng nghe nói đến, thay vì gợi ý các bộ phim phổ biến mà họ đã biết.

- **Tính đa dạng (Diversity)**

Tính đa dạng đo lường mức độ khác biệt giữa các mục trong danh sách gợi ý. Một danh sách gợi ý đa dạng giúp tăng khả năng người dùng tìm thấy ít nhất một mục họ yêu thích.

Ví dụ: Gợi ý danh sách bao gồm phim hành động, phim hài và phim khoa học viễn tưởng sẽ đa dạng hơn danh sách chỉ gồm phim hành động.

2.3. Content based filtering

2.3.1. Giới thiệu

Content-Based Filtering (Lọc dựa trên nội dung) là một trong những phương pháp phổ biến trong các hệ thống khuyến nghị, nơi các đề xuất cho người dùng được tạo ra dựa trên sự tương đồng giữa các mục được người dùng yêu thích trong quá khứ và các đặc tính của các mục mới. Mục tiêu của content-based filtering là giới thiệu các sản phẩm, dịch vụ hoặc thông tin mà người dùng có thể quan tâm, dựa trên các đặc điểm cụ thể của những mục mà họ đã tương tác hoặc đánh giá cao trước đó.

Phương pháp này dựa trên hai nguồn dữ liệu chính:

- **Nội dung sản phẩm:** Đây là thông tin mô tả các mục cần đề xuất, có thể là văn bản mô tả sản phẩm từ nhà sản xuất, hoặc các thuộc tính khác như thể loại, tác giả, đạo diễn, v.v. **Ví dụ:** một bộ phim có thể được mô tả qua các thể loại như "Science Fiction", "Action", hoặc "Horror".

- **Hồ sơ người dùng (User Profiles):** Hồ sơ này được xây dựng từ phản hồi của người dùng về các sản phẩm. Phản hồi có thể là phản hồi rõ ràng (explicit feedback) như đánh giá của người dùng, hoặc phản hồi suy diễn (implicit feedback) như các hành động của người dùng (**Ví dụ:** thời gian xem, tần suất tương tác với sản phẩm, ...). Hồ sơ người dùng liên kết các thuộc tính của sản phẩm với sở thích của người dùng, từ đó tạo ra các đề xuất dựa trên sở thích đã được ghi nhận.

2.3.2. Ý tưởng

- Content-based filtering dựa vào thông tin mô tả về các mục (items) trong hệ thống, chẳng hạn như thể loại, từ khóa, tác giả, và các đặc tính khác của sản phẩm hoặc dịch vụ. Để làm được điều này, hệ thống sẽ phân tích và so sánh các đặc tính này với những sản phẩm hoặc dịch vụ mà người dùng đã thích hoặc đã đánh giá trong quá khứ.
Ví dụ: Nếu một người dùng đã thể hiện sự yêu thích đối với một số bộ phim khoa học viễn tưởng, hệ thống sẽ đề xuất những bộ phim khác cùng thể loại hoặc có đặc điểm tương tự.
- Phương pháp này không dựa vào hành vi hoặc đánh giá của người dùng khác trong hệ thống, mà chủ yếu tập trung vào sở thích và tương tác của chính người dùng đó. Điều này có nghĩa là, mặc dù mô hình này có thể không có được các đề xuất mang tính đa dạng cao như phương pháp collaborative filtering (lọc cộng tác), nhưng nó lại có thể đưa ra những đề xuất rất chính xác và cá nhân hóa.

2.3.3. Các thuật toán và phương pháp

Đối với hệ thống Content-based, việc xây dựng thuật toán gồm 2 bước chính:

- **Bước 1:** Trích xuất đặc trưng của items và biểu diễn chúng dưới dạng các vector đặc trưng. Bước này có thể gọi là xây dựng **Item Profile**.
- **Bước 2:** Tiến hành áp dụng các thuật toán để dự đoán hoặc gợi ý.

2.3.3.1. Xây dựng Item Profile:

Trong các hệ thống content-based, chúng ta cần xây dựng một bộ hồ sơ (Profile) cho mỗi item. Profile này được biểu diễn dưới dạng toán học là một vector đặc trưng (**Feature Vector**) n chiều. Trong những trường hợp đơn giản (ví dụ như item là dữ liệu dạng văn bản hoặc dữ liệu có cấu trúc đơn giản), vector đặc trưng được trực tiếp trích xuất từ item. Từ đó chúng ta có thể xác định các item có nội dung tương tự bằng cách tính độ tương đồng giữa các feature vector của chúng. Một số phương pháp thường được sử dụng để xây dựng vector đặc trưng là:

- **TF-IDF (Term frequency - Invert Document Frequency).**

Phương pháp này được sử dụng đối với các item có dữ liệu ở dạng văn bản.

Công thức:

$$TF(t, d) = \frac{\text{Số lần xuất hiện của từ } t \text{ trong văn bản } d}{\text{Tổng số từ trong văn bản } d}$$

$$IDF(t, D) = \log \frac{\text{Tổng số văn bản trong tập mẫu } D}{\text{Số văn bản chứa từ } t}$$

$$TF - IDF(t, d, D) = TF(t, d) . IDF(t, D)$$

với:

t: từ đang xét

d: văn bản đang xét

D: tập mẫu

▪ **Biểu diễn nhị phân (One-Hot Encoding).**

Phương pháp này được sử dụng đối với các item có cấu trúc đơn giản, chẳng hạn như giá trị phân loại.

Ví dụ:

Movie	Adventure	Action	Science-Fiction	Drama	Crime	Thriller
Star Wars IV	1	1		1	0	0
Saving Private Ryan	0	0		0	1	0
American Beauty	0	0		0	1	0
City of Gold	0	0		0	1	1
Interstellar	0	0		1	1	0
The Matrix	1	1		1	0	1

Ảnh: Ví dụ về One-Hot Encoding

Trong ví dụ ở hình trên, ta có thể thu được các vector đặc trưng của từng movie như sau:

- Star Wars IV: [1, 1, 1, 0, 0, 0]
- Saving Private Ryan: [0, 0, 0, 1, 0, 0]
- American Beauty: [0, 0, 0, 1, 0, 0]
- City of God: [0, 0, 0, 1, 1, 0]
- Interstellar: [0, 0, 1, 1, 0, 0]
- The Matrix: [1, 1, 1, 0, 0, 1]

Sau khi đã có được các vector đặc trưng thì ta có thể sử dụng các phương pháp như Độ tương đồng Cosine để tính toán khoảng cách (độ tương đồng) giữa chúng.

Công thức Độ tương đồng Cosine (**Cosine Similarity**)

$$\text{similarity}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2 \sum_{i=1}^n B_i^2}}.$$

Trong đó A, và B là vectors tương ứng với 2 users bắt kì đã được chuẩn hoá.

2.3.3.2. Tiến hành áp dụng các thuật toán để dự đoán hoặc gợi ý

Sau khi đã có được độ tương đồng của các items, ta có thể áp dụng các thuật toán để tiến hành dự đoán kết quả đánh giá:

- **KNN (K-Nearest Neighbors):** là một thuật toán học máy không tham số (*non-parametric*), có thể được sử dụng để tìm ra các mục tương tự nhất (nearest neighbors) dựa trên độ tương đồng đã được tính toán, từ đó gợi ý các mục mà người dùng có thể quan tâm.

Bước 1: Chọn tham số K, tham số này có thể được thử nghiệm nhiều giá trị để chọn ra số K có kết quả gợi ý tốt nhất.

Bước 2: Sau khi đã có K, điểm dự đoán có thể được tính dựa trên trung bình trọng số của các điểm số từ K mục lân cận với công thức:

$$\text{Predicted Rating}(i) = \frac{\sum_{j \in N} (\text{Similarity}(i, j) \cdot \text{Rating}(j))}{\sum_{j \in N} |\text{Similarity}(i, j)|}$$

trong đó:

i : mục cần dự đoán

j : các mục lân cận K gần nhất

N : tập hợp K mục lân cận

$\text{Similarity}(i, j)$: Độ tương đồng giữa i và j

$\text{Rating}(j)$: Điểm đánh giá của người dùng cho mục j

Bước 3: Sau khi tính toán xong có thể xếp hạng các mục dựa trên điểm số dự đoán và đề xuất các mục có điểm dự đoán cao nhất cho người dùng.

2.3.4. Ưu điểm và nhược điểm

- **Ưu điểm:**

- Do hệ thống dựa vào các thuộc tính của item và sở thích cũng như tương tác của chính người dùng đang sử dụng hệ thống để đưa ra dự đoán nên sẽ giúp giải quyết được vấn đề Cold-start khi có một item mới được thêm vào hệ thống.
- Hệ thống có thể hoạt động ổn định, chính xác trong các tình huống khi dữ liệu người dùng còn hạn chế hoặc không có sẵn vì hệ thống tập trung vào từng người dùng cụ thể, không phụ thuộc vào cộng đồng.

- **Nhược điểm:**

- Các gợi ý có thể bị thiếu tính đa dạng và bất ngờ do hệ thống chỉ cung cấp những gợi ý dựa trên những sở thích, tương tác hiện tại của người dùng.
- Không giải quyết được vấn đề Cold-start khi có người dùng mới vì hệ thống phải dựa vào các hành vi, sở thích của người dùng để tiến hành đề xuất.

2.4. Collaborative filtering

Khi xây dựng recommender system sử dụng Content based filtering sẽ có hai nhược điểm chính:

+ *Thứ nhất*, khi xây dựng mô hình cho một user, các hệ thống Content-based không tận dụng được thông tin từ các users khác. Những thông tin này thường rất hữu ích vì hành vi mua hàng của các users thường được nhóm thành một vài nhóm đơn giản; nếu biết hành

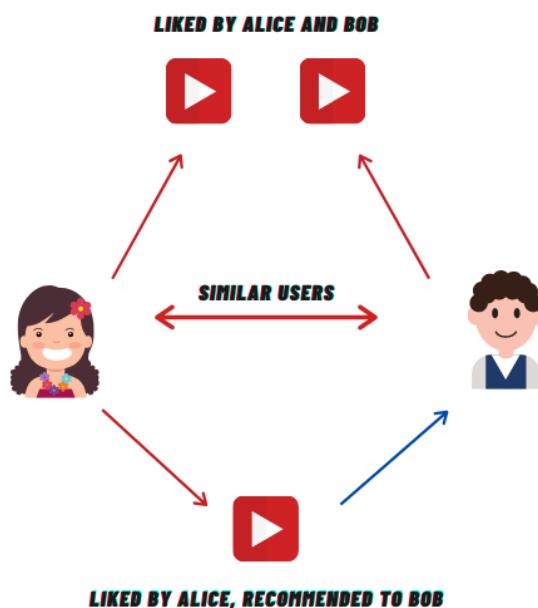
vì mua hàng của một vài *users* trong nhóm, hệ thống nên *suy luận* ra hành vi của những *users* còn lại.

+ *Thứ hai*, không phải lúc nào chúng ta cũng có *bản mô tả* cho mỗi *item*. Việc yêu cầu *users* gắn *tags* còn khó khăn hơn vì không phải ai cũng sẵn sàng làm việc đó; hoặc có làm nhưng sẽ mang xu hướng cá nhân. Các thuật toán NLP cũng phức tạp hơn ở việc phải xử lý các từ gần nghĩa, viết tắt, sai chính tả, hoặc được viết ở các ngôn ngữ khác nhau.

Những nhược điểm phía trên có thể được giải quyết bằng *Collaborative Filtering* (CF).

Trong Collaborative Filtering, ta thực hiện khuyến nghị cho user bằng cách dựa vào độ tương đồng giữa các users, hay nói cách khác ta sẽ đi tìm các users có sự tương đồng về sở thích với nhau. Nói một cách dễ hiểu, nếu ta biết được user A thuộc một nhóm người có sở thích B, thì ta chỉ cần khuyến nghị cho user A những items mà những người trong nhóm B thích.

COLLABORATIVE FILTERING



Ví dụ, cả Alice và Bob đều thích các phim về tình cảm lãng mạn (tức là đều đánh giá các bộ phim thuộc thể loại này từ 4 đến 5 sao). Dựa vào lịch sử xem phim của Alice, ta thấy Alice thích bộ phim A, vậy nhiều khả năng Bob cũng thích phim này, từ đó hệ thống sẽ đề xuất phim A cho Bob.

Trong collaborative filtering không sử dụng đặc trưng của các item để thực hiện khuyến nghị, thay vào đó sẽ thực hiện phân loại users vào các nhóm có cùng sở thích và khuyến nghị cho mỗi user dựa vào sở thích của nhóm mà user đó thuộc.

Có nhiều thuật toán và kỹ thuật để xây dựng Collaborative Filtering:

- Memory-based method: các phương pháp memory-based còn được gọi là neighborhood-based collaborative filtering algorithms (lọc cộng tác dựa trên lân cận). Đây là một trong những thuật toán học công tác trong đó đánh giá của các cặp user và item(user-item combinations) được dự đoán dựa trên các neighborhoods (hàng xóm) của chúng. Các neighborhoods này có thể được định nghĩa theo hai cách:
 - + User-based collaborative filtering (sẽ được trình bày trong phần sau)
 - + Item-based collaborative filtering (sẽ được trình bày trong phần sau)

Ưu điểm của memory-based method là chúng dễ triển khai và các gợi ý tạo ra thường khá dễ để giải thích. Tuy nhiên phương pháp này không hoạt động tốt khi dữ liệu thưa thớt.

- Model-based methods:
 - + Phương pháp dựa trên mô hình (Model-based methods): đối với các phương pháp dựa trên mô hình, machine learning và data mining được sử dụng để làm các mô hình dự đoán. Do đó, ta có thể sử dụng Matrix Factorization (phân rã mã trận), các mô hình deep learning như Autoencoders, RNN, CNN,... để thực hiện xây dựng hệ thống khuyến nghị.

2.5. Neighborhood based collaborative filtering

2.5.1. Giới thiệu

Neighborhood-Based Collaborative Filtering (lọc cộng tác dựa trên lân cận) là một trong những thuật toán ra đời sớm nhất được phát triển cho phương pháp lọc cộng tác khi nó đã được giới thiệu từ năm 1992 trong bài báo “Using Collaborative Filtering to Weave an Information Tapestry” bởi David Goldberg, David Nichols, Brian M. Oki, và Douglas B. Terry.

2.5.2. Ý tưởng

Neighborhood-Based Collaborative Filtering dựa trên ý tưởng rằng những người dùng tương tự sẽ có các hành vi tương tự, và các sản phẩm tương tự sẽ nhận được các đánh giá tương tự.

Neighborhood-Based Collaborative Filtering có 2 loại chính: user-based collaborative filtering và item-based collaborative filtering.

+ User-based collaborative filtering: xác định mức độ quan tâm của mỗi *user* tới một *item* dựa trên mức độ quan tâm của *similar users* tới *item* đó. Ví dụ, khi muốn gợi ý cho user A, ta sẽ đi tìm nhóm các users khác tương đồng với user A đó dựa vào các sản phẩm mà user A đã đánh giá. Từ đó, hệ thống sẽ thực hiện lấy các sản phẩm *nhóm user tương đồng* đó *đã đánh giá cao* và *gợi ý cho user A*.

+ Item-based collaborative filtering: thay vì xác định *user similarities* như user-based collaborative filtering, hệ thống sẽ xác định *item similarities*. Từ đó, hệ thống gợi ý những *items gần giống* với những *items* mà user có mức độ quan tâm cao.

Ví dụ: Người dùng A đã mua *item_1* và *item_2*. Hệ thống tìm ra những sản phẩm thường xuyên được mua cùng với các sản phẩm này này. Nó phát hiện rằng *item_3* và *item_4* thường được mua cùng với *item_1* và *item_2*. Dựa trên sự tương đồng giữa các sản

phẩm, hệ thống gợi ý item _3 và item _4 cho người dùng A, mặc dù người dùng này chưa từng tương tác với chúng trước đó.

2.5.3. Thuật toán

- User-based collaborative filtering

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

Bảng trên ghi lại đánh giá của user ui với item ij . Nhiệm vụ của hệ thống là dựa vào các ô đã có giá trị trong ma trận trên (dữ liệu thu được từ trong quá khứ), thông qua mô hình đã được xây dựng, dự đoán các ô còn trống (của user hiện hành), sau đó sắp xếp kết quả dự đoán (ví dụ, từ cao xuống thấp) và chọn ra Top-N items theo thứ tự rating giảm dần, từ đó gợi ý chúng cho người dùng.

Công việc quan trọng nhất phải làm trước tiên trong User-user Collaborative Filtering là phải xác định được *sự giống nhau* (*similarity*) giữa hai *users*.

Để đo *similarity* giữa hai *users*, cách thường làm là xây dựng vector đặc trưng cho mỗi user rồi áp dụng một hàm có khả năng đo *similarity* giữa hai vectors đó. Với mỗi user, thông tin duy nhất chúng ta biết là các *ratings* mà *user* đó đã thực hiện. Tuy nhiên, khó khăn là các cột này thường có rất nhiều *mising ratings* vì mỗi *user* thường chỉ *rated* một số lượng rất nhỏ các *items*. Cách khắc phục là bằng cách nào đó, ta *giúp* hệ thống *điền* các giá trị này sao cho việc *điền* không làm ảnh hưởng nhiều tới *sự giống nhau* giữa hai vector. Việc *điền* này chỉ phục vụ cho việc tính *similarity* chứ không phải là *suy luận* ra giá trị cuối cùng. Đơn giản nhất có thể thay giá trị ‘0’ hay một cách khác là ‘2.5’ – giá trị trung bình

giữa 0 và 5. Tuy nhiên, cách tính này có độ chính xác thấp vì những giá trị này sẽ hạn chế với những users dễ hoặc khó tính.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
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
	↓	↓	↓	↓	↓	↓	↓
\bar{u}_j	3.25	2.75	2.5	1.33	2.5	1.5	3.33

Thay vào đó, ta thực hiện tính trung bình cộng rating của mỗi user (như hàng dưới cùng của hình trên)

	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 đó, ta thực hiện trừ rating của mỗi cặp user-item đi giá trị trung bình rating của user đó, còn các cặp user-rating chưa có giá trị (?) ta sẽ gán cho chúng giá trị 0.

Mục đích của cách xử lý này là:

- Phân loại ratings thành 2 loại: giá trị âm (user không thích item) và dương (user thích item). Các giá trị bằng 0 là những item chưa được đánh giá.
- Số chiều của ma trận user-item thường rất lớn, trong khi lượng ratings biết trước thường rất nhỏ so với kích thước của ma trận. Nếu thay dấu ‘?’ bằng ‘0’ thì chúng ta có thể sử dụng sparse matrix để chỉ lưu các giá trị khác 0 và không quan tâm đến các giá trị bằng 0, nhờ đó có thể tiết kiệm bộ nhớ tính toán đáng kể, đồng thời tốc độ tính toán các phép toán trên ma trận thua thót nhanh hơn vì ít dữ liệu hơn.

Tính toán độ tương đồng

Sau khi đã chuẩn hóa dữ liệu, ta có thể thực hiện tính toán độ tương đồng giữa các user với nhau. Phương pháp phổ biến và hiệu quả đó là sử dụng **Cosine Similarity**.

$$\text{similarity}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2 \sum_{i=1}^n B_i^2}}.$$

Trong đó A, và B là vectors tương ứng với 2 users *bắt kè* **đã được chuẩn hóa** như ở trên. Cosine similarity tính toán góc giữa hai vector trong không gian đa chiều, và đó cũng chính là độ tương đồng giữa hai user.

Giả sử ta có 4 người dùng và các đánh giá về 3 bộ phim như sau:

Người dùng / Phim	Phim A	Phim B	Phim C
Người dùng 1	5	3	0
Người dùng 2	4	0	2
Người dùng 3	3	4	5
Người dùng 4	0	5	3

- **Bước 1:** Xây dựng vector cho người dùng 1 và người dùng 2.
- Người dùng 1: [5,3,0][5, 3, 0][5,3,0]
- Người dùng 2: [4,0,2][4, 0, 2][4,0,2]

Bước 2: Tính cosine similarity giữa người dùng 1 và người dùng 2:

$$A \cdot B = (5 \cdot 4) + (3 \cdot 0) + (0 \cdot 2) = 20$$

$$\|A\| = \sqrt{5^2 + 3^2 + 0^2} = \sqrt{25 + 9} = \sqrt{34} \approx 5.83$$

$$\|B\| = \sqrt{4^2 + 0^2 + 2^2} = \sqrt{16 + 4} = \sqrt{20} \approx 4.47$$

$$\text{CosineSimilarity} = \frac{20}{3,83 \cdot 4,47} \approx 0.77$$

Vì giá trị cosine similarity gần 1, người dùng 1 và người dùng 2 có mức độ tương đồng cao, nghĩa là họ có xu hướng đánh giá các bộ phim một cách tương tự.

Ngoài ra, có thể sử dụng **Person corelation**, một phương pháp tính similarity cũng khá tương đồng với cosine similarity

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

Giả sử:

- Người dùng A đã đánh giá các sản phẩm P1, P2, P3.
- Người dùng B đã đánh giá các sản phẩm P1, P3, P4.
- Người dùng C chỉ đánh giá P1.

Các sản phẩm mà không được đánh giá ta sẽ chuẩn hóa thành 0.

Khi tính Cosine Similarity giữa các người dùng, lấy ví dụ là giữa người dùng A và người dùng B, Cosine không xử lý các giá trị thiếu, vì vậy nó sẽ tính toán độ tương đồng giữa tất cả các sản phẩm mà cả hai người dùng đã đánh giá. Điều này sẽ làm cho độ tương đồng giữa người dùng A và B bị giảm đi, vì hệ số tương quan giữa các sản phẩm sẽ bị ảnh hưởng bởi các giá trị 0 (là các sản phẩm mà 2 người dùng chưa có đánh giá chung, ví dụ với người dùng A là sản phẩm P4, với người dùng B là sản phẩm P2), mặc dù trên thực tế hai người dùng này có thể có sự tương đồng nếu chỉ xét các sản phẩm chung mà họ đã đánh giá.

Tuy nhiên nếu sử dụng **Person corelation**, nó sẽ bỏ qua các sản phẩm P4 (vì người dùng A không đánh giá nó) và sản phẩm P2 (vì người dùng B không đánh giá nó), cho phép nó đánh giá chính xác hơn mức độ tương đồng của A và B dựa trên các sản phẩm mà cả hai người đã đánh giá.

Trong các tình huống dữ liệu thưa, nơi mà người dùng chỉ đánh giá một số ít sản phẩm, Pearson Correlation là lựa chọn tốt hơn so với các phương pháp như Cosine Similarity vì nó chỉ tính toán sự tương quan dựa trên các đánh giá có sẵn và không bị ảnh hưởng bởi sự thiếu hụt dữ liệu, giúp tránh đưa ra các kết luận sai lệch khi so sánh giữa các người dùng.

Dự đoán ratings

Ta sẽ dự đoán ratings của một user với mỗi item dựa trên k users gần nhất (neighbor users), tương tự như phương pháp K-nearest neighbors (KNN):

$$\hat{y}_{i,u} = \frac{\sum_{u_j \in N(u,i)} \bar{y}_{i,u_j} \text{sim}(u, u_j)}{\sum_{u_j \in N(u,i)} |\text{sim}(u, u_j)|}$$

Trong đó, $N(u, i)$ là tập k users gần nhất (có độ tương đồng cao nhất) với user u và đã từng đánh giá item i. Ví dụ, dự đoán rating của user u1 cho item i1 với k = 2 là số users gần nhất:

- Bước 1: Xác định các users đã rated cho i1, đó là u0, u3, u5
- Bước 2: Lấy similarities của u1 với u0, u3, u5 từ ma trận sau:

	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	-0.40	-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	-0.40	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

Ảnh: Ma trận tương đồng giữa các cặp user được tính bằng cosine similarity

Kết quả lần lượt là: {u0, u3, u5}: {0.83, -0.4, -0.23}. Với k = 2. Chọn 2 giá trị lớn nhất là 0.83 và -0.23, tương ứng với các users u0, u5. Hai users này có ratings (đã qua bước chuẩn hóa) với i1 là: {u0, u5} {0.75, 0.5}

- Bước 3: Tính ratings theo công thức:

$$\hat{y}_{i_1, u_1} = \frac{0.83 \times 0.75 + (-0.23) \times 0.5}{0.83 + |-0.23|} \approx 0.48$$

Thực hiện dự đoán cho các trường hợp missing ratings (chưa có dự đoán), ta sẽ thu được ma trận ratings matrix:

	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

Ảnh: ma trận dự đoán đánh giá (dùng dữ liệu đã chuẩn hóa)

Tuy nhiên, vì ta dùng các giá trị ratings từ ma trận user-item đã qua chuẩn hóa, vì vậy ta cần convert ngược lại thành ma trận đánh giá ban đầu:

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	5	5	2	0	1	1.68	2.70
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	2.10	4
i_4	2	0	4	2.9	4.06	3.10	5

Ảnh: ma trận dự đoán đánh giá hoàn chỉnh

Hạn chế của User-based collaborative filtering:

Trên thực tế, số lượng *users* luôn lớn hơn số lượng *items* rất nhiều. Dẫn đến việc ma trận tương đồng trở nên rất lớn:

	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	-0.40	-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	-0.40	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

Ảnh: ma trận tương đồng giữa tất cả các cặp user được xây dựng ra sau khi tính toán độ tương đồng của từng cặp user

- Item-based collaborative filtering

Item-based collaborative giải quyết nhược điểm của User-based collaborative bằng cách tính toán *similarity* giữa các *items* rồi *recommend* những *items* gần giống với *item* yêu thích của một *user*. Hay nói cách khác, thay vì tìm xem user này tương đồng với những users nào dựa vào items mà các users đã rating, thì ta làm ngược lại đó là tìm xem item này tương đồng với những items nào dựa vào các rating mà users đã tạo.

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

Lấy ví dụ, User-based sẽ thực hiện tính toán tương đồng giữa user và các user dựa cột rating ứng với mỗi user. Ngược lại Item-based sẽ thực hiện tính toán tương đồng giữa item và ik dựa vào hàng rating ứng với mỗi item.

Điều này sẽ mang lại những lợi ích sau đây:

- + Thông thường, số lượng items sẽ nhỏ hơn đáng kể so với số lượng users, điều này dẫn đến ma trận tương đồng item-item được lập ra sẽ nhỏ hơn nhiều, giúp tiết kiệm chi phí tính toán.

- + Số lượng items nhỏ hơn đáng kể so với users cũng giúp giải quyết vấn đề thiếu hụt dữ liệu, bởi vì nếu xét theo hàng rating ta sẽ luôn có nhiều rating được đánh giá hơn là xét theo cột, bởi vì một item thường sẽ có nhiều người đánh giá, ngược lại một người chưa chắc đã đánh giá cho nhiều item.

- + Bản chất của các items là ít thay đổi, trong khi đó nếu ta tập trung vào users, vì users là con người nên sở thích của họ hoàn toàn có thể thay đổi theo thời gian, dẫn đến đánh giá khuyến nghị không còn chính xác. Ví dụ: một user nào đó tuần trước có sở thích đọc truyện tranh, nhưng một tuần sau đó nữa anh ta không còn sở thích đó và chuyển sang sở thích khác. Vậy thì việc dựa vào sở thích của anh ta để khuyến nghị cho những người dùng khác khi sử dụng User-based collaborative trở nên không còn chính xác nữa.

	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

Các bước xây dựng hệ khuyến nghị sử dụng Item-based filtering cũng gần giống với User-based filtering, chỉ cần thay đổi xét từ cột rating thành hàng rating, các bước tính toán còn lại tương tự:

	u_0	u_1	u_2	u_3	u_4	u_5	u_6	
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

Ảnh: *Tính trung bình rating của mỗi item i hóa*

Đầu tiên ta cũng thực hiện chuẩn hóa dữ liệu bằng cách tính trung bình rating của mỗi item, sau đó trừ các rating cho số trung bình ứng với mỗi item, với các missing data (?) ta gán giá trị 0.

	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

Ảnh: ma trận tương đồng

Tiếp đến ta hoàn toàn có thể xây dựng ma trận tương đồng bằng công thức cosine similarity hoặc các công thức khác.

2.5.4. Ưu điểm và nhược điểm

Ưu điểm của Neighborhood-based Collaborative Filtering:

- Dễ hiểu và dễ triển khai: là phương pháp khá dễ hiểu so với các phương pháp sử dụng học máy. Không yêu cầu quá nhiều kiến thức phức tạp, chỉ cần tính toán sự tương đồng giữa người dùng hoặc sản phẩm. Hiện nay đã có rất nhiều thư viện python giúp đơn giản hóa việc triển khai Neighborhood-based Collaborative Filtering như **Surprise**, **Scikit-learn**, **TFRS**...
- Đưa ra các khuyến nghị cá nhân hóa cao: Phương pháp này có thể cung cấp các khuyến nghị rất chính xác cho người dùng, vì nó dựa vào dữ liệu lịch sử của người dùng và những người dùng tương tự để tạo ra các đề xuất.

Nhược điểm của Neighborhood-based Collaborative Filtering:

- Tính thay đổi theo thời gian: sở thích của người dùng có thể bị thay đổi theo thời gian, phương pháp này có thể gặp khó khăn trong việc bắt kịp với sự thay đổi đó.
- Không có khả năng xử lý cold start: khó khăn khi khuyến nghị cho người dùng mới hoặc sản phẩm mới vì không có đủ dữ liệu để tìm ra sự tương đồng.

2.5.5. Case study

- Amazon từng sử dụng Item-based collaborative filtering cho hệ thống recommender của họ vào năm 2003. Điều này đã được đăng trong bài báo: Amazon.com recommendations: item-to-item collaborative filtering xuất bản bởi G. Linden, B. Smith và J. York.

2.6. Matrix factorization method

2.6.1. Giới thiệu

Như đã đề cập ở phần 2.4, trong Collaborative filtering có 2 phương pháp để triển khai đó là: Memory-based method và Model-based method.

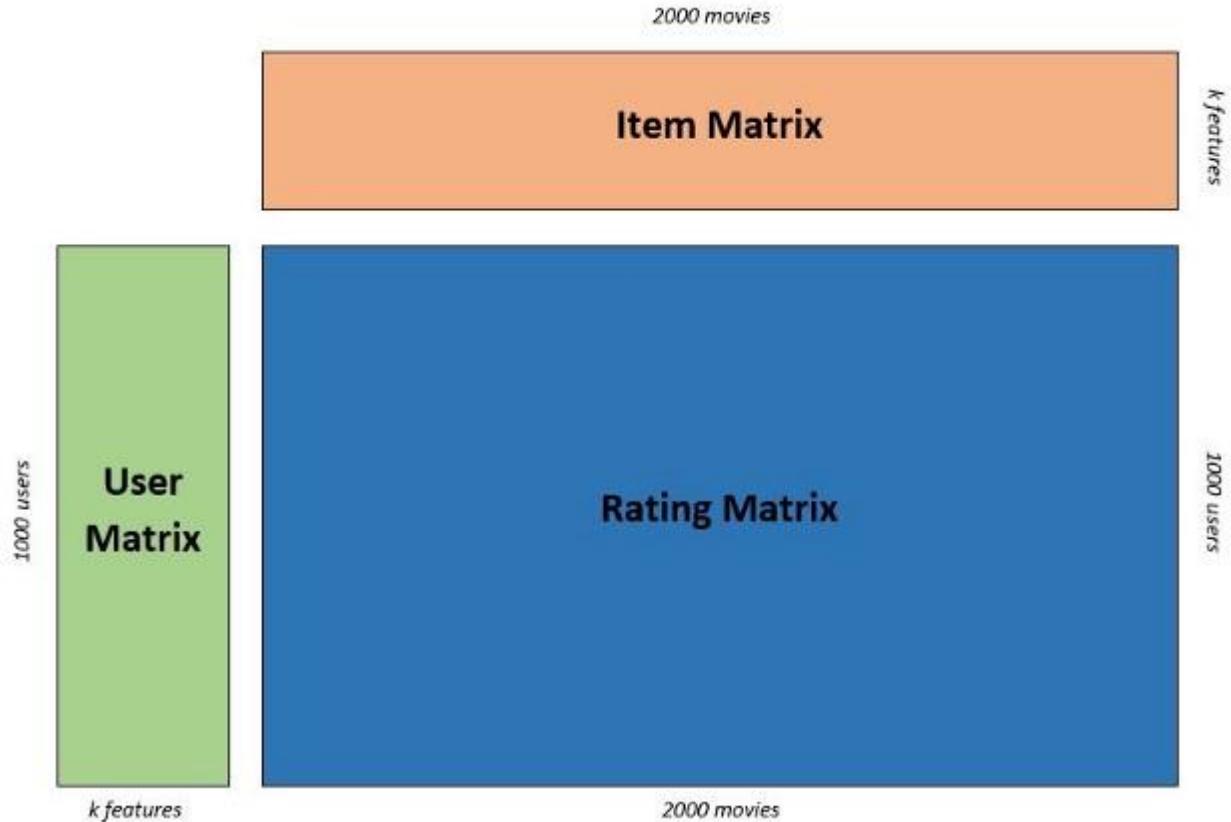
Trong khi Neighborhood-based Collaborative Filtering đã đề cập ở phần 2.4 là một thuật toán thuộc Memory-based method, Matrix factorization (kỹ thuật phân rã ma trận) trình bày ở phần này thuộc Model-based method.

Matrix Factorization (Phân rã ma trận) là một phương pháp toán học mạnh mẽ, thường được sử dụng trong lĩnh vực khai thác dữ liệu và học máy, đặc biệt là trong các hệ thống gợi ý. Kỹ thuật này giúp phân rã một ma trận lớn và phức tạp thành hai hoặc nhiều ma trận nhỏ hơn, qua đó trích xuất các thông tin tiềm ẩn (latent factors) từ dữ liệu. Các yếu tố tiềm ẩn (latent factors) có thể gọi theo cách khác là các đặc trưng (đặc điểm). Đặc trưng là đặc điểm nổi bật của một mục hay người dùng. Trong trường hợp của hệ thống giới thiệu phim, các đặc trưng của phim có thể là thể loại, diễn viên, cốt truyện,...

Sau cuộc thi “Netflix Price Challenge” - một cuộc thi do Netflix tổ chức, với mục tiêu cải thiện hệ thống gợi ý của họ bằng cách dự đoán chính xác hơn các xếp hạng phim của người dùng, Matrix Factorization là một trong những kỹ thuật lọc cộng tác được sử dụng phổ biến nhất. Bởi vì một trong các kỹ thuật cụ thể của Matrix Factorization là Funk-SVD - một biến thể của SVD (Singular Value Decomposition) đã được nhóm chiến thắng là BellKor’s Pragmatic Chaos sử dụng kết hợp với nhiều mô hình khác để giành giải thưởng.

2.6.2. Ý tưởng

Matrix Factorization (Kỹ thuật phân rã ma trận) là kỹ thuật để phân tích và tóm gọn thông tin từ một ma trận lớn, bằng cách "chia nhỏ" nó thành hai hoặc nhiều ma trận nhỏ hơn, dễ xử lý hơn. Điều đó tương tự như việc tách một công thức phức tạp thành hai công thức đơn giản hơn nhưng vẫn giữ được ý nghĩa.



Giả sử ta đang xây dựng hệ thống gợi ý phim, ta có ma trận đánh giá với hàng là các user, cột là các item, và các ô giao giữa user và item là các rating tương ứng của user cho item.

Tuy nhiên, rõ ràng không phải ai cũng đánh giá tất cả các phim (bảng có nhiều ô trống). Nhiệm vụ đưa ra là dự đoán cho những ô trống đó.

Đối với phương pháp Neighborhood-based Collaborative Filtering thì ý tưởng của nó sẽ là đi tính độ tương đồng giữa item-item hoặc user-user trên “Rating Matrix”. Khác với Neighborhood-based Collaborative Filtering, Matrix Factorization sẽ thực hiện phân rã bảng thành hai bảng nhỏ hơn:

- Một bảng User Matrix chứa đặc điểm của người dùng (ví dụ: Alice thích hành động và hài hước).

- Một bảng Item Matrix chứa đặc điểm của phim (ví dụ: Phim A thuộc thể loại hành động, Phim B là hài hước).

Khi nhân hai bảng này lại, ta có thể dự đoán: Alice sẽ thích phim nào dựa trên sự tương đồng giữa sở thích của cô ấy và đặc điểm của phim.

Như đã trình bày trước đó, Matrix Factorization sẽ tự động trích xuất các thông tin tiềm ẩn (latent factors) từ dữ liệu, điều đó có nghĩa là Matrix Factorization sẽ tự động học và không cần biết trước người dùng thích thể loại nào hay phim thuộc thể loại gì.

Điều đó tương tự như việc ta phân tích bảng Rating Matrix thành:

- Người dùng = Sở thích (hành động, lãng mạn, giật gân).
- Phim = Thể loại (hành động, thể thao, lịch sử).

Khi ta biết được người dùng có sở thích phim “hành động”, ta có thể gợi ý phim “hành động” cho người dùng đó, trong khi không cần phải hiểu “hành động” là gì, chỉ cần để máy tự học các yếu tố này từ dữ liệu.

2.6.3. Thuật toán

Trong phương pháp Content based filtering được đề cập ở phần 2.3 , mỗi *item* được mô tả bằng một vector x được gọi là *item profile*. Trong Content based filtering, ta cần tìm một vector hệ số w tương ứng với mỗi *user* sao cho y là *rating* đã biết mà *user* đó cho *item* xấp xỉ với:

$$y \approx \mathbf{xw}$$

	A	B	C	D	E	F
Mưa nửa đêm	5	5	0	0	1	?
Cô úa	5	?	?	0	?	?
Vùng lá me bay	?	4	1	?	?	1
Con cò bé bé	1	1	4	4	4	?
Em yêu trường em	1	0	5	?	?	?

Ảnh: ma trận đánh giá với cột là các users, hàng là items, các ô là rating tương ứng

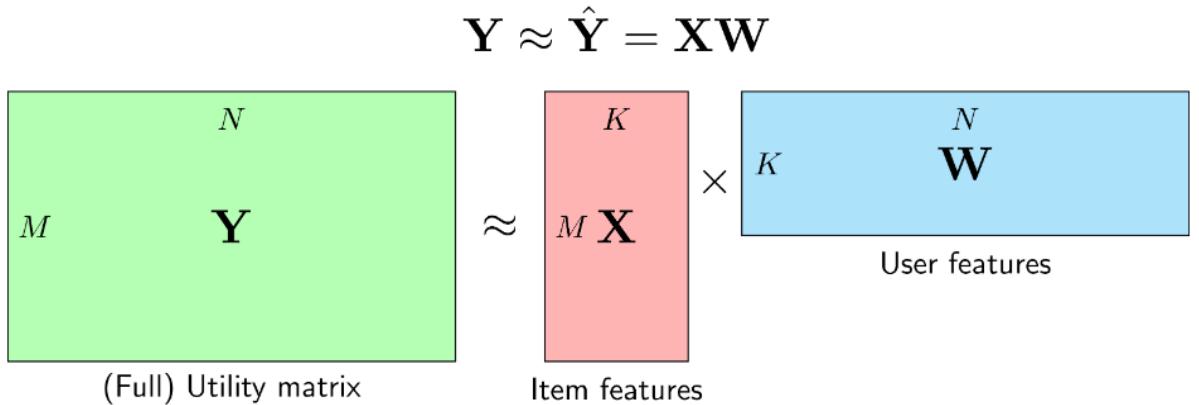
Nếu ta cũng mang công thức này vào Ma Trận Đánh Giá như hình trên , thì kết quả sẽ như sau:

$$\mathbf{Y} \approx \begin{bmatrix} \mathbf{x}_1 \mathbf{w}_1 & \mathbf{x}_1 \mathbf{w}_2 & \dots & \mathbf{x}_1 \mathbf{w}_N \\ \mathbf{x}_2 \mathbf{w}_1 & \mathbf{x}_2 \mathbf{w}_2 & \dots & \mathbf{x}_2 \mathbf{w}_N \\ \dots & \dots & \ddots & \dots \\ \mathbf{x}_M \mathbf{w}_1 & \mathbf{x}_M \mathbf{w}_2 & \dots & \mathbf{x}_M \mathbf{w}_N \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_M \end{bmatrix} [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \dots \quad \mathbf{w}_N] = \mathbf{XW}$$

Chú ý rằng, \mathbf{x} được xây dựng dựa trên thông tin mô tả của item và quá trình xây dựng này độc lập với quá trình đi tìm hệ số phù hợp cho mỗi user. Như vậy, việc xây dựng item profile đóng vai trò rất quan trọng và có ảnh hưởng trực tiếp lên hiệu năng của mô hình.Thêm nữa, việc xây dựng từng mô hình riêng lẻ cho mỗi user dẫn đến kết quả chưa thực sự tốt vì không khai thác được đặc điểm của những users gần giống nhau.

Bây giờ, giả sử rằng ta không cần xây dựng từ trước các item profile \mathbf{x} mà vector đặc trưng cho mỗi item này có thể được huấn luyện đồng thời với mô hình của mỗi user (ở đây là 1 vector hệ số). Điều này nghĩa là, biến số trong bài toán tối ưu là cả ma trận \mathbf{X} và ma

trận W; trong đó, X là ma trận của toàn bộ *item profiles*, mỗi **hàng** tương ứng với 1 *item*, W là ma trận của toàn bộ *user models*, mỗi **cột** tương ứng với 1 *user*.



Ảnh: Ma trận Y được tách thành X và W

Với cách làm này, chúng ta đang cố gắng xấp xỉ Ma Trận Đánh Giá Y bằng tích của hai ma trận X và W.

- Ý tưởng chính đằng sau Matrix Factorization cho Recommendation Systems là tồn tại các *latent features* mô tả sự liên quan giữa các *items* và *users*. Ví dụ với hệ thống gợi ý các bộ phim, tính chất ẩn có thể là *hình sự*, *chính trị*, *hành động*, *hài*, ...; cũng có thể là một sự kết hợp nào đó của các thể loại này; hoặc cũng có thể là bất cứ điều gì mà chúng ta không thực sự cần đặt tên. Mỗi *item* sẽ mang tính chất ẩn ở một mức độ nào đó tương ứng với các hệ số trong vector x của nó, hệ số càng cao tương ứng với việc mang tính chất đó càng cao. Tương tự, mỗi *user* cũng sẽ có xu hướng thích những tính chất ẩn nào đó và được mô tả bởi các hệ số trong vector w của nó. Hệ số cao tương ứng với việc *user* thích các bộ phim có tính chất ẩn đó. Giá trị của biểu thức xw sẽ cao nếu các thành phần tương ứng của x và w đều cao. Điều này nghĩa là *item* mang các tính chất ẩn mà *user* thích, vậy thì nên gợi ý *item* này cho *user* đó.

Ví dụ:

+ **Alice**: Thích phim hài (hệ số 0.8) và phim hành động (hệ số 0.7).

+ **Phim A**: Là phim hành động hài (hệ số hành động: 0.9, hệ số hài: 0.8).

Khi nhân hai vector:

$$x \cdot w = (0.9 \times 0.7) + (0.8 \times 0.8) = 0.63 + 0.64 = 1.27$$

Kết quả cao nhận được khá cao nên hệ thống sẽ gợi ý **Phim A** cho Alice.

- Trong các bài toán thực tế, số lượng *items* M và số lượng *users* N thường rất lớn. Việc tìm ra các mô hình đơn giản giúp dự đoán *ratings* cần được thực hiện một cách nhanh nhất có thể. **Neighborhood-based Collaborative Filtering** không yêu cầu việc *learning* quá nhiều, nhưng trong quá trình dự đoán (*inference*), ta cần đi tìm độ *similarity* của *user* đang xét với *toàn bộ* các *users* còn lại rồi suy ra kết quả. Ngược lại, với Matrix Factorization, việc *learning* có thể hơi phức tạp một chút vì phải lặp đi lặp lại việc tối ưu một ma trận khi cố định ma trận còn lại, nhưng việc *inference* đơn giản hơn vì ta chỉ cần lấy tích của hai vector xw , mỗi vector có độ dài K là một số nhỏ hơn nhiều so với M,N. Vậy nên quá trình *inference* không yêu cầu khả năng tính toán cao. Việc này khiến nó phù hợp với các mô hình có tập dữ liệu lớn.
- Việc lưu trữ hai ma trận X và W yêu cầu lượng bộ nhớ nhỏ khi so với việc lưu toàn bộ *Ma Trận tương đồng* trong **Neighborhood-based Collaborative Filtering**. Cụ thể, ta cần bộ nhớ để chứa $K(M+N)$ phần tử thay vì lưu M^2 hoặc N^2 của *Ma Trận tương đồng* phía **Neighborhood-based Collaborative Filtering**.

Hàm mất mát

Hàm mất mát trong Matrix Factorization được dùng để đo lường **độ chênh lệch giữa kết quả dự đoán và dữ liệu thực tế**. Đây là bước quan trọng giúp mô hình **học và tối ưu hóa** các tham số (các vector đặc trưng x và w) để đưa ra gợi ý chính xác hơn.

Matrix Factorization hoàn toàn có thể được tối ưu bằng Gradient Descent. Tại mỗi điểm dữ liệu tương ứng với (người dùng, sản phẩm, mức độ quan tâm), ta cần tính giá trị

ước lượng đánh giá (y) rồi xây dựng hàm mất mát cho điểm dữ liệu này dựa trên giá trị thực tế và giá trị dự đoán y. Tùy vào từng bài toán mà hàm mất mát có thể được xây dựng một cách khác nhau.

2.6.4. SVD (Singular Value Decomposition)

SVD (Singular Value Decomposition) là một kỹ thuật phân rã ma trận quan trọng trong đại số tuyến tính, được ứng dụng rộng rãi trong nhiều lĩnh vực như xử lý tín hiệu, nén dữ liệu, học máy, và đặc biệt là trong hệ thống gợi ý.

SVD giúp phân rã một ma trận A (kích thước $m \times n$) thành ba ma trận:

$$\mathbf{A}_{m \times n} = \mathbf{U}_{m \times m} \Sigma_{m \times n} (\mathbf{V}_{n \times n})^T$$

Trong đó, U,V là các *ma trận trực giao*, Σ là ma trận *đường chéo không vuông* với các phần tử trên đường chéo $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r \geq 0 = 0 = \dots = 0$, ma trận này chứa các giá trị đặc biệt gọi là **singular values**.

Trong lĩnh vực hệ thống gợi ý, SVD chính là một kỹ thuật cự thể của Matrix Factorization và được sử dụng để phân tích mối quan hệ giữa người dùng và sản phẩm, dựa trên ma trận đánh giá. Kỹ thuật này đã được ứng dụng thành công trong bài toán Netflix Prize.

$$\mathbf{A}_{m \times n} = \mathbf{U}_{m \times m} \times \begin{matrix} & \text{red} \\ & \text{pink} \\ \text{white} & \ddots \end{matrix} \times \Sigma_{m \times n} \times \mathbf{V}_{n \times n}^T$$

(m < n)

$$\mathbf{A}_{m \times n} = \mathbf{U}_{m \times m} \times \begin{matrix} & \text{red} \\ & \text{pink} \\ \text{white} & \ddots \end{matrix} \times \Sigma_{m \times n} \times \mathbf{V}_{n \times n}^T$$

(m > n)

Ảnh 2.6.4.1: mô tả SVD của ma trận $A_{m \times n}$ trong hai trường hợp: $m < n$ và $m > n$.

Trường hợp $m=n$ có thể xếp vào một trong hai trường hợp trên.

Để dễ hình dung, ta xét bối cảnh xây dựng recommender system cho hệ thống quản lý thư viện.

Ta sẽ xây dựng một Ma Trận đánh giá quen thuộc với :

Hàng: Đại diện cho từng độc giả.

Cột: Đại diện cho từng cuốn sách.

Ô: Điểm số mà độc giả đánh giá cho cuốn sách đó (hoặc điền ? nếu chưa đọc).

Tuy nhiên, bảng này có rất nhiều ô trống vì không ai đọc hết toàn bộ sách. Đây là nơi **SVD (Singular Value Decomposition)** xuất hiện để giúp dự đoán những điểm đánh giá còn thiếu và đưa ra gợi ý.

Bước đầu tiên, SVD sẽ thực hiện chia nhỏ bảng đánh giá lớn thành ba bảng nhỏ hơn để tìm hiểu:

Một bảng cho thấy mỗi độc giả "thích" các đặc điểm ẩn ở mức độ nào (tương ứng với bảng U trong ảnh 2.6.4.1)

Một bảng thể hiện mức độ "đậm đặc" của từng đặc điểm: quan trọng nhất, ít quan trọng hơn,... (tương ứng với bảng Σ trong ảnh 2.6.4.1)

Một bảng cho biết mỗi cuốn sách có các đặc điểm ẩn nào (tương ứng với bảng V trong ảnh 2.6.4.1)

Bước tiếp theo, bằng cách ghép các bảng nhỏ này lại, SVD dự đoán được những ô trống trong bảng gốc. Điều này giống như nói: "*Độc giả này thích phiêu lưu, cuốn sách này là phiêu lưu, vậy điểm đánh giá có thể là 4 sao.*" Từ bảng mới này, ta có thể gợi ý được cho độc giả các cuốn sách có điểm dự đoán cao, mặc dù họ chưa từng đọc.

SVD có thể dễ dàng triển khai dựa vào thư viện **Surprise** và sẽ được triển khai ở phần sau trong tài liệu này.

2.6.5. Ưu và nhược điểm

Ưu điểm của Matrix Factorization:

- Có khả năng tự học các đặc điểm dựa trên các latent features, khác với Content-based filtering phải học dựa trên đặc điểm đầu vào của các items như màu sắc, category,...

- Hiệu quả với dữ liệu lớn và thưa vì nó chỉ cần học các đặc điểm ẩn từ một phần nhỏ của dữ liệu, và nhờ việc phân tách một ma trận lớn ban đầu thành nhiều ma trận nhỏ giúp tiết kiệm bộ nhớ và chi phí tính toán đáng kể.

- Vì Matrix factorization trích xuất các latent features nên nó có thể tiết lộ các điểm tương đồng và sở thích tiềm ẩn giữa người dùng và sản phẩm, từ đó nâng cao chất lượng và sự đa dạng của các gợi ý.

Matrix factorization cũng có một số nhược điểm đối với hệ thống gợi ý:

- Thứ nhất, nó có thể gặp phải vấn đề overfitting và underfitting, điều này có thể ảnh hưởng đến độ chính xác và khả năng tổng quát của các gợi ý. Overfitting xảy ra khi các vector đặc trưng quá khớp với dữ liệu và bắt được nhiều hoặc các giá trị ngoại lai, trong

khi underfitting xảy ra khi các vector đặc trưng khớp với dữ liệu quá kém và bỏ lỡ thông tin quan trọng. Để tránh những vấn đề này, matrix factorization cần cân bằng giữa việc khớp dữ liệu và điều chỉnh các vector đặc trưng.

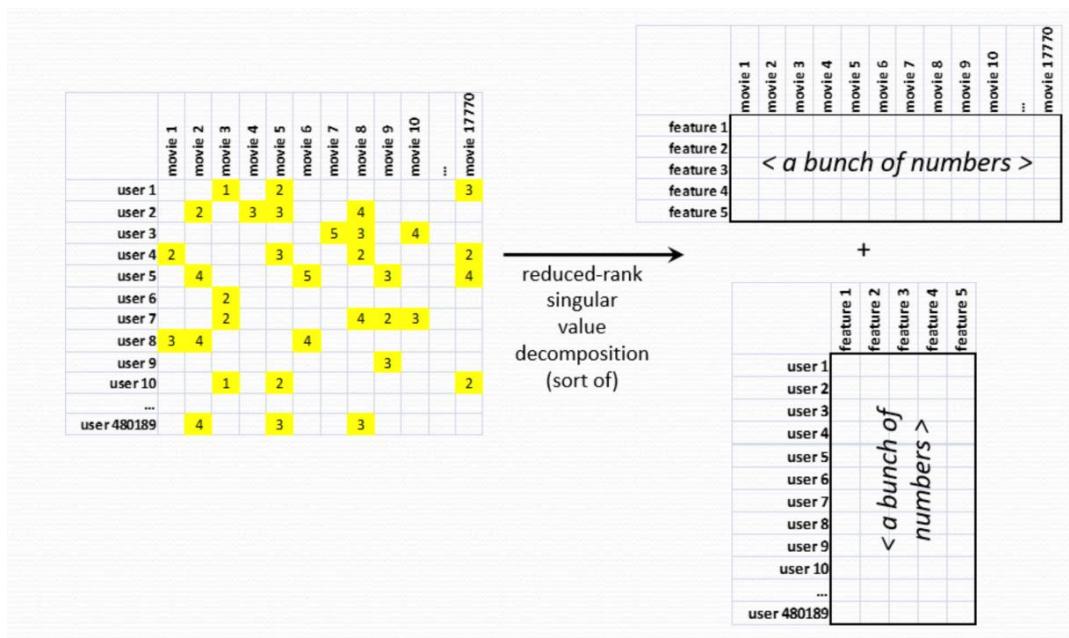
- Thứ hai, nó có thể nhạy cảm với việc lựa chọn các tham số, như số lượng đặc trưng, learning rate,... Những tham số này có thể ảnh hưởng đến hiệu suất và khả năng hội tụ của thuật toán matrix factorization. Để tìm các tham số tối ưu, matrix factorization cần thực hiện kiểm tra chéo (cross-validation) hoặc tìm kiếm lưới (grid search), điều này có thể tốn thời gian và tài nguyên tính toán.

- Thứ ba, Matrix Factorization chỉ sử dụng phép tuyến tính để tính điểm đánh giá dự đoán, nhưng trong thực tế điểm số đánh giá có thể phụ thuộc vào các sự tương tác phức tạp giữa các đặc trưng mà không thể chỉ giải thích bằng một phép toán tuyến tính đơn giản.

2.6.6. Case study

Như đã đề cập ở phần 2.6.1, SVD (một kỹ thuật cụ thể của Matrix Factorization) đã từng thắng giải Netflix Prize.

Simon Funk, một trong những thí sinh, đã viết một chương trình C ngắn gọn và đơn giản, thực hiện phương pháp gradient descent ngẫu nhiên trên một mô hình đơn giản. Mô hình này cho rằng ma trận đánh giá phim $C \times M$ của C khách hàng và M bộ phim có thể được phân rã gần đúng thành tích của ma trận $C \times K$ (với C nhiều hơn K) và ma trận $K \times M$ (M nhiều hơn K).



Ảnh: phân rã ma trận ban đầu thành các ma trận nhỏ hơn sử dụng SVD

Ý tưởng này chính là của SVD nhằm phân rã ma trận thành các ma trận nhỏ hơn, trong đó có:

- + Một ma trận chứa các đặc trưng đã học về từng khách hàng còn
- + Một ma trận chứa các đặc trưng đã học về từng bộ phim.

Khi nhân các đặc trưng này với nhau, chúng ta có thể dự đoán mức độ yêu thích của khách hàng đối với một bộ phim.

Điều đặc biệt là không cần gán nhãn thể loại phim mà hệ thống vẫn tự động học các đặc trưng này trong quá trình huấn luyện.

2.7. Deep learning method

2.7.1. Giới thiệu

Ngày nay, Deep Learning đã đi sâu vào rất nhiều lĩnh vực, giải quyết được rất nhiều bài toán khó trong thực tế, từ Computer Vision cho đến Natural Language Processing.

Như một điều tất yếu lĩnh vực Recommender System cũng được hưởng lợi từ sự phát triển mạnh mẽ đó. Có thể nói rằng, với sự tham gia của Deep Learning, các hệ thống khuyến nghị hiện đại ngày nay như Youtube, Amazon, Netflix, Facebook, ... đã bỏ xa các phương pháp truyền thống trong cuộc đua mang lại lợi ích thiết thực cho các doanh nghiệp.

2.7.2. Giới thiệu

- Lịch sử ra đời

RBM ban đầu được đề xuất với tên gọi Harmonium bởi Paul Smolensky vào năm 1986, và trở nên nổi bật sau khi Geoffrey Hinton cùng các cộng sự áp dụng các thuật toán học nhanh cho chúng vào giữa những năm 2000. RBM đã được ứng dụng trong nhiều lĩnh vực như giảm chiều dữ liệu, phân loại, bộ lọc cộng tác, học đặc trưng, mô hình hóa chủ đề, miễn dịch học, và thậm chí cả cơ học lượng tử nhiều vật thể.

Trong lĩnh vực recommender system, nó đã được sử dụng từ những năm 2007, khá lâu trước khi AI trở nên phổ biến như ngày nay. Tuy vậy, RBM vẫn là kỹ thuật được sử dụng trong hệ thống khuyến nghị ngày nay.

Tại Netflix Prize, sau khi cuộc thi diễn ra, Netflix nhận ra rằng Matrix Factorization (cụ thể là SVD) cùng với RBM đã cho kết quả tốt nhất ở cuộc thi này ở điểm số RMSE. Họ nhận ra rằng, việc kết hợp giữa Matrix Factorization và RBM đã cho ra kết quả tốt hơn đáng kể, từ 8.9 xuống còn 8.8 điểm RMSE (RMSE càng thấp tức là càng tốt). Vài năm về trước, Netflix xác nhận rằng họ vẫn đang sử dụng RBM như một phần trong hệ thống khuyến nghị mà họ đang triển khai ở môi trường production.

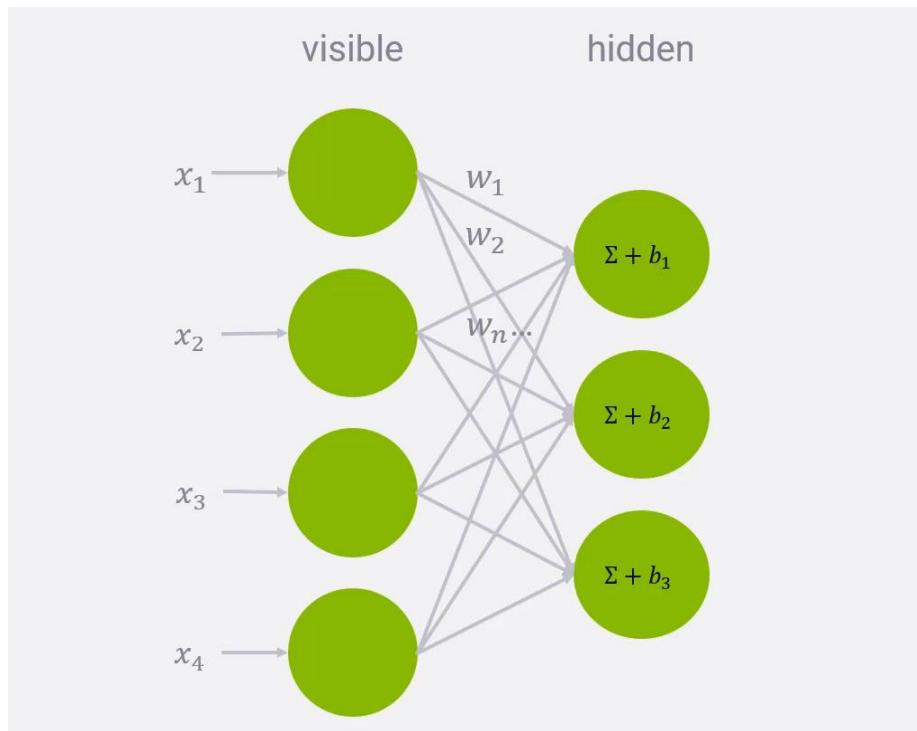
- Ý tưởng

Restricted Boltzmann Machine là một unsupervised deep learning model (mô hình học sâu không giám sát), trong đó mỗi nút được kết nối với tất cả các nút khác. Restricted Boltzmann Machine - RBM là mô hình đồ thị xác suất không hướng, bao gồm một lớp các biến quan sát và một lớp các biến ẩn. Không có nút nào trong cùng một lớp được kết nối với nhau, vì lý do này mà chúng được gọi là Restricted Boltzmann Machine. Chúng ta cũng

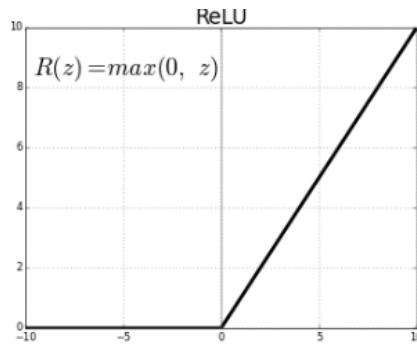
có thể nghĩ về RBM như việc chuyển đổi dữ liệu đầu vào từ dimension sang các dimensions khác. Chúng chủ yếu được sử dụng như là các khối xây dựng cho Deep Network.

RBM chủ yếu được sử dụng như một mô hình sinh (generative model), nhưng chúng cũng có thể được sử dụng như một mô hình phân biệt (discriminative model). Để sử dụng RBM như một mô hình phân biệt, chúng ta có thể sử dụng các đặc trưng được học từ lớp ẩn làm đầu vào cho một mô hình phân biệt chuẩn.

Cấu trúc mạng RBM:



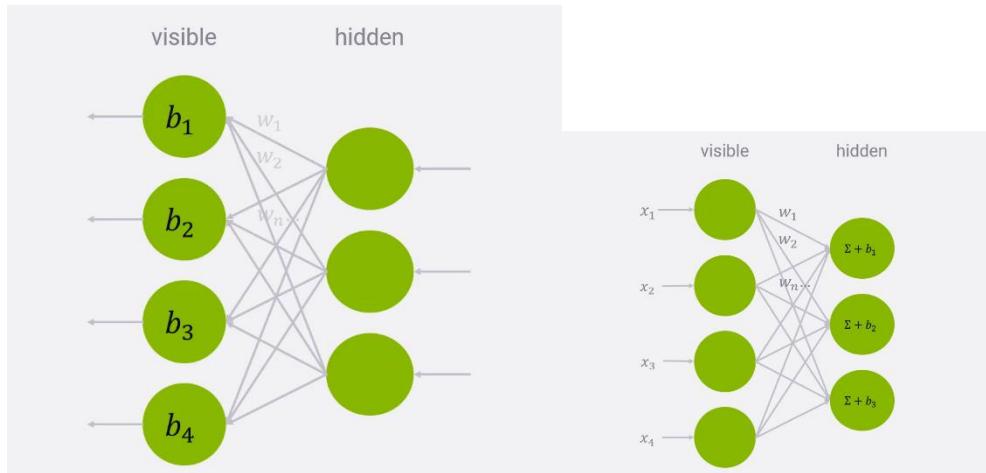
RBM gồm có 2 lớp: visible layer và hidden layer. Mô hình sẽ được train bằng cách đưa dữ liệu training vào visible layer, sau đó thực hiện train và tính toán weight, bias giữa chúng. Một activation function như ReLU sẽ được dùng để tạo ra output cho mỗi neuron ở hidden layer:



Ảnh: ReLU activation function

Từ cấu trúc mạng, ta có thể nói rằng mô hình này được gọi là Restricted Boltzmann Machine bởi vì giao tiếp giữa các neuron trong cùng một layer bị Restricted (hạn chế) bởi vì neuron trong layer này chỉ có thể nối tới các neuron của tầng khác mà thôi và không thể nối các neuron trong cùng một layer.

Quá trình train một RBM model sẽ diễn ra tương tự các mạng neural network thông thường: thực hiện forward pass và backward pass lặp đi lặp lại ở mỗi epoch để tìm ra weight và bias hợp lý, với điều kiện sao cho evaluate result là thấp nhất có thể.

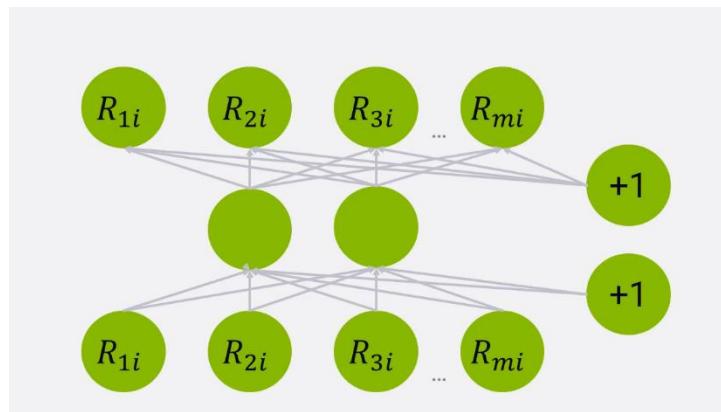


Ảnh: RBM backward pass

Ảnh: RBM forward pass

2.7.3. Autoencoder based recommender system

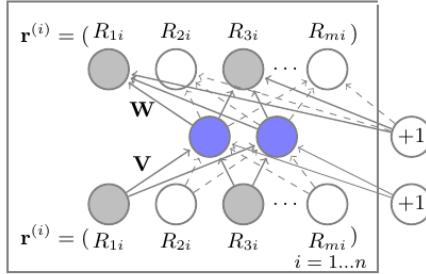
RBM được đề cập ở phần trước là một mô hình deep learning ra đời từ rất lâu trước đây, trước cả khi AI trở nên phổ biến. Trong khi đó, Deep Learning ngày nay đã phát triển một cách vượt bậc, vậy điều gì sẽ xảy ra nếu chúng ta áp dụng các kỹ thuật deep learning phức tạp hơn để giải quyết bài toán khuyến nghị? Autoencoder based chính là một phương pháp áp dụng những kỹ thuật đó.



Ảnh: Autoencoder for recommendations (autorec)

- Lịch sử ra đời

AutoRec (Autoregressive Neural Network for Collaborative Filtering) là một trong những mô hình tiên phong trong việc ứng dụng mạng nơ-ron Autoencoder vào hệ thống gợi ý. Mô hình này được giới thiệu lần đầu tiên trong nghiên cứu "AutoRec: Autoencoders Meet Collaborative Filtering" bởi nhóm tác giả Suvash Sedhain , Aditya Krishna Menon , Scott Sanner, Lexing Xie của Australian National University vào năm 2015. Nghiên cứu này đánh dấu một bước chuyển đổi quan trọng trong lĩnh vực gợi ý, khi các kỹ thuật học sâu (deep learning) bắt đầu thay thế các phương pháp truyền thống như Matrix Factorization.



Ảnh: Autorec model trong bài nghiên cứu AutoRec: Autoencoders Meet Collaborative Filtering

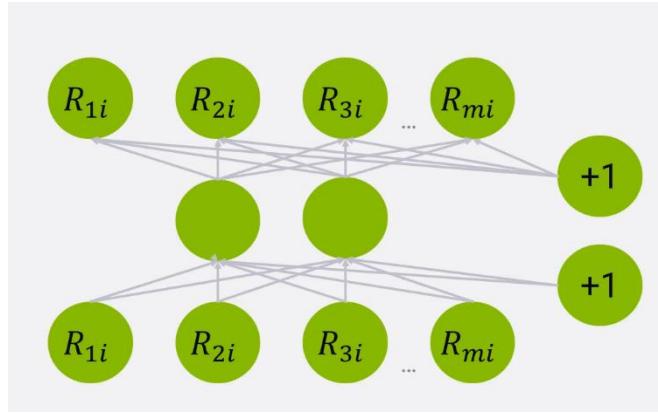
Trước khi AutoRec xuất hiện, các phương pháp như Matrix Factorization được sử dụng phổ biến trong hệ thống gợi ý nhờ khả năng giảm số chiều và dự đoán các giá trị bị thiếu trong ma trận đánh giá. Tuy nhiên, Matrix Factorization có một số hạn chế, đặc biệt là ở việc xử lý các mối quan hệ phi tuyến giữa người dùng và sản phẩm.

AutoRec được đề xuất như một giải pháp để khắc phục những hạn chế này, bằng cách sử dụng mạng nơ-ron Autoencoder để học biểu diễn phi tuyến từ dữ liệu. Mô hình này không chỉ giữ lại những lợi ích của Matrix Factorization mà còn tận dụng được sức mạnh của mạng nơ-ron để khai thác các đặc trưng phức tạp hơn trong dữ liệu.

- Ý tưởng

Autoencoder-based Recommender System là một ứng dụng tiên tiến của học sâu (deep learning) trong lĩnh vực hệ thống gợi ý. Autoencoder là một loại mạng nơ-ron học không giám sát, được thiết kế để tái tạo đầu vào của nó ở đầu ra bằng cách học một biểu diễn nén (compressed representation) trong một không gian ẩn (latent space). Đặc điểm này của Autoencoder rất phù hợp để giải quyết các bài toán liên quan đến gợi ý, nơi mà dữ liệu thường có kích thước lớn và thưa thớt (sparse).

Autoencoder vượt trội hơn so với các phương pháp truyền thống, như Matrix Factorization, nhờ khả năng xử lý dữ liệu phi tuyến và học các đặc trưng phức tạp. Ngoài ra, Autoencoder cũng có thể mở rộng để tích hợp thêm các loại dữ liệu khác nhau, chẳng hạn như đặc điểm người dùng, mô tả sản phẩm, hoặc các thông tin ngữ cảnh.



Ảnh: cấu trúc một autorec model

Autorec model sẽ có 3 layer:

- + Input layer ở phía dưới cùng chứa rating. Mỗi phần tử trong đầu vào đại diện cho đánh giá của người dùng về một sản phẩm (hoặc ngược lại, tùy thuộc vào cách sử dụng). Các giá trị không xác định (missing ratings) thường được điền bằng giá trị mặc định hoặc giữ nguyên là null.
- + Hidden layer ở giữa chứa một số lượng neurons nhỏ hơn số lượng đầu vào, nhằm mục đích nén dữ liệu (dimensionality reduction). Trọng số (weights) và bias kết nối giữa input layer và hidden layer được tối ưu hóa trong quá trình huấn luyện.
- + Output layer ở trên cùng đưa ra dự đoán

Autorec cũng sẽ sử dụng một hàm mất mát (loss function) để tính toán sai số giữa các giá trị đầu ra dự đoán và các giá trị thực tế tại những vị trí mà dữ liệu đã biết. Hàm mất mát sử dụng cho Autorec thường là Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

2.7.4. RNN for Clickstream recommender system

Recurrent Neural Network (RNN) là một loại mạng nơ-ron nhân tạo đặc biệt được thiết kế để xử lý dữ liệu tuần tự, nơi các điểm dữ liệu có mối liên kết thời gian hoặc trình

tự logic. Trong recommender system, RNN cũng được áp dụng để trở thành một công cụ mạnh mẽ để khai thác thông tin từ các chuỗi hành vi của người dùng, chẳng hạn như lịch sử xem phim, nhấp chuột, mua sắm, hoặc bất kỳ hành động nào có tính thời gian. Loại khuyến nghị dựa trên dữ liệu thời gian thực trong phiên người dùng này được gọi là Session-based Recommendation.

- Lịch sử ra đời

Mạng thần kinh hồi quy (recurrent neural network, viết tắt RNN) là một trong những đột phá lớn trong lĩnh vực trí tuệ nhân tạo, được thiết kế để xử lý dữ liệu tuần tự và học các mối quan hệ giữa các bước trong chuỗi. RNN được phát triển dựa trên công trình của David Rumelhart vào năm 1986. Một loại mạng RNN đặc biệt tên Hopfield netowrks được được John Hopfield phát hiện lại vào năm 1982, nhưng phải đến khi cơ chế lan truyền ngược qua thời gian (Backpropagation Through Time - BPTT) được phát triển, RNN mới thực sự trở nên khả thi để huấn luyện trên các tập dữ liệu lớn.

Trong những năm đầu, các hệ thống gợi ý chủ yếu dựa vào phương pháp lọc cộng tác và gợi ý dựa trên nội dung, như đã đề cập trước đó. Tuy nhiên, những phương pháp này thường gặp khó khăn trong việc xử lý các hành vi người dùng có tính tuần tự, chẳng hạn như các chuỗi nhấp chuột (clickstream) hoặc các phiên tương tác ngắn hạn.

Khoảng đầu thập niên 2010, với sự phát triển của sức mạnh tính toán và các kỹ thuật tối ưu hóa, RNN bắt đầu được áp dụng vào các bài toán gợi ý, đặc biệt là các hệ thống gợi ý dựa trên phiên làm việc (session-based recommender systems). Công trình tiên phong trong lĩnh vực này bao gồm nghiên cứu **Session-based Recommendations with Recurrent Neural Networks** của Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas và Domonkos Tikk (2015), trong đó giới thiệu cách sử dụng RNN để mô hình hóa các chuỗi hành động trong một phiên làm việc.

- Ý tưởng

Trong bối cảnh các ứng dụng cần tích hợp hệ thống khuyến nghị, không phải người dùng nào cũng muốn đăng nhập vào hệ thống ở lần đầu tiên họ sử dụng ứng dụng. Vì vậy

không thể nào xây dựng một khuyến nghị phù hợp khi chưa có dữ liệu người dùng trong hệ thống. Session-based recommendation ra đời để thu thập dữ liệu trong một phiên hoạt động (session) của người dùng

Bộ dataset đưa vào một RNN recommender system sẽ có cấu trúc như sau:

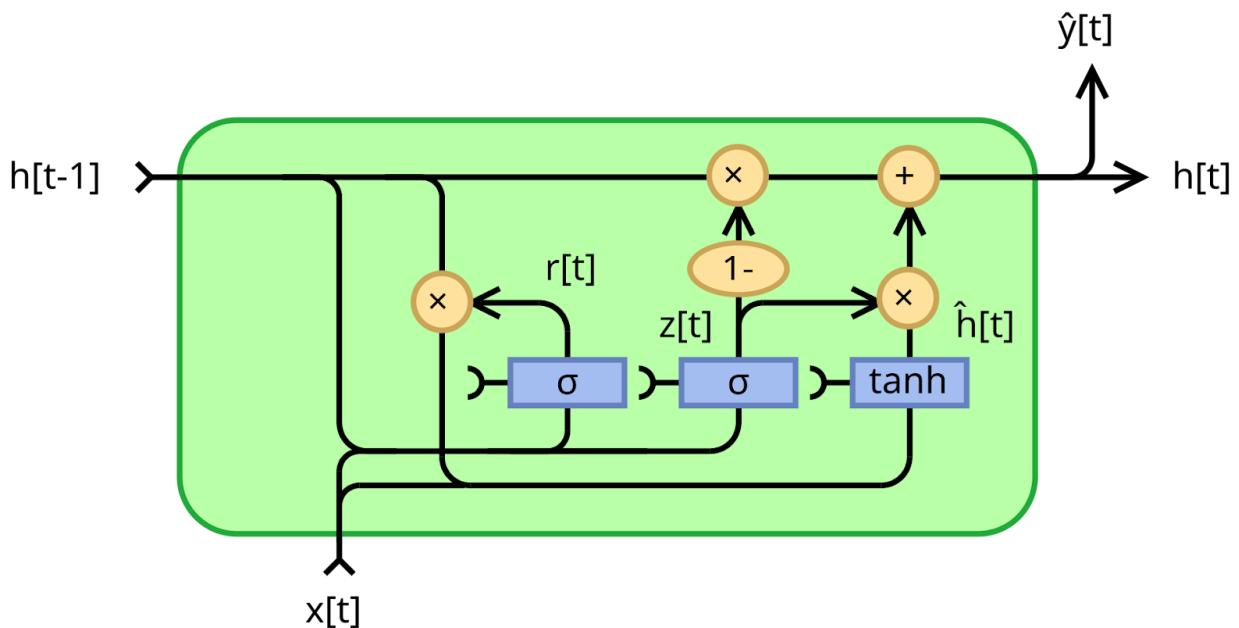
Event: Mô tả loại tương tác giữa người dùng và sản phẩm: ví dụ như view, click, scroll,...

VisitorID: Mã định danh duy nhất cho từng người dùng.

ItemID: Mã định danh duy nhất của sản phẩm mà người dùng tương tác. Để tương thích với PyTorch và tính toán hiệu quả, các mã này được mã hóa bằng công cụ LabelEncoder của Sklearn.

Item_type: Một danh sách gồm các loại như quần áo, giày dép, phụ kiện, túi xách, trang sức.

Trong nghiên cứu **Session-based Recommendations with Recurrent Neural Networks**, phần lớn nội dung của bài báo tập trung vào cách điều chỉnh RNN để hoạt động hiệu quả với dữ liệu clickstream theo phiên (session-based). RNN là một cấu trúc khá phức tạp, thay vì chỉ sử dụng các neuron đơn giản, chúng dựa vào các cấu trúc phức tạp hơn như LSTM (Long short-term memory), hoặc trong trường hợp này là các đơn vị hồi tiếp có công GRU (gated recurrent units).



Ảnh: Gated recurrent units (GRUs)

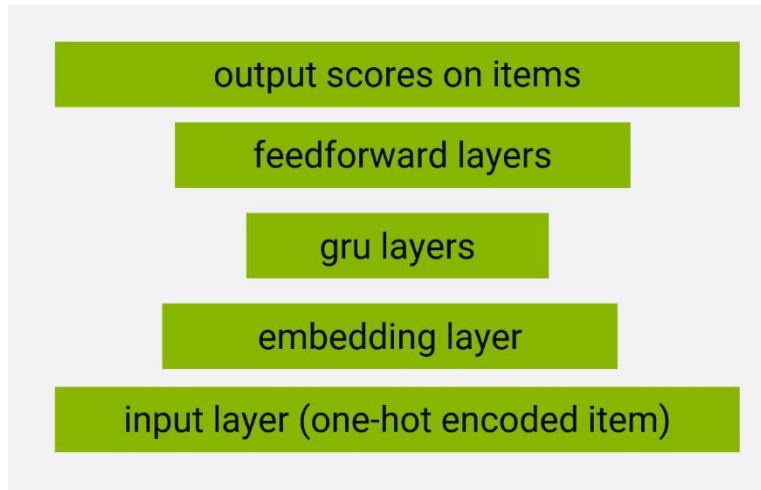
LSTM (Long Short-Term Memory): Một loại RNN nâng cao, có khả năng ghi nhớ lâu dài và kiểm soát thông tin nào cần ghi nhớ hoặc quên.

GRU (Gated Recurrent Unit): Cũng giống LSTM, nhưng cấu trúc đơn giản hơn và ít tham số hơn.

Trong nghiên cứu này, tác giả sử dụng **GRU** thay vì LSTM vì nó vừa đảm bảo hiệu suất cao, vừa giảm độ phức tạp tính toán. GRU có cơ chế cổng (gate), giúp nó tự động học cách cập nhật hoặc bỏ qua thông tin trong chuỗi dữ liệu, phù hợp với việc phân tích hành vi clickstream.

Phương pháp được đề cập trong nghiên cứu được tùy chỉnh để giải quyết bài toán gợi ý theo session, nên nó còn thường được gọi là GRU4Rec.

2.7.5. GRU4Rec



Kiến trúc của một GRU4Rec sẽ giống như trên, mô tả quá trình xử lý của một event trong click stream:

1. Dữ liệu đầu vào tại input layer là sản phẩm vừa được người dùng xem. Sản phẩm này được mã hóa dưới dạng **1-of-N encoding** (một cách biểu diễn dữ liệu mà mỗi sản phẩm là một vector với đúng một vị trí là 1, còn lại là 0).

2. **Embedding Layer:** Dữ liệu mã hóa này được đưa vào **embedding layer**, một lớp giúp chuyển đổi vector đầu vào thành một không gian đặc trưng có ý nghĩa hơn, thường có kích thước nhỏ gọn hơn.
3. **GRU Layers:** Sau đó, dữ liệu được đưa vào nhiều lớp GRU liên tiếp. Để đơn giản, ở đây chỉ hiển thị như một khối duy nhất, nhưng thực tế có thể có nhiều lớp GRU nằm ở giữa để xử lý thông tin tuần tự.
4. **Feed-Forward Layers:** Tiếp theo là các lớp **feed-forward** (các lớp nơ-ron truyền thống). Những lớp này không sử dụng cơ chế phức tạp như GRU, mà chỉ là các tầng nơ-ron thông thường để tinh chỉnh và xử lý thêm dữ liệu.
5. Cuối cùng, hệ thống tính toán **điểm số** (scores) cho tất cả các sản phẩm, từ đó chọn ra các sản phẩm mà mạng nơ-ron dự đoán có khả năng cao sẽ được người dùng xem tiếp theo trong chuỗi clickstream.

2.7.6. Ưu, nhược điểm của Deep Learning

- Ưu điểm:
 - Khi cần phải giải quyết các bài toán liên quan đến chuỗi sự kiện, deep neural network là lựa chọn tốt.
 - Với sự phát triển của các Deep Learning Framework, ngày nay ta có thể sử dụng hybrid nhiều model Deep Learning chung với nhau để giải quyết vấn đề khuyến nghị.
 - Deep learning có khả năng tự động học các đặc trưng phức tạp từ dữ liệu mà không cần sự can thiệp thủ công, đặc biệt hữu ích khi dữ liệu có sự phức tạp hoặc phi cấu trúc, chẳng hạn như trong các hệ thống gợi ý dựa trên hình ảnh hoặc văn bản.
- Bên cạnh đó Deep Learning khi ứng dụng vào recommender system cũng có nhiều nhược điểm:
 - Các mô hình deep learning thường đòi hỏi phần cứng mạnh mẽ (đặc biệt là GPU) và tài nguyên tính toán lớn, điều này có thể dẫn đến chi phí cao, đặc biệt là trên môi trường Production.

- Vấn đề Black Box: các mô hình Deep Learning thường bị coi là black box vì chúng ta không dễ dàng giải thích cách mà các mô hình đưa ra các quyết định.
- Vấn đề về dữ liệu: Deep Learning cần một lượng dữ liệu training lớn để có thể đạt được hiệu quả.
- Khó khăn trong việc điều chỉnh các tham số vì có rất nhiều tham số cần điều chỉnh.

2.7.7. Case Study

Trong tài liệu **Deep Learning for Recommender Systems: A Netflix Case Study** được đăng tải bởi các thành viên của Netflix, họ đã xác nhận rằng Netflix đã sử dụng deep learning để cải thiện hệ thống gợi ý của mình. Ban đầu, họ không thấy sự cải thiện đáng kể so với các phương pháp không sử dụng deep learning. Tuy nhiên, khi thêm nhiều đặc trưng khác nhau vào dữ liệu đầu vào, các mô hình deep learning đã bắt đầu thể hiện hiệu quả vượt trội.

2.8. Hybrid method

2.8.1. Giới thiệu

Trong các phần trước ta đã tìm hiểu về nhiều phương pháp khác nhau, thông thường mỗi phương pháp sẽ có những ưu nhược điểm nhất định nên sẽ phù hợp để giải quyết từng vấn đề cụ thể. Vậy sẽ ra sao nếu ta kết hợp chúng lại ?

Hybrid method trong recommender system là sự kết hợp của nhiều kỹ thuật hoặc phương pháp khác nhau nhằm tận dụng ưu điểm của từng phương pháp và giảm thiểu các hạn chế của chúng. Các phương pháp kết hợp này thường kết hợp nhiều phương pháp lại với nhau như collaborative filtering và content-based filtering, hoặc kết hợp nhiều kỹ thuật khác như phân tích matrix factorization, và thậm chí các mô hình học sâu (deep learning).

Bằng cách tích hợp nhiều phương pháp, các hệ thống kết hợp (hybrid) có thể giảm thiểu những điểm yếu của từng phương pháp riêng biệt, chẳng hạn như cold-start problem.

Các sự kết hợp phổ biến bao gồm collaborative filtering và content-based filtering. Có nhiều phương pháp được sử dụng để kết hợp các kỹ thuật này, chẳng hạn như phương pháp trọng số, chuyển đổi, kết hợp, hoặc phương pháp cấp độ meta.

Các hệ thống kết hợp thường cho ra các khuyến nghị chính xác hơn so với các hệ thống sử dụng một phương pháp duy nhất. Tuy nhiên, việc triển khai hiệu quả yêu cầu phải được tối ưu cẩn thận để cân bằng các thành phần và đảm bảo hiệu suất.

Đặc biệt, hybrid method cũng có thể hữu ích trong việc giải quyết cold start problem. Ý tưởng là trong giai đoạn đầu khi còn ít dữ liệu, việc sử dụng các mô hình Collaborative filtering sẽ rất khó khăn, do đó chúng ta có thể sẽ kết hợp Collaborative filtering với Content based filtering để sử dụng các đặc điểm của item từ ban đầu và gợi ý được cho user.

Ví dụ, nếu ứng dụng của chúng ta vừa triển khai và chưa có một lượng user đủ lớn, việc áp dụng Collaborative filtering gần như là không thể vì không có đủ dữ liệu đầu vào. Do đó ta có thể áp dụng Hybrid method để kết hợp Collaborative filtering với Content based, nhờ có content based mà ta có thể gợi ý cho user dựa theo đặc điểm sẵn có của item mà không cần phải đợi đủ dữ liệu user. Sau này khi hệ thống có nhiều người dùng hơn, ta có thể điều chỉnh lại trọng số của Collaborative filtering để Hybrid method nghiên về phương pháp này nhiều hơn.

2.8.2. Case Study

Netflix hiện tại đang sử dụng một Hybrid Recommender System, kết hợp rất nhiều phương pháp lại với nhau, trong đó có cả những phương pháp:

Học củng cố (Reinforcement learning): Dựa trên hành vi của người dùng, Netflix thay đổi nội dung trên màn hình theo thời gian thực. Do đó, hệ thống luôn ở trong trạng thái thay đổi liên tục và thay đổi tùy thuộc vào các tương tác của người dùng.

Mạng nơ-ron sâu (Deep neural networks): bao gồm cả RBM đã từng đề cập trong đề tài này

Phân tách ma trận (Matrix factorization): bao gồm cả SVD đã từng đề cập trong đề tài này

Và còn nhiều phương pháp khác được kết hợp lại.

2.9. Một số thư viện xây dựng Recommender System

- Surprise:

Surprise là một thư viện Python scikit dùng để xây dựng và đánh giá các hệ thống đề xuất, cung cấp một API đơn giản và dễ hiểu, giúp người mới bắt đầu cũng có thể sử dụng được. Được phát triển trên nền tảng SciPy, Surprise cung cấp một loạt các thuật toán như collaborative filtering, bao gồm các phương pháp dựa trên matrix factorization như Singular Value Decomposition (SVD) và Non-negative Matrix Factorization (NMF). Nó cũng hỗ trợ các phương pháp neighborhood-based như k-Nearest Neighbors (k-NN) và cung cấp các công cụ để đánh giá mô hình.

Trong khuôn khổ đề tài này sẽ sử dụng thư viện Surprise là chủ yếu để xây dựng thực nghiệm và các ứng dụng minh họa.

- Librecommender

LibRecommender là một thư viện khác bên cạnh surprise, dễ dùng hơn và có thể áp dụng vào production. Thư viện này bao gồm hai module gồm có module training (gọi là libreco) và module serving (gọi là lib-serving) để có thể nhanh chóng huấn luyện và triển khai các mô hình đề xuất khác nhau.

Các tính năng chính bao gồm:

- Triển khai một số thuật toán đề xuất phổ biến như FM, DIN, LightGCN,...
- Xây dựng hybrid recommender system, cho phép người dùng sử dụng cả collaborative-filtering hoặc content-based features.

- Tiết kiệm bộ nhớ, tự động chuyển đổi các tính năng phân loại và tính năng phân loại nhiều giá trị thành biểu diễn thưa thớt.
- Hỗ trợ huấn luyện cho cả dữ liệu explicit và implicit, cũng như lấy mẫu tiêu cực cho dữ liệu implicit.
- Cung cấp quy trình làm việc từ đầu đến cuối: xử lý dữ liệu/tiền xử lý -> huấn luyện mô hình -> đánh giá -> save/load -> serving.
- Hỗ trợ dự đoán và đề xuất cho vấn đề cold-start.
- Hỗ trợ đề xuất tính năng động và chuỗi.
- Cung cấp API thống nhất và dễ sử dụng cho tất cả các thuật toán.
- Dễ dàng huấn luyện lại mô hình với user-item mới từ dữ liệu mới.

- **Tensorflow**

TensorFlow là một thư viện mã nguồn mở mạnh mẽ được phát triển bởi Google, chủ yếu được sử dụng để xây dựng và triển khai các mô hình học máy (machine learning) và học sâu (deep learning). Được thiết kế để dễ dàng sử dụng và hiệu quả, TensorFlow hỗ trợ nhiều nền tảng và môi trường khác nhau, từ máy tính cá nhân cho đến các hệ thống phân tán phức tạp.

TensorFlow có các tính năng chính sau:

1. **Xây dựng mô hình học sâu:** hỗ trợ nhiều loại mô hình học sâu, bao gồm mạng nơ-ron tích chập (CNN), mạng nơ-ron hồi tiếp (RNN), mạng đối kháng (GANs), và nhiều loại mô hình khác, giúp giải quyết các bài toán phức tạp như nhận dạng ảnh, xử lý ngôn ngữ tự nhiên (NLP), và dự đoán chuỗi thời gian.
2. **Dễ dàng triển khai:** TensorFlow cho phép triển khai mô hình học sâu trên nhiều nền tảng khác nhau, từ các thiết bị di động, máy tính để bàn đến các máy chủ với hiệu suất cao. Nó hỗ trợ cả việc sử dụng GPU và TPU để tăng tốc độ tính toán.

3. **API linh hoạt và dễ sử dụng:** TensorFlow cung cấp nhiều API cho các cấp độ người dùng khác nhau. Đối với người mới bắt đầu, TensorFlow cung cấp API cấp cao (Keras) giúp xây dựng mô hình dễ dàng. Trong khi đó, các nhà nghiên cứu và phát triển chuyên sâu có thể sử dụng API cấp thấp để tạo ra các mô hình phức tạp và tối ưu hóa chúng.
4. **Tích hợp với các công cụ khác:** TensorFlow có khả năng tích hợp với nhiều công cụ và thư viện khác như TensorFlow Lite (dành cho các ứng dụng di động), TensorFlow.js (dành cho các ứng dụng web), và TensorFlow Extended (dành cho việc triển khai mô hình vào sản xuất).
5. **Cộng đồng phát triển lớn:** Là một dự án mã nguồn mở, TensorFlow có một cộng đồng phát triển rất lớn và sôi động, với hàng nghìn đóng góp từ các nhà phát triển và nhà nghiên cứu trên toàn thế giới. Điều này giúp người dùng dễ dàng tìm kiếm tài liệu, ví dụ và hỗ trợ kỹ thuật.

TensorFlow được ứng dụng rộng rãi trong nhiều lĩnh vực như y tế, tài chính, robot, xe tự lái và nhiều ngành công nghiệp khác, trở thành một trong những công cụ quan trọng nhất trong cộng đồng học máy hiện nay.

Trong lĩnh vực recommender system, Tensorflow cũng được ứng dụng để xây dựng các mạng neural network để thực hiện phương pháp Deep Learning for Recommender System.

- **TSRS (Tensor FLow Recommend System)**

Đây là một thư viện của google được build để implement recommend system dễ dàng bằng TensorFlow + Keras.

Cơ chế hoạt động gồm 2 stage chính:

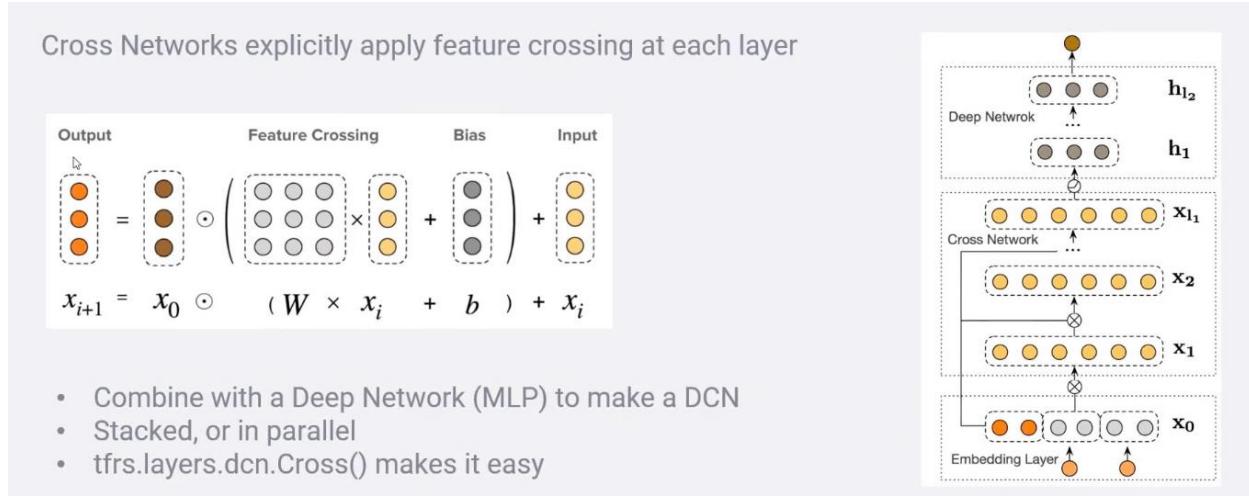
- **Retrival stage:** stage này sẽ thực hiện tương tự build ra embedding layer của user và movie (trong bài toán movie lens). Dựa vào đó và map chúng

vào các vector đa chiều để tính toán độ tương đồng giữa các user và giữa các movie => Kết quả của stage này là sẽ cho ra một list các candidate item (gọi là các recommend item tiềm năng) dựa vào độ tương đồng để dự đoán là một user nào đó đã xem qua movie này chưa.

- **Ranking stage:** stage này sẽ thực hiện lấy input là các candidate items từ Retrieval stage phía trên để đưa vào một model deep learning nhiều lớp để train => và dự đoán rating của từng cặp user - movie tương ứng (đến bước này ta mới sử dụng dữ kiện của rating score, còn retrieval stage không hề dùng đến mà chỉ sử dụng các đặc điểm của movie và user để tính độ tương đồng)

Các concept khác của TSRS

- Side features and Deep Retrieval: hiểu đơn giản là thay vì chỉ lấy các đặc trưng cơ bản của movie như là title, description ta sẽ lấy nhiều đặc trưng hơn để bỏ vào retrieval stage: như timestamp,... điều này giúp xử lý vấn đề thiếu hụt dữ liệu, cold start problem....
- Multi task recommendation: trong các web ecommerce thì ta có thể kết hợp nhiều loại behaviour của user lại để recommend, mà với mỗi loại như vậy ta cần dùng các model chuyên biệt,... việc này làm phức tạp hóa vấn đề mà ta có thể dùng "joint model" mà TSRF giới thiệu để kết hợp tất cả các dạng recommend này lại với nhau.
- Deep and cross Network:



Ví dụ: một user A nào đó mua táo, và sách dạy nấu ăn \Rightarrow ta recommend cho họ một cái máy ép táo (vì có thể đoán họ mua táo về để chế biến, mà chế biến thường cần máy ép) \Rightarrow cách recommend này có thể được recommend bằng cách áp dụng Cross Network. Chi tiết cách implement đã được Google viết trong doc của TSRF.

2.10. Scaling recommender system

Recommender System có thể được triển khai bằng rất nhiều kỹ thuật, thuật toán khác nhau. Tuy nhiên, việc mở rộng quy mô (scaling) các hệ thống gợi ý này là một thách thức lớn khi phải xử lý lượng dữ liệu khổng lồ từ hàng triệu người dùng và sản phẩm.

Các thách thức của việc xây dựng hệ khuyến nghị cho một hệ thống lớn có thể kể đến:

Khối lượng dữ liệu lớn: Với hàng tỷ lượt tương tác giữa người dùng và sản phẩm, việc lưu trữ và xử lý dữ liệu đòi hỏi hạ tầng mạnh mẽ và các phương pháp tối ưu hóa.

Độ phức tạp của thuật toán: Các thuật toán tiên tiến như matrix factorization, deep learning, hoặc graph-based recommendation cần nhiều tài nguyên tính toán.

Tốc độ và thời gian thực: Hệ thống phải cung cấp gợi ý nhanh chóng, đặc biệt với các ứng dụng yêu cầu phản hồi theo thời gian thực.

Cold-start problem: Xử lý dữ liệu cho người dùng hoặc sản phẩm mới mà chưa có lịch sử tương tác vẫn luôn là bài toán khó.

Hiện nay, có nhiều cách để scaling recommender system, đáp ứng được lượng dữ liệu không lồ:

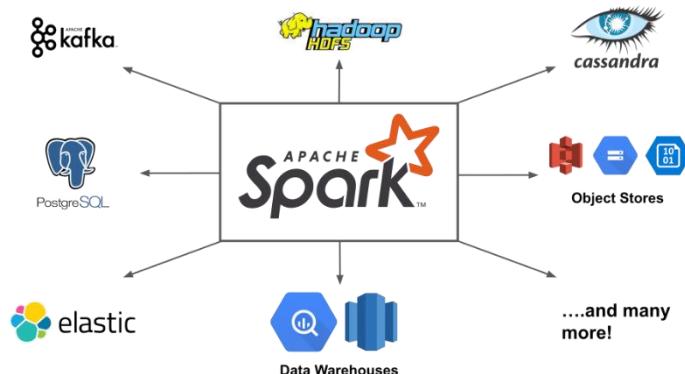
Phân tán và song song hóa: Sử dụng hạ tầng như Hadoop, Spark hoặc Kubernetes để xử lý dữ liệu phân tán và train mô hình song song.

Tối ưu hóa thuật toán: Áp dụng các phương pháp nén ma trận, sampling, hoặc giảm độ phức tạp của mô hình để giảm yêu cầu tính toán.

Triển khai hệ thống kết hợp (Hybrid): Kết hợp nhiều phương pháp như collaborative filtering và content-based filtering để tăng hiệu quả và độ linh hoạt.

Caching và Precomputing: Tính toán trước các gợi ý phổ biến hoặc lưu trữ các kết quả gợi ý tạm thời để tăng tốc độ phản hồi.

2.10.1. Giới thiệu về Apache Spark



Apache Spark là một framework xử lý dữ liệu phân tán mạnh mẽ, được thiết kế để xử lý khối lượng lớn dữ liệu với tốc độ cao. Apache Spark được phát triển

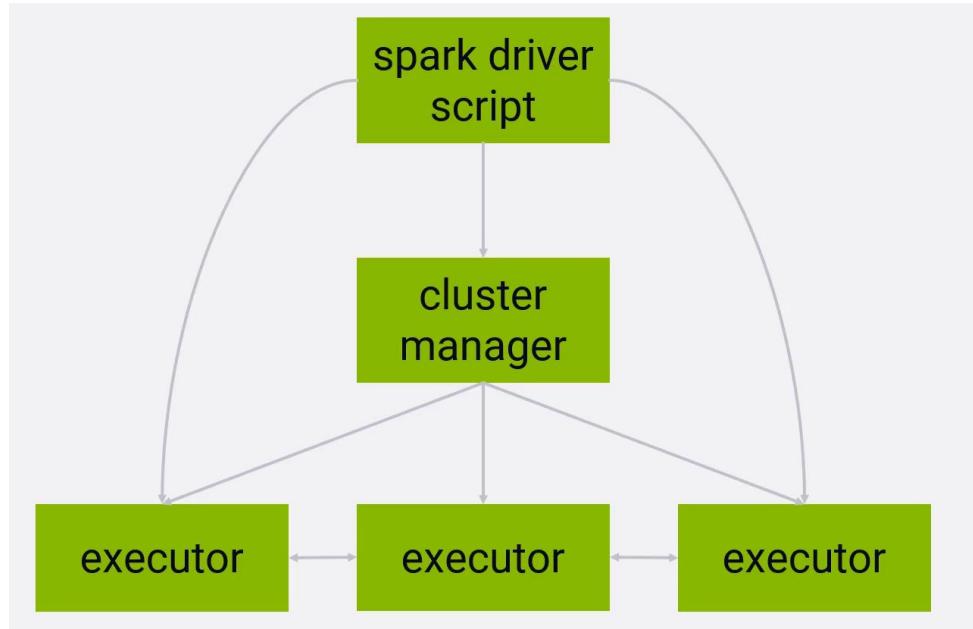
vào năm 2009 bởi AMPLab. Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay.

Spark hỗ trợ nhiều mô hình tính toán như xử lý theo lô (batch processing), xử lý theo luồng (streaming), truy vấn dữ liệu (SQL), và học máy (machine learning). Với khả năng phân tán và tối ưu hóa trên hạ tầng như Hadoop HDFS, Amazon S3, hoặc Kubernetes, Spark trở thành công cụ lý tưởng để mở rộng quy mô (scaling) các hệ thống gợi ý (Recommender Systems).

Ta sẽ sử dụng Spark để giải quyết vấn đề scaling của Recommender System vì các lý do sau:

- Spark có thể xử lý dữ liệu được lưu trữ phân tán trên nhiều node, giúp giải quyết vấn đề khói lượng lớn dữ liệu người dùng và sản phẩm. Nói cách khác, thay vì chỉ xử lý dữ liệu, training model cho hệ khuyến nghị trên một máy duy nhất, ta có thể dùng Spark để áp dụng nguyên lý “chia để trị”: phân tán dữ liệu và xử lý cùng một lúc trên nhiều máy khác nhau, giúp tăng hiệu suất đáng kể.
- Nhờ áp dụng “chia để trị” - phân tán xử lý dữ liệu trên nhiều máy khác nhau, mà ta sẽ dễ dàng scaling khi hệ thống lớn dần bằng cách chỉ cần thêm một máy mới vào khi cần thiết.
- Với engine xử lý trong bộ nhớ (in-memory), Spark nhanh hơn nhiều so với các framework truyền thống như Hadoop MapReduce.
- Không những thế, Thư viện **MLlib** của Spark tích hợp các thuật toán phổ biến như Alternating Least Squares (ALS) cho Collaborative Filtering, rất phù hợp để xây dựng hệ thống gợi ý. Thuật toán này sẽ được sử dụng để minh họa trong tài liệu này ở phần ...

2.10.2. Kiến trúc của Spark



*Ảnh: kiến trúc của Apache Spark

Đầu tiên, ta cần viết một **spark driver script**, có thể bằng Python, Scala, hoặc Java,... để định nghĩa cách mà ta xử lý dữ liệu sử dụng API của thư viện Spark cung cấp. Khi mà ta thực hiện chạy driver script này,

Spark driver script sẽ được chuyển tới **cluster manager**, tại đây nó thực hiện sẽ phân bổ tài nguyên sao cho hợp lý. **Cluster manager** sẽ phân bổ một loạt các executor processes trên từng executor để thực hiện công việc xử lý dữ liệu thực tế.

Nói một cách dễ hiểu, **Spark driver script** sẽ như một chò nhận chỉ dẫn của chúng ta, sau đó **cluster manager** sẽ quyết định xem là sẽ xử lý lượng dữ liệu không lồ đó như thế nào, chia như thế nào cho hợp lý. Dữ liệu không lồ sau đó sẽ được chia cho các **executor** xử lý song song.

Trong spark driver script, ta sẽ sử dụng các thư viện API do Spark cung cấp để có thể giao tiếp với nó:

2.10.3. Spark SQL

- **Chức năng chính:** Cung cấp công cụ để xử lý dữ liệu có cấu trúc bằng cách sử dụng SQL hoặc DataFrame API.
- **Các tính năng quan trọng:**
 - **DataFrame API:** Một giao diện tương tự như bảng trong cơ sở dữ liệu, cho phép thao tác dễ dàng với dữ liệu có cấu trúc.
 - **SQL Query:** Hỗ trợ viết các truy vấn SQL trực tiếp trên dữ liệu.
 - **Tương thích với Hive:** Có thể đọc và xử lý dữ liệu từ các bảng Hive.
 - **Catalyst Optimizer:** Tự động tối ưu hóa các truy vấn để tăng hiệu năng.
- **Spark MLlib (Machine Learning Library)**
- **Chức năng chính:** Cung cấp các công cụ để xây dựng và triển khai các mô hình học máy (Machine Learning).
- **Các tính năng quan trọng:**
 - **Thuật toán học máy:** Hỗ trợ các thuật toán phổ biến như phân cụm (clustering), phân loại (classification), và hồi quy (regression).
 - **Tích hợp Spark Core:** Sử dụng bộ nhớ trong và RDD để tăng tốc xử lý.
 - **Pipeline API:** Hỗ trợ xây dựng quy trình học máy từ tiền xử lý dữ liệu đến đánh giá mô hình.
 - **Hỗ trợ tính năng phân tán:** Các thuật toán được tối ưu để xử lý trên dữ liệu lớn phân tán.

2.10.4. Spark Streaming

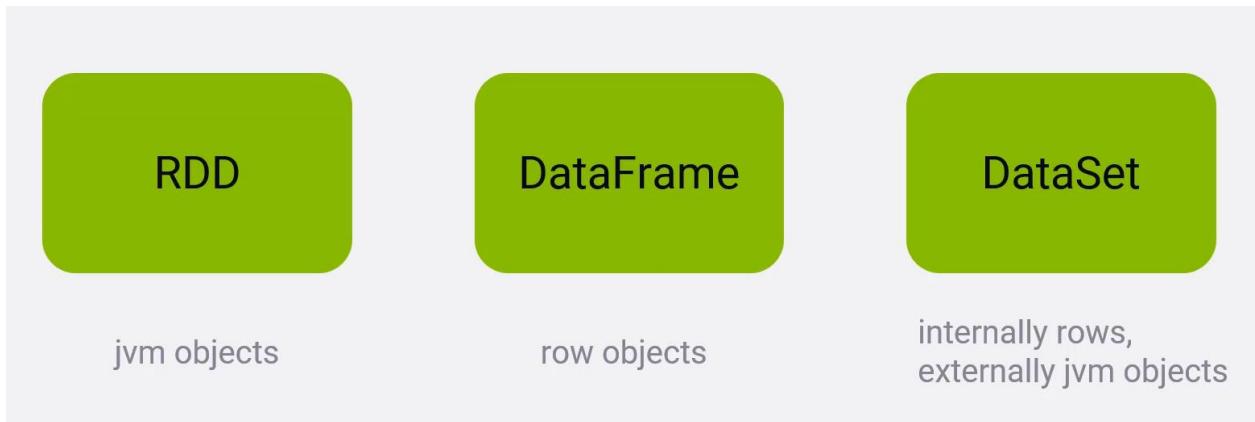
- **Chức năng chính:** Xử lý dữ liệu theo thời gian thực (real-time).
- **Các tính năng quan trọng:**
 - Mini-batch processing: Xử lý dữ liệu theo các lô nhỏ (micro-batches) với độ trễ thấp.

- Tích hợp với nguồn dữ liệu streaming: Hỗ trợ Kafka, Flume, Amazon Kinesis, và socket trực tiếp.
- Khả năng chịu lỗi: Sử dụng RDD để đảm bảo xử lý an toàn khi xảy ra lỗi.

2.10.5. GraphX

- **Chức năng chính:** Một thư viện để phân tích và xử lý dữ liệu đồ thị (graph data).
- **Các tính năng quan trọng:**
 - Graph abstraction: Cung cấp giao diện để xử lý các đỉnh (vertex) và cạnh (edge) của đồ thị.
 - Thuật toán đồ thị: Hỗ trợ các thuật toán như PageRank, Connected Components, và Triangle Count.
 - Tích hợp Spark Core: Kết hợp xử lý đồ thị và dữ liệu song song.

Apache Spark cung cấp ba cấu trúc dữ liệu chính: **RDD**, **DataFrame**, và **Dataset**. Mỗi loại có đặc điểm riêng và phù hợp với các nhu cầu xử lý dữ liệu khác nhau.



- RDD: ra đời sớm nhất, nhưng API phức tạp hơn so với DataFrame và Dataset, đồng thời hiệu suất thấp hơn vì không có tối ưu hóa tự động như DataFrame/Dataset.

- Dataframe: Là nâng cấp của RDD, DataFrame tổ chức dữ liệu thành dạng bảng (gồm hàng và cột), tương tự như bảng trong cơ sở dữ liệu. API của nó cũng dễ sử dụng hơn vì khá giống với SQL. Trong khuôn khổ đề tài này sẽ sử dụng Dataframe để triển khai xử lý dữ liệu ở phần ...
- Dataset được thiết kế để kết hợp ưu điểm của RDD và DataFrame. Nó cung cấp type-safety như RDD và tối ưu hóa tự động như DataFrame.

2.11. Case Study

2.11.1. Amazon

Amazon.com là một trong những đơn vị tiên phong trong việc ứng dụng hệ thống gợi ý trong thương mại điện tử, đóng vai trò quan trọng trong thành công của nền tảng này. Bắt đầu với mô hình bán sách trực tuyến, Amazon đã mở rộng danh mục sản phẩm, bao gồm sách, điện tử, đồ gia dụng và nhiều mặt hàng khác. Hệ thống gợi ý của Amazon ngày nay dựa trên dữ liệu từ hành vi mua sắm, duyệt web và đánh giá của người dùng để cung cấp các gợi ý sản phẩm cá nhân hóa, từ đó thúc đẩy doanh thu và tăng cường trải nghiệm người dùng.

- Các nguồn dữ liệu chính để xây dựng Recommender System:
 - **Explicit Ratings:** Người dùng đánh giá sản phẩm theo thang điểm 5 sao.
 - **Implicit Ratings:** Dữ liệu từ hành vi duyệt web và mua sắm của người dùng.
 - Dữ liệu lịch sử của người dùng
- Các thuật toán và mô hình:
 - **Item-based Collaborative Filtering:** phương pháp này tập trung vào mối quan hệ giữa các sản phẩm. Nếu một người dùng đã mua sản phẩm X, hệ thống sẽ gợi ý các sản phẩm khác tương tự như X mà các người dùng khác cũng đã mua.

- **Content-Based Filtering:** Phương pháp này có thể rất hiệu quả khi người dùng đã thể hiện sự quan tâm rõ ràng đến một loại sản phẩm cụ thể
- **Hybrid Models:** mô hình kết hợp giữa Item-based Collaborative Filtering và Content-Based Filtering giúp cải thiện độ chính xác của gợi ý, đồng thời giảm thiểu các vấn đề như **cold-start problem** và **thưa dữ liệu** (sparse data).
- **Matrix Factorization:** Phương pháp này đặc biệt hữu ích trong việc xử lý những ma trận lớn, nơi mỗi hàng tương ứng với một người dùng và mỗi cột là một sản phẩm.
- **Deep Learning:** Với học sâu, Amazon có thể xử lý khối lượng lớn dữ liệu người dùng và sản phẩm, bao gồm cả hình ảnh, văn bản và hành vi người dùng, để tạo ra các gợi ý chính xác và linh hoạt hơn. Hệ thống học sâu có thể tự động học từ dữ liệu lớn mà không cần phải lập trình cụ thể cho mỗi tình huống.
 - **Ảnh hưởng và thách thức:** Hệ thống gợi ý của Amazon giúp tăng sự tham gia và doanh thu, với các gợi ý sản phẩm chiếm tỷ lệ lớn trong doanh thu của Amazon. Tuy nhiên, hệ thống này cũng gặp phải các thách thức như vấn đề **cold-start** đối với người dùng mới hoặc sản phẩm mới và **data sparsity** trong một số danh mục sản phẩm.

2.11.2. Netflix

Netflix, một trong những nền tảng phát trực tuyến lớn nhất thế giới hiện nay, bắt đầu là một công ty cho thuê DVD qua thư. Dần dần, Netflix đã chuyển đổi thành dịch vụ phát trực tuyến, cung cấp một kho phim và chương trình truyền hình phong phú cho người dùng trên toàn cầu. Netflix không chỉ cho phép người dùng xem các bộ phim và chương trình truyền hình yêu thích mà còn cung cấp khả năng đánh giá các bộ phim và chương trình này, giúp xây dựng hệ thống gợi ý phim mạnh mẽ.

- Các nguồn dữ liệu chính để xây dựng Recommender System:
 - **Explicit Ratings:** Người dùng có thể đánh giá các bộ phim và chương trình truyền hình theo thang điểm 5 sao.
 - **Implicit Ratings:** Các hành động như việc người dùng xem phim nào, thời gian xem, hoặc việc bỏ qua một bộ phim.
- **Netflix Prize:** Một trong những đóng góp quan trọng của Netflix đối với cộng đồng nghiên cứu là cuộc thi Netflix Prize. Cuộc thi này nhằm mục đích cải thiện hệ thống gợi ý của Netflix, gọi là Cinematch, bằng cách tìm ra những mô hình hợp tác lọc (collaborative filtering) tốt hơn. Chi tiết bộ dữ liệu:
 - **Bộ dữ liệu huấn luyện:** Bao gồm hơn 100 triệu lượt đánh giá của 480,189 người dùng đối với 17,770 bộ phim.
 - **Bộ dữ liệu thử nghiệm:** Dữ liệu này chứa các bộ triplet (User, Movie, GradeDate) nhưng không bao gồm điểm đánh giá thực tế. Người tham gia cuộc thi phải dự đoán các đánh giá dựa trên các mô hình của bộ dữ liệu huấn luyện.Nhiều mô hình như **Latent Factor Models** và **SVD++** đã trở thành những phương pháp phổ biến trong nghiên cứu và ứng dụng hệ thống gợi ý sau cuộc thi.
- Các thuật toán và mô hình:
 - **Collaborative Filtering:** Đây là phương pháp chính mà Netflix sử dụng. Hệ thống tìm kiếm các mẫu hành vi giữa người dùng để đưa ra các gợi ý. Phương pháp này có thể là user-based filtering hoặc item-based filtering.
 - **Deep Learning:** Các mô hình này giúp hệ thống hiểu được các đặc điểm phức tạp của hành vi người dùng và phim mà có thể không dễ dàng nhận thấy bằng các mô hình truyền thống.

- **Hybrid Models:** Kết hợp nhiều mô hình khác nhau để tối ưu hóa kết quả gợi ý, như kết hợp lọc cộng tác và lọc nội dung, nhằm xử lý các vấn đề như **cold-start problem** khi có ít dữ liệu về người dùng hoặc bộ phim.
- **Ảnh hưởng:** Hệ thống gợi ý của Netflix không chỉ giúp tăng trải nghiệm người dùng mà còn đóng góp lớn vào việc phát triển các thuật toán gợi ý. Những cải tiến trong hệ thống của Netflix đã góp phần thúc đẩy sự phát triển của công nghệ gợi ý toàn cầu và thúc đẩy sự chuyển mình của các nền tảng trực tuyến khác.

2.12. Các vấn đề khi xây dựng recommender system

2.12.1. Data Sparsity

Data Sparsity (Thưa thớt dữ liệu) là một vấn đề phổ biến trong các hệ thống gợi ý, xảy ra khi hệ thống có một lượng lớn người dùng và sản phẩm, nhưng số lượng tương tác hoặc đánh giá thực tế lại rất ít. Tình trạng này thường dẫn đến dữ liệu đầu vào của hệ thống bị thưa thớt, gây khó khăn trong việc xây dựng các mô hình gợi ý hiệu quả.

Giải pháp:

- Sử dụng Matrix Factorization (Phân tích ma trận). Phương pháp này phân rã ma trận đánh giá lớn thành hai ma trận nhỏ hơn, trong đó mỗi ma trận đại diện cho một không gian tiềm ẩn (latent space) của người dùng và sản phẩm. Nhược điểm của nó là hiệu quả giảm khi dữ liệu quá thưa thớt.
- Sử dụng Hybrid Systems: Kết hợp Collaborative Filtering và Content-Based Filtering để giảm tác động của độ thưa thớt dữ liệu. Hiệu quả hơn Matrix Factorization khi dữ liệu quá thưa thớt nhưng yêu cầu thiết kế hệ thống phức tạp hơn.

2.12.2. Cold-Start Problem

Một trong những thách thức lớn nhất của Recommender System là vấn đề **Cold-Start**, khi hệ thống không có đủ dữ liệu ban đầu để hoạt động hiệu quả. Đây là tình huống phổ biến ở các giai đoạn đầu triển khai hệ thống hoặc khi người dùng và sản phẩm mới xuất hiện trong hệ thống, ảnh hưởng trực tiếp đến khả năng gợi ý, khiến hệ thống mất đi hiệu quả trong việc cung cấp các đề xuất chính xác và hữu ích.

Cold-Start Problem có thể được chia thành ba trường hợp chính:

- **Người dùng mới (Cold-Start User):** Khi một người dùng mới tham gia, hệ thống chưa có thông tin về hành vi, sở thích hay đánh giá của người dùng này. Hệ thống không thể gợi ý sản phẩm phù hợp.
- **Sản phẩm mới (Cold-Start Item):** Khi một sản phẩm mới được thêm vào, do chưa có tương tác nào giữa sản phẩm và người dùng, hệ thống không thể xác định nhóm đối tượng quan tâm đến sản phẩm này.
- **Hệ thống mới (Cold-Start System):** Khi một hệ thống mới được triển khai, toàn bộ dữ liệu đều ở trạng thái ban đầu. Các thuật toán cần thời gian để thu thập đủ dữ liệu tương tác và tối ưu hóa kết quả gợi ý.

Giải pháp: Sử dụng Hybrid Systems nhằm kết hợp nhiều phương pháp gợi ý khác nhau để tận dụng ưu điểm của từng phương pháp và giảm thiểu hạn chế.

Ví dụ: Kết hợp Collaborative Filtering và Content-Based Filtering: Gợi ý dựa trên lịch sử tương tác (CF) nhưng có thể bổ sung bằng thông tin thuộc tính sản phẩm (CBF) khi dữ liệu tương tác không đủ.

2.12.3. Privacy

Privacy (bảo mật thông tin cá nhân) là một trong những vấn đề quan trọng trong hệ thống gợi ý, đặc biệt trong bối cảnh thu thập và xử lý dữ liệu người dùng ngày càng gia tăng. Hệ thống gợi ý sử dụng thông tin như lịch sử đánh giá, sở thích, và hành vi người dùng để cá nhân hóa trải nghiệm. Tuy nhiên, điều này có thể làm lộ thông tin nhạy cảm của người dùng như quan điểm chính trị, hoặc các thói quen cá nhân, dẫn đến rủi ro về bảo mật và vi phạm quyền riêng tư.

Giải pháp:

- **Privacy at Data Collection Time:** Dữ liệu được thu thập dưới dạng tổng hợp hoặc nhiễu hóa thay vì lưu giữ các thông tin cá nhân cụ thể. Nhược điểm là nó sẽ làm tăng độ phức tạp vì cần các kỹ thuật khai thác dữ liệu chuyên biệt. Các kỹ thuật phổ biến gồm:
 - **Distributed Protocols:** Thu thập dữ liệu từ người dùng qua các giao thức phân tán, đảm bảo rằng không một cá nhân hoặc tổ chức nào có quyền truy cập vào toàn bộ dữ liệu.
 - **Perturbation Techniques:** Thêm nhiễu vào dữ liệu người dùng, khiến dữ liệu gốc khó bị truy vết.
 - **Aggregate Data Collection:** Thu thập dữ liệu ở dạng tổng hợp, không lưu trữ các giá trị cá nhân.
- **Privacy at Data Publication Time:** Dữ liệu được xử lý để ẩn danh trước khi công bố, thường bằng cách sử dụng các kỹ thuật nhóm hóa và làm mờ (**Group-based Anonymization**). Nhược điểm đó là hiệu quả không tuyệt đối do một số thông tin nhạy cảm vẫn có thể bị suy luận qua dữ liệu công khai. Các kỹ thuật phổ biến gồm:
 - **k-Anonymity:** Tất cả các bản ghi thuộc cùng một nhóm phải trông giống nhau ít nhất ở một mức độ nào đó, đảm bảo rằng một cá nhân không thể bị phân biệt.

- **I-Diversity:** Đảm bảo rằng trong mỗi nhóm, các giá trị nhạy cảm có sự đa dạng nhất định, tránh việc suy luận trực tiếp.
- **t-Closeness:** Đảm bảo phân phối các giá trị nhạy cảm trong mỗi nhóm gần giống với phân phối tổng thể, ngăn chặn các suy đoán sai lệch.

2.13. Các phương pháp khuyến nghị hay nhưng chưa phổ biến

2.13.1. DeepFM

Ý tưởng đơn giản của DeepFm là nó combine 2 method: Matrix Factorizaion và Deep learning để xây dựng recommender system.

Trong đó:

- Deep Learning mạnh về recommend high order feature
- Matrix factorization mạnh về recommend low order feature

Vậy high order và low order feature là gì?

Low-order features: Là những đặc trưng đơn giản, trực tiếp và thường xuất hiện trong dữ liệu. Chúng là mối quan hệ giữa các thuộc tính cơ bản mà không cần qua xử lý phức tạp. Ví dụ:

- Tuổi của người dùng.
- Giá của sản phẩm.
- Số lần một sản phẩm được mua.

High-order features: Là các đặc trưng phức tạp hơn, được sinh ra từ việc kết hợp hoặc tương tác giữa nhiều đặc trưng low-order. Những đặc trưng này thường được học thông qua các mô hình học sâu (Deep Learning) để tìm ra mối quan hệ tiềm ẩn mà con người khó thấy được. Ví dụ:

- Mỗi quan hệ giữa sở thích của người dùng và danh mục sản phẩm.
- Xu hướng người dùng mua sản phẩm giảm giá vào cuối tháng.
- Mối tương quan giữa độ tuổi và loại sản phẩm người dùng hay mua.

Cụ thể thì:

Low-order features:

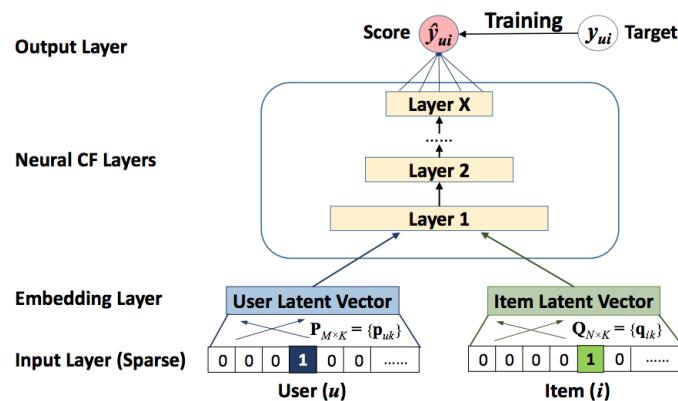
- Người dùng A thích xem sản phẩm thuộc danh mục "Điện thoại".
- Sản phẩm X có giá 10 triệu VND.

High-order features (tương tác phức tạp giữa các đặc trưng):

- Người dùng A có xu hướng mua sản phẩm "Điện thoại giá tầm 10-15 triệu" khi có khuyến mãi lớn.
- Người dùng A thường mua điện thoại của hãng Y sau khi đã xem các bài đánh giá trên mạng xã hội.

Ý tưởng và phương pháp triển khai đã được trình bày trong bài báo **DeepFM: A Factorization-Machine based Neural Network for CTR Prediction** bởi Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li và Xiuqiang He.

2.13.2. NCF



Neural Collaborative Filtering (NCF) là một phương pháp hiện đại trong Recommender Systems, sử dụng các mô hình học sâu (Deep Learning) để thay thế các thuật toán truyền thống như Matrix Factorization (MF) trong việc học các mối quan hệ giữa người dùng và sản phẩm từ ma trận tương tác (user-item interaction matrix).

Vậy nó khác gì DeepFM giới thiệu ở trên:

NCF: Tập trung vào tương tác user-item, mạnh về việc học high-order interactions giữa user và item.

DeepFM: Phù hợp với các bài toán phức tạp hơn, học cả low-order và high-order interactions từ dữ liệu nhiều đặc trưng.

3. NGHIÊN CỨU THỰC NGHIỆM

3.1. Thực nghiệm các phương pháp khuyến nghị

Trong phần này sẽ thực hiện thực nghiệm các phương pháp khuyến nghị đã trình bày như Content based filtering, KNN, Matrix Factorization, Deep learning, Hybrid để đánh giá và so sánh mức độ hiệu quả của các kỹ thuật.

Thông tin dataset thực nghiệm:

- Tên dataset: Amazon reviews 2013
- Mô tả: Bộ dữ liệu này bao gồm các đánh giá từ Amazon. Dữ liệu bao quát trong khoảng thời gian 18 năm, bao gồm khoảng ~35 triệu đánh giá tính đến tháng 3 năm 2013. Các đánh giá bao gồm thông tin về sản phẩm và người dùng, xếp hạng, và nội dung đánh giá dạng văn bản.

Do hạn chế về cơ sở hạ tầng nên ta chỉ sử dụng một phần dữ liệu trong tập dataset này: đó là tập **Beauty product reviews** (các đánh giá người dùng cho ngành hàng Mỹ phẩm - làm đẹp của Amazon). Tập đánh giá này có tổng cộng 252,056 reviews và nặng 46M cho file nén.

Ngoài ra ta cũng sẽ sử dụng tập categories.txt.gz chứa thông tin các category của tất cả sản phẩm trong Amazon. Tập này có tổng cộng 4872535 records.

Cấu trúc mẫu của tập **Beauty product reviews** như sau:

product/productId: B00006HAXW

product/title: Rock Rhythm & Doo Wop: Greatest Early Rock

product/price: unknown

review/userId: A1RSDE90N6RSZF

review/profileName: Joseph M. Kotow

review/helpfulness: 9/9

review/score: 5.0

review/time: 1042502400

review/summary: Pittsburgh - Home of the OLDIES

review/text: I have all of the doo wop DVD's and this one is as good or better than the 1st ones. Remember once these performers are gone, we'll never get to see them again. Rhino did an excellent job and if you like or love doo wop and Rock n Roll you'll LOVE this DVD !!

Tập dữ liệu gốc do Amazon gốc cung cấp là một file txt với cấu trúc có dạng: <đối tượng>/<cột>: <giá trị>. Ví dụ product/productId: B00006HAXW có nghĩa đây là dữ liệu của đối tượng product, cột productId, giá trị là B00006HAXW.

Để dễ dàng xử lý và đưa vào sử dụng trong các thư viện recommender như Surprise, ta cần chuyển dữ liệu dạng này thành file csv:

```

1 import csv
2
3 def parse_category_line(category_line):
4     """Phân tích một dòng category thành các phần theo cây phân cấp."""
5     return category_line.strip().split(", ")
6
7
8 def txt_to_csv(input_txt_file, output_csv_file):
9     """
10     Chuyển đổi dữ liệu từ file TXT sang CSV.
11     """
12     # Tạo danh sách lưu trữ các hàng dữ liệu CSV
13     rows = []
14     []
15     # Đọc file TXT và xử lý từng dòng
16     with open(input_txt_file, "r") as f:
17         current_product_id = None
18         for line in f:
19             if not line.startswith(" "): # Nếu là dòng productId
20                 current_product_id = line.strip()
21             else: # Nếu là dòng category
22                 category_parts = parse_category_line(line)
23                 # Tạo một hàng dữ liệu mới với các cấp độ category
24                 row = [current_product_id] + category_parts
25                 rows.append(row)
26
27     # Xác định số cấp độ tối đa của category
28     max_levels = max(len(row) for row in rows) - 1
29
30     # Tạo tiêu đề cho file CSV
31     headers = ["productId"] + [f"CategoryLevel{i+1}" for i in range(max_levels)]
32
33     # Ghi dữ liệu vào file CSV
34     with open(output_csv_file, "w", newline="", encoding="utf-8") as csv_file:
35         writer = csv.writer(csv_file)
36         writer.writerow(headers) # Ghi tiêu đề
37         for row in rows:
38             # Điện thêm cột trống nếu hàng không đủ cấp độ
39             row += [""] * (max_levels - (len(row) - 1))
40         writer.writerow(row)
41
42     print(f"Dữ liệu đã được chuyển đổi và lưu vào {output_csv_file}.")

```

Kết quả ta thu được file csv với các cột:

- productId: mã sản phẩm - string
- title: tiêu đề sản phẩm - string
- price: giá tiền - float
- userId: mã sản phẩm - string
- profileName: tên người đánh giá - string

- helpfulness: phần trăm người dùng thấy bài đánh giá hữu ích
- score: điểm đánh giá (từ 0 - 5) - float
- time: timestamp thời điểm đánh giá
- summary: tiêu đề đánh giá - string
- text: nội dung đánh giá - string

Tương tự ta cũng thực hiện convert file categories.txt.gz và thu được file csv với các cột:

- productId: mã sản phẩm.
- Các cột category: CategoryLevel1, CategoryLevel2,... CategoryLevel9. Tổng cộng mỗi product có 9 cấp category, level càng lớn thì cấp càng sâu.

Để so sánh mức độ hiệu quả của các kỹ thuật khuyến nghị, ngoài so sánh các điểm số như RMSE, MSE,... ta cũng cần quan tâm tới Top N result, tức là Top N sản phẩm mà hệ khuyến nghị đưa ra. Điều này cực kì quan trọng vì trong thực tế, điểm RMSE hoặc MSE có thể tốt, nhưng Top N result mới là điều cần lưu tâm hơn, bởi vì nó chính là kết quả thực tế mà sẽ show ra cho người dùng cuối.

Để dễ dàng so sánh Top N result, ta sẽ chọn ra một user duy nhất để so sánh Top N result cho user này:

Mã user: A281NPSIMI1C2R

Danh sách các rating của user:

Mã sản phẩm	Tên sản phẩm	Rating	Timestamp

B000BR15G K	Komenuka Bijin 10-Product Trial/Sample Set from Rice Bran	5	1155772800
B000BR15G K	Komenuka Bijin 10-Product Trial/Sample Set from Rice Bran	5	1153872000
B0006FQN5 0	Venom Gloss	5	1207612800
B000LC648 O	Barielle Intensive Nail Renewal Oil .50 Fl.Oz.	5	1170460800
B000FBK4L O	Thymes Bath Salts Envelope (2 oz)	5	1231718400
B00021DUM 2	Bare Escentuals bareMinerals Blush	5	1143331200
B000FKJZK G	Zia Natural Skincare Sea Tonic Aloe Toner, 6-Ounce Spray Bottles (Pack of 2)	5	1169769600

B000GEWA 26	Derma e Complete E Cranberry Creme, Intense Moisturizing Formula- Creme Hydrante, 2 oz (Pack of 2)	5	1159574400
----------------	--	---	------------

Mô tả đặc điểm từng sản phẩm để sau này chúng ta dễ so sánh và đoán được sở bộ sở thích của người dùng:

1. Komenuka Bijin 10-Product Trial/Sample Set from Rice Bran: là một bộ sản phẩm thử nghiệm gồm 10 sản phẩm chăm sóc da và tóc từ cám gạo. Bộ sản phẩm này bao gồm các sản phẩm như sữa rửa mặt, dầu gội, dầu xả, kem dưỡng da và các sản phẩm khác, giúp bạn trải nghiệm toàn diện các lợi ích của cám gạo trong việc chăm sóc da và tóc. Bộ sản phẩm này được thiết kế để cung cấp dưỡng chất, làm sạch và bảo vệ da và tóc. Có thể sử dụng bộ sản phẩm này để có một trải nghiệm spa tại nhà, mang lại cảm giác thư giãn và cải thiện sức khỏe làn da và tóc.

2. Komenuka Bijin 10-Product Trial/Sample Set from Rice Bran: là một bộ sản phẩm thử nghiệm gồm 10 sản phẩm chăm sóc da và tóc từ cám gạo. Bộ sản phẩm này bao gồm các sản phẩm như sữa rửa mặt, dầu gội, dầu xả, kem dưỡng da và các sản phẩm khác, giúp trải nghiệm toàn diện các lợi ích của cám gạo trong việc chăm sóc da và tóc.

3. Barielle Intensive Nail Renewal Oil: là một loại dầu dưỡng móng tay và móng chân, giúp phục hồi và bảo vệ móng khỏi hư tổn. Sản phẩm này chứa các thành phần như dầu mè, dầu hoa rum, dầu hướng dương và dầu dừa, cùng với keratin và vitamin E, giúp giữ ẩm và làm mềm lớp biểu bì. Sản phẩm này cũng giúp ngăn ngừa gãy móng và làm lành các lớp biểu bì khô.

4. Thymes Bath Salts Envelope: là một sản phẩm muối tắm giúp làm mềm và dưỡng ẩm da. Sản phẩm này chứa muối Epsom giàu khoáng chất tự nhiên, kết hợp với các thành

phần dưỡng da như mật ong, dầu jojoba và lô hội. Các loại muối tắm của Thymes có nhiều mùi hương khác nhau như Lavender, Eucalyptus, và Kimono Rose, mang lại cảm giác thư giãn và dễ chịu khi ngâm mình trong bồn tắm.

5. The bareMinerals Blush by Bare Escentuals: là lựa chọn phổ biến cho vẻ ngoài tự nhiên, rạng rỡ, được thiết kế để mang lại màu sắc mềm mại, dễ tán, phù hợp với mọi tông màu da. Phấn má hồng này được biết đến với công thức sạch, không chứa talc và có nhiều tông màu phù hợp với sở thích khác nhau.

6. Zia Natural Skincare Sea Tonic Aloe Toner: là một loại toner dưỡng da không chứa cồn, giúp cung cấp độ ẩm và làm dịu da. Sản phẩm này chứa các thành phần tự nhiên như nước hoa hồng, lô hội, và chiết xuất tảo biển, giúp làm dịu và nuôi dưỡng làn da.

7. Derma E Complete E Cranberry Creme: là một loại kem dưỡng ẩm mạnh mẽ, được thiết kế để cung cấp độ ẩm sâu và làm dịu da khô, ngứa. Sản phẩm này chứa chiết xuất từ quả nam việt quất (cranberry), một nguồn giàu tocotrienols và các axit béo omega-3 và omega-6, kết hợp với vitamin E để mang lại độ ẩm và phục hồi độ mềm mại cho da.

Qua danh sách trên, ta có thể đoán được sơ bộ người dùng này là phụ nữ, và quan tâm đến những sản phẩm về da, đặc biệt là các sản phẩm từ thiên nhiên hoặc có thành phần dưỡng ẩm cao. Ngoài ra người này cũng quan tâm đến các sản phẩm về tóc, chăm sóc móng và trang điểm.

Nhìn chung người này:

- **Ưu tiên các sản phẩm từ thiên nhiên**
- **Tập trung vào dưỡng ẩm và phục hồi**
- **Yêu thích trải nghiệm spa tại nhà**

Tiền xử lý dữ liệu

Ta sẽ xây dựng class **AmazonReview** để thực hiện tiền xử lý dữ liệu từ file csv:

```
1 class AmazonReview:
2     productID_to_name =
3         name_to_productID = {}
4             {}
5
# Tập này sẽ được xử lý như sau: file Beaty.csv gốc sẽ được lọc ra các cột cần thiết (userId, productId, score, time)
6     ratingsPath =
7         '/home/haphuthinh/Workplace/School_project/do-an-1/Recommender-system-UIT/AmazonRatingData/Beauty-rating.csv'
8
# Tập này sẽ được xử lý như sau: file Beaty.csv gốc sẽ được lọc ra các cột cần thiết (productId, title)
9     productsPath =
10        '/home/haphuthinh/Workplace/School_project/do-an-1/Recommender-system-UIT/AmazonRatingData/Beauty-product.csv'
11
12    def loadAmazonReviewDataset(self):
13        # Định dạng reader cho tập ratings
14        reader = Reader(line_format='user item rating timestamp', sep=',', skip_lines=1)
15
16        # Tải dữ liệu ratings vào Surprise Dataset
17        ratingsDataset = Dataset.load_from_file(self.ratingsPath, reader=reader)
18
19        # Đọc file products để xây dựng ánh xạ productId .. title
20        self.productID_to_name =
21        self.name_to_productID = {}
22        with open(self.productsPath, newline='', encoding='ISO-8859-1') as csvfile:
23            productReader = csv.reader(csvfile)
24            next(productReader) # Bỏ qua header
25            for row in productReader:
26                productId = row[0]
27                productName = row[1]
28                self.productID_to_name[productId] = productName
29                self.name_to_productID[productName] = productId
30
31    return ratingsDataset
32
33    def getPopularityRanks(self):
34        # Tính độ phổ biến của sản phẩm từ tập ratings
35        ratings = defaultdict(int)
36        rankings = defaultdict(int)
37        with open(self.ratingsPath, newline='', encoding='utf-8') as csvfile:
38            reviewReader = csv.reader(csvfile)
39            next(reviewReader)
40            for row in reviewReader:
41                productId = row[1]
42                ratings[productId] += 1
43            rank = 1
44            for productId, ratingCount in sorted(ratings.items(), key=lambda x: x[1], reverse=True):
45                rankings[productId] = rank
46                rank += 1
47            return rankings
48
49    def getProductName(self, productId):
50        # Lấy tên sản phẩm từ productId
51        return self.productID_to_name.get(productId, "")
52
53    def getProductId(self, productName):
54        # Lấy productId từ tên sản phẩm
55        return self.name_to_productID.get(productName, 0)
```

Mục đích của class này là cung cấp các method để dễ dàng truy cập vào dữ liệu rating từ file csv như getProductName(), getProductId(), getPopularityRank(): hàm này sẽ tính ra độ phổ biến của sản phẩm.

3.1.1. Thực nghiệm Content based filtering

- Xây dựng bộ dữ liệu:

Do có sử dụng thêm các brand từ file **brands.txt** nên ta sẽ dùng bộ dữ liệu **Clothing_&_Accessories.txt** (các đánh giá người dùng cho ngành hàng Thời trang của Amazon). Tập đánh giá này có tổng cộng 581,933 reviews và nặng 78MB cho file nén.

Cấu trúc mẫu của tập brands.txt:

- B0000A0QE6 Jos. A. Bank
- B0000A0QE2 Jos. A. Bank
- B0000C2LF3 Canyon Ridge
- B0000C2LF7 Canyon Ridge

với cột đầu tiên là productId và cột thứ hai là brand

Cấu trúc mẫu của tập cũng tương tự như tập **Beauty Review** đã trình bày phái trên. Sau khi convert sang file CSV và kết hợp thêm thông tin của tập brand.txt cũng như loại bỏ các thông tin không cần thiết thì sẽ cho ra tập dữ liệu mẫu **products.csv** có dạng:

- **productId:** Mã sản phẩm (string)
- **title:** Tên sản phẩm (string)
- **price:** Giá thành sản phẩm (double)
- **userId:** Mã người dùng (string)
- **score:** Điểm đánh giá (double)
- **time:** Thời điểm đánh giá (long)
- **brand:** Thương hiệu (string)

- Để dễ dàng so sánh Top N result, ta sẽ chọn ra một user duy nhất để so sánh Top N result cho user này:

- Mã user: A2LGUI0DDHZ41X
- Danh sách các rating của user:

productId	title	price	userId	score	time	brand
B00062NOGO	Amazon.com: Columbia Men's Steens Mountain Sweater: Clothing	177.7	A2LGUI0DDHZ41X	5	1357344000	Columbia
B000AJGPM	Amazon.com: Levi's 517 Rigid Boot Cut Jeans 40W X 30L: Clothing	61.61	A2LGUI0DDHZ41X	5	1071014400	Levi's
B0002IVG90	Amazon.com: Moving Comfort Women's Fiona Bra: Clothing	127.76	A2LGUI0DDHZ41X	5	1316822400	Moving Comfort
B0000CBXHW	Amazon.com: Dominique Longline Lace Torsollette Style 8949: Clothing	72.04	A2LGUI0DDHZ41X	4	1255478400	Dominique
B00028B0OU	Amazon.com: Dickies Men's Original 874 Washed Work Pant: Clothing	113.02	A2LGUI0DDHZ41X	5	1357516800	Dickies
B000KEKGC8	Amazon.com: The Office - Gun Show T-Shirt: Clothing	186.8	A2LGUI0DDHZ41X	5	1294790400	The Office
B0009E76A	Amazon.com: Spasilk 100% Cotton Sleeveless Lap Shoulder 3-Pack Bodysuit: Clothing	74.84	A2LGUI0DDHZ41X	4	1356566400	Spasilk
B0001F3NAS	Amazon.com: JanSport Classics Series Superbreak Backpack (Alien Green): Sports & Outdoors	106.64	A2LGUI0DDHZ41X	5	1353369600	JanSport
B0006G42IE	Amazon.com: Weatherproof Men's Silk Cotton Button Down Woven Long Sleeve Shirt, Marine, Large: Clothing	25.7	A2LGUI0DDHZ41X	5	1120435200	Weatherproof
B0002VN5K4	Amazon.com: Russell Athletic Men's Basic Cotton Long Sleeve Tee: Clothing	127.62	A2LGUI0DDHZ41X	5	1350864000	Russell Athletic
B00078R086	Amazon.com: Jerzees 562 8 oz 50/50 Crew Neck: Clothing	136.18	A2LGUI0DDHZ41X	1	1321574400	Jerzees
B0000C2SX4	Amazon.com: Dockers Men's Signature Khaki Big & Tall Pleated Pant: Clothing	183.65	A2LGUI0DDHZ41X	5	1355270400	Dockers
B0002FHIMQ	Amazon.com: Fruit of the Loom 4930 Cotton Long-Sleeve T-Shirt: Clothing	101.6	A2LGUI0DDHZ41X	3	1299283200	Fruit of the Loom
B00063716M	Amazon.com: Glamorise Women's Wonderwire Seamless Molded Support Bra #9195: Clothing	165.12	A2LGUI0DDHZ41X	5	1333152000	Jessica London
B000G7W6Z0	Therasock SmartKnit Men's / Women's Seamless Crew Socks.	184.14	A2LGUI0DDHZ41X	5	1251072000	
B0000APLWJ	Amazon.com: Wrinkle-Resistant Cotton Twill with Side Elastic Pleated Pants (NAVY, 46 33"; INSEAM): Clothing	169.58	A2LGUI0DDHZ41X	5	1320624000	Jos. A. Bank
B000E58WJ2	Amazon.com: JanSport Classics Series Superbreak Backpack (Alien Green): Sports & Outdoors	189.04	A2LGUI0DDHZ41X	4	1156723200	JanSport
B0000AWG3T	Amazon.com: Dickies Men's Original 874 Washed Work Pant: Clothing	115.23	A2LGUI0DDHZ41X	4	1293494400	Dickies
B000ND1LYGI	Amazon.com: JanSport Classics Series Superbreak Backpack (Alien Green): Sports & Outdoors	125.59	A2LGUI0DDHZ41X	5	1314748800	JanSport
B00026YTZY	Amazon.com: Classic Low Rise Brief - 2 Pack, black, 32: Clothing	182.25	A2LGUI0DDHZ41X	5	1361923200	Jockey

- Triển khai code

Ta sẽ sử dụng AlgoBase từ thư viện Surprise, và sẽ tiến hành xây dựng thuật toán

ContentKNN từ đó:

```
from surprise import AlgoBase
```

Tiếp đến chúng ta sẽ load Dataset từ lớp ProductData:

```
pd = ProductData()
```

sau đó ta sẽ tiến hành lấy các thuộc tính mà chúng ta cần để tiến hành xây dựng Recommender System. Và ở trường hợp này chính là **price** và **brand**

Định nghĩa hàm getPrices trong class ProductData

```

def getPrices(self):
    scaler = MinMaxScaler()
    prices = defaultdict(str) # Dictionary mặc định kiểu string

    # Đọc dữ liệu từ file CSV và lưu giá vào danh sách tạm
    product_ids = []
    price_list = []

    with open(self.productsPath, newline='', encoding='ISO-8859-1') as csvfile:
        productReader = csv.reader(csvfile)
        next(productReader) # Bỏ qua dòng header

        for row in productReader:
            productID = row[0] # Lấy productId từ cột thứ 2 (index 1)
            try:
                price = float(row[2].strip()) # Lấy price từ cột thứ 5 (index 4) và loại bỏ khoảng trắng
            except ValueError:
                continue # Bỏ qua nếu không thể chuyển đổi giá trị price thành float

            if productID:
                product_ids.append(productID)
                price_list.append(price)

    # Chuẩn hóa giá trị price
    price_array = np.array(price_list).reshape(-1, 1) # Chuyển đổi thành mảng 2D để dùng với MinMaxScaler
    price_scaled = scaler.fit_transform(price_array)

    # Lưu kết quả vào dictionary
    for productID, scaled_price in zip(product_ids, price_scaled):
        prices[productID] = scaled_price[0] # scaled_price là mảng, lấy giá trị đầu tiên

    return prices

```

Định nghĩa hàm getBrands trong class ProductData

```

def getBrands(self):
    brands = defaultdict(str) # Tạo dictionary mặc định kiểu str
    with open(self.productsPath, newline='', encoding='ISO-8859-1') as csvfile:
        productReader = csv.reader(csvfile)
        next(productReader) # Bỏ qua dòng header

        for row in productReader:
            productID = row[0] # Lấy productId từ cột thứ 2 (index 1)
            brand = row[6].strip() # Lấy brand từ cột thứ 7 (index 6) và loại bỏ khoảng trắng

            # Kiểm tra brand không rỗng hoặc NaN trước khi thêm vào dictionary
            if brand and productID:
                brands[productID] = brand

    return brands

```

Các hàm trên là các hàm dùng để xây dựng các vector đặc trưng từ các thuộc tính brand và price. Sau đó sử dụng các hàm này trong để xây dựng thuật toán ContentKNN:

```

brands = pd.getBrands()
prices = pd.getPrices()

```

Tiếp theo ta sẽ tiến hành tính toán độ tương đồng của của brand và price thông qua các hàm:

computeBrandSimilarity

```
def computeBrandSimilarity(self, product1, product2, brands):
    if brands[product1] == brands[product2]:
        return 1.0
    else:
        return 0.0
```

computePriceSimilarity

```
def computePriceSimilarity(self, product1, product2, prices):
    return 1.0 - abs(prices[product1] - prices[product2])
```

và dùng độ tương đồng đó để tiến hành train dữ liệu bằng hàm fit của AlgoBase

```
def fit(self, trainset):
    AlgoBase.fit(self, trainset)

    # Compute item similarity matrix based on content attributes

    # Load up genre vectors for every movie
    pd = ProductData()
    brands = pd.getBrands()
    prices = pd.getPrices()

    print("Computing content-based similarity matrix...")

    # Compute genre distance for every movie combination as a 2x2 matrix
    self.similarities = np.zeros((self.trainset.n_items, self.trainset.n_items))

    for thisRating in range(self.trainset.n_items):
        if (thisRating % 100 == 0):
            print(thisRating, " of ", self.trainset.n_items)
        for otherRating in range(thisRating+1, self.trainset.n_items):
            thisProductId = self.trainset.to_raw_iid(thisRating)
            otherProductId = self.trainset.to_raw_iid(otherRating)
            brandSimilarity = self.computeBrandSimilarity(thisProductId, otherProductId, brands)
            priceSimilarity = self.computePriceSimilarity(thisProductId, otherProductId, prices)
            self.similarities[thisRating, otherRating] = brandSimilarity * priceSimilarity
            self.similarities[otherRating, thisRating] = self.similarities[thisRating, otherRating]

    print("...done.")

    return self
```

Sau khi đã xây dựng xong thuật toán thì ta sẽ tiến hành đánh giá

```
contentKNN = ContentKNNAlgorithm()
evaluator.AddAlgorithm(contentKNN, "ContentKNN")
```

```
evaluator.Evaluate(True)
```

```
evaluator.SampleTopNRecs(pd)
```

- Kết quả thu được

Trên tập 119754 record:

productId	title	price	predictedScore	brand
B0000C3XY1	Amazon.com: Austin Reed Andover Jacket, Petites: Clothing	55.07	5	Austin Reed
B0000C3XZ0	Amazon.com: Austin Reed Andover Jacket, Petites: Clothing	42.04	5	Austin Reed
B000A7XQ7	Amazon.com: Toolshed Men's 3/4 Shed Compression Short (sz. S, White : Flmaes): Clothing	53.72	5	Toolshed
B000A7XQX	Amazon.com: Toolshed Men's 3/4 Shed Compression Short (sz. L, Sledge Grey): Clothing	57.33	5	Toolshed
B000A7XS1	Amazon.com: Toolshed Men's 3/4 Shed Compression Short (sz. XXXL, Sledge White): Clothing	164.62	5	Toolshed
B0006V1NX6	Amazon.com: National 3/4 Sleeve Blouse: Clothing	152.44	5	National
B000F318V2	Amazon.com: Hollywood Gel Blood, 1 Oz.: Clothing	40.43	5	Cinema Secrets
B0000ANHKV	Men's HeatGear® Compression 7" Shorts Bottoms by Under Armour	52.64	5	
B0000ANHKJ	Men's HeatGear® Compression 7" Shorts Bottoms by Under Armour	185.79	5	
B0006SQ04W	Amazon.com: Cacharel Men's Tonal Diagonal Pinstripe Dress Shirt, Tan, 16.5-34/35: Clothing	154.17	5	Cacharel

Algorithm	RMSE	MAE
ContentKNN	1.3886	1.0322

- Đánh giá chung:

RMSE và MAE tuy không quá tệ nhưng vẫn chưa đạt mức tốt, chỉ ở mức trung bình khá đối với một hệ thống Content-based Filtering. Nguyên nhân dẫn đến điểm số đó có thể là do tập dữ liệu mẫu dùng để train chưa thực sự tốt, còn quá nhiều dữ liệu nhiễu.

Về phần kết quả đề xuất thì có thể thấy:

- o Về price: trả về các sản phẩm thuộc khoảng giá phù hợp khi $\text{Min}(x'[\text{price}]) > \text{Min}(x[\text{price}])$ ($40.43 > 25.7$) và $\text{Max}(x'[\text{price}]) < \text{Max}(x[\text{price}])$ ($185.79 < 189.04$).
- o Về brand: Trả về các sản phẩm thuộc các brand hoàn toàn khác so với lịch sử đánh giá. Điều này có thể xảy ra do 2 nguyên nhân:

- Thuật toán để trích xuất đặc trưng và tính độ tương đồng của thuộc tính brand chưa tốt.
- Mỗi brand trong dữ liệu mẫu vô tình chỉ có một sản phẩm (trường hợp này ít xảy ra).

Tóm lại:

Cần cải thiện thêm về thuật toán để làm giảm RSME và MAE cũng như dự đoán chính xác các thuộc tính.

3.1.2. Thực nghiệm Neighborhood based collaborative filtering

Neighborhood based collaborative filtering, hay còn gọi là KNN (**k-Nearest Neighbors**) áp dụng các thuật toán dựa trên khoảng cách hoặc độ tương đồng để cận gần nhất. Phương pháp này đã được trình bày lý thuyết ở phần ...

- Triển khai code

Ta sẽ sử dụng KNNBasic từ thư viện Surprise, thư viện này giúp ta dễ dàng triển khai KNN:



```
from surprise import KNNBasic
```

Tiếp đến ta thực hiện load dataset từ class **AmazonReview**: amazon review dataset, evaluation data (dữ liệu để đánh giá) và ranking (độ phổ biến của từng product trong tập review). Các dữ liệu này sẽ được đưa vào bộ Evaluator:

```
1 # Load up common data set for the recommender algorithms
2 (amazonReview, evaluationData, rankings) = LoadAmazonReviewData()
3
4 # Construct an Evaluator
5 evaluator = Evaluator(evaluationData, rankings, amazonReview)
```

Như đã tìm hiểu ở phần cơ sở lý thuyết, KNN có hai dạng: User-based và Item-based, ta sẽ code như sau:

- User-based:

```
1 UserKNN = KNNBasic(sim_options={'name': 'pearson', 'user_based': True})
2 evaluator.AddAlgorithm(UserKNN, "User KNN")
3
```

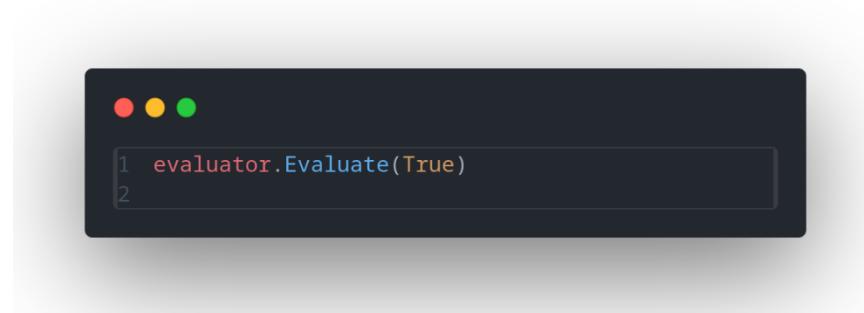
- Item-based:

```
1 ItemKNN = KNNBasic(sim_options={'name': 'pearson', 'user_based': False})
2 evaluator.AddAlgorithm(ItemKNN, "Item KNN")
```

Ở đây ta dùng Person để tính toán độ tương đồng:

$$r = \frac{\sum (x - m_x)(y - m_y)}{\sqrt{\sum (x - m_x)^2 \sum (y - m_y)^2}}$$

Sau đó ta thực hiện chạy evaluate để đánh giá kết quả:



- Kết quả thu được

Trên 100 records

Algorithm	RMSE	MAE	HR	cHR	ARHR	Coverage	Diversity	Novelty
User KNN	1.6429	1.256	0	0	0	1	0.9789	9.9732
Item KNN	1.6316	1.2016	0.0366	0.0366	0.0366	1	0.9789	9.9732

Trên 10k records

Algorithm	RMSE	MAE	HR	cHR	ARHR	Coverage	Diversity	Novelty
User KNN	1.3079	1.035	0.0011	0.0011	0.0005	1	0.9778	516.2702
Item KNN	1.2922	1.0048	0.0049	0.0049	0.0047	1	0.9784	515.884

Top 10 Result:

User KNN:

- Oscar Eau de Toilette for Women by Oscar de La Renta: 4.17

- Optimum Care Anti-Breakage Therapy Moisture Replenish Cream Hairdress: 4.17
- Artec Textureline Smoothing Serum, 8.4-Ounce Pump (Pack of 2): 4.17
- Michel Design Works Bath Gift Set - Beach: 4.17
- Liz Claiborne Curve Eau de Toilette Spray: 4.17
- Palladio Herbal Lipstick,: 4.17
- SOFT SHEEN Carson Optimum Care Anti-Breakage Therapy Featherlight Hairdress 4oz/113g: 4.17
- Island Essence Lotion: 4.17
- Creed Spring Flower by Creed for Women - 2.5 oz Millesime Spray: 4.17
- Island Essence Lotion: 4.17

Total time for recommending top N: 1.2868332862854004 seconds

Memory usage: 1716.23828125 MB

Item KNN:

- Oscar Eau de Toilette for Women by Oscar de La Renta: 4.17
- Optimum Care Anti-Breakage Therapy Moisture Replenish Cream Hairdress: 4.17
- Artec Textureline Smoothing Serum, 8.4-Ounce Pump (Pack of 2): 4.17
- Michel Design Works Bath Gift Set - Beach: 4.17
- Liz Claiborne Curve Eau de Toilette Spray: 4.17
- Palladio Herbal Lipstick,: 4.17
- SOFT SHEEN Carson Optimum Care Anti-Breakage Therapy Featherlight Hairdress 4oz/113g: 4.17
- Island Essence Lotion: 4.17
- Creed Spring Flower by Creed for Women - 2.5 oz Millesime Spray: 4.17
- Island Essence Lotion: 4.17

Total time for recommending top N: 0.04280233383178711 seconds

Memory usage: 1123.48046875 MB

- **Đánh giá chung**

Nhìn chung Item KNN (0.004s) cho ra kết quả nhanh hơn User KNN (1.28s) đúng như giải thích ở phần lý thuyết bởi vì Item KNN sử dụng một ma trận nhỏ hơn so với User KNN.

Ngoài ra, tất cả các điểm số đánh giá như RMSE, MSE,... của Item KNN cũng cho kết quả tốt hơn User KNN.

Riêng Diversity và Novelty cả hai xấp xỉ ngang nhau.

Xét về Top N recommend, cả 2 đều cho ra kết quả gần như giống nhau.

Tuy nhiên item ở top 1 là Oscar Eau de Toilette for Women by Oscar de La Renta và Liz Claiborne Curve Eau de Toilette Spray ở top 5: đây là các sản phẩm nước hoa, trong khi user ta đang xét chưa từng đánh giá nước hoa bao giờ.

Optimum Care Anti-Breakage Therapy Moisture Replenish Cream Hairdress ở top 2 là một sản phẩm chăm sóc tóc, đề xuất này khá hợp lý vì user này có vẻ hứng thú với các sản phẩm chăm sóc tóc.

3.1.3. Thực nghiệm Matrix factorization method

Matrix Factorization cũng là một kỹ thuật thuộc Collaborative Filtering, sử dụng phân rã ma trận để đưa ra khuyến nghị. Lý thuyết của Matrix Factorization đã được trình bày ở phần ...

- **Triển khai code**

Chúng ta sẽ triển khai Matrix Factorization bằng cách áp dụng SVD (Singular value decomposition) - một kỹ thuật thuộc Matrix Factorization.

SVD cũng được thư viện Surprise hỗ trợ và có thể dễ dàng triển khai:

```
● ● ●
```

```
1 from surprise import SVD
```

Ta thực hiện load dataset từ class AmazonRating theo cách tương tự như đã trình bày ở phần thực nghiệm KNN:

```
● ● ●
```

```
1 (amazonReview, evaluationData, rankings) = LoadAmazonReviewData()
```

Khởi tạo bộ evalutation để đánh giá kết quả:

```
● ● ●
```

```
1 evaluator = Evaluator(evaluationData, rankings, amazonReview)
```

Thực hiện khởi tạo SVD và đưa vào bộ evaluate để tiến hành chạy test:



```
1 SVDUntuned = SVD()
2 evaluator.AddAlgorithm(SVDUntuned, "SVD - Untuned")
3
4 evaluator.Evaluate(True)
```

Thực hiện lấy Top N result cho user test, với 10 là số lượng result sẽ lấy.



```
1 evaluator.SampleTopNRecs("A281NPSIMI1C2R", 10)
2
```

- Kết quả thu được

Tập 100 record:

Algorithm	RMSE	MAE	HR	cHR	ARHR	Coverage	Diversity	Novelty
SVD Untuned	1.5573	1.1706	0.0122	0.0122	0.0061	0.9878	0.9808	14.2912

Tập 10k record:

Algorithm	RMSE	MAE	HR	cHR	ARHR	Coverage	Diversity	Novelty
SVD Untuned	1.2555	0.9857	0.005	0.005	0.0022	0.9998	0.9927	102.437

Top 10 result:

- Liz Claiborne Curve Eau de Toilette Spray: 5.00
- Oscar By Oscar De La Renta For Women. Eau De Toilette Spray 8 Ounces: 5.00
- So De La Renta By Oscar De La Renta For Women. Eau De Toilette Spray 3.3 Ounces: 5.00
- Revlon SkinLights Instant Skin Brightener, SPF 15, Warm Light 03, 1.5 Fluid Ounce (44.3 ml): 5.00
- ANAIS ANAIS For Women By CACHAREL Eau de Toilette Spray: 5.00
- Calvin Klein Truth Eau de Parfum Spray: 5.00
- Faconnable Cologne by Faconnable for men Colognes: 5.00
- Moisturizing Day Cream: 5.00
- Max Factor Pan-Stik Ultra Creamy Makeup .5 oz (14 g): 5.00
- Revlon Moondrops lipstick: 5.00

Total time for recommending top N: 0.36658763885498047 seconds

Memory usage: 1069.44921875 MB

Tuy nhiên, nếu chỉ lấy top 10 thì prediction rating bị trùng là 5.00 quá nhiều, vậy ta sẽ thử lấy top N lớn hơn với N = 30:

- Liz Claiborne Curve Eau de Toilette Spray: 5.00
- Oscar By Oscar De La Renta For Women. Eau De Toilette Spray 8 Ounces: 5.00

- So De La Renta By Oscar De La Renta For Women. Eau De Toilette Spray 3.3 Ounces: 5.00
- Revlon SkinLights Instant Skin Brightener, SPF 15, Warm Light 03, 1.5 Fluid Ounce (44.3 ml): 5.00
- ANAIS ANAIS For Women By CACHAREL Eau de Toilette Spray: 5.00
- Calvin Klein Truth Eau de Parfum Spray: 5.00
- Faconnable Cologne by Faconnable for men Colognes: 5.00
- Moisturizing Day Cream: 5.00
- Max Factor Pan-Stik Ultra Creamy Makeup .5 oz (14 g): 5.00
- Revlon Moondrops lipstick: 5.00
- Qtica Intense Cuticle Repair (select option/size): 5.00
- Camille Beckman Silky Body Cream: 5.00
- Garnier Nutrisse Haircolor, 452 Dark Reddish Brown Chocolate Cherry: 5.00
- HAPPY For Men By CLINIQUE Cologne Spray: 5.00
- No-Rinse Body Bath with Odor Eliminator - 16 oz.: 5.00
- Elemis Soothing Toner, Apricot, 6.8 Ounce: 5.00
- Island Song Coco-Mango Body Lotion 6.25 fl oz Bottle: 5.00
- DUNE For Women By CHRISTIAN DIOR Eau De Toilette Spray: 5.00
- Hawaiian Tropic Aloe After SUN Moisturizer 16 oz. Pump: 5.00
- Canus Li'l Goat's Milk Baby Butter, 8-Ounce Jars (Pack of 3): 5.00
- Eau d'Hadrien Perfume by Annick Goutal for women Personal Fragrances: 5.00
- PERLIER by Perlier: PINK PEONY MOISTURE SHIELD BODY CREAM--7OZ: 5.00
- Halston Catalyst Eau de Toilette Spray: 5.00
- Neutrogena Visibly Firm Body Lotion, Active Copper, 8.5 Ounce (Pack of 2): 4.99

- American Crew Firm Hold Styling Gel for Men, 15.2-Ounce Bottles (Pack of 2): 4.99
- Aw Mendenhall #02303 5lb Heavy Duty Powder Hand Soap: 4.99
- Kayline Dryer & 3-irons Pedestal Appliance Holder #pb5 "Chrome Finish": 4.98
- Elemis Sp@home Instant Refreshing Gel 5.10 fl oz (150 ml): 4.98
- Kent OS11 Soft Men's Hairbrush: 4.98
- Pravana Intense Therapy Leave-In Treatment 10.1oz: 4.98

Total time for recommending top N: 0.3407282829284668 seconds

Memory usage: 1068.0234375 MB

Có thể thấy điểm số đánh giá RMSE, MSE tốt hơn đáng kể khi có data nhiều hơn, đúng với tính chất của Matrix Factorization (ma trận càng nhiều dữ liệu càng dễ tìm ra các đặc trưng).

Tuy nhiên, SVD được sử dụng ở trên vẫn chưa được điều chỉnh các tham số tùy chỉnh, vậy nên ta sẽ thử tuning lại mô hình này để đạt được điểm số tốt hơn:

```
1 param_grid = {'n_epochs': [20, 30], 'lr_all': [0.005, 0.010],
2                 'n_factors': [50, 100]}
3 gs = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=3)
4
5 gs.fit(evaluationData)
6
7 # best RMSE score
8 print("Best RMSE score attained: ", gs.best_score['rmse'])
9
10 # # combination of parameters that gave the best RMSE score
11 print(gs.best_params['rmse'])
12
13 # Construct an Evaluator to, you know, evaluate them
14 evaluator = Evaluator(evaluationData, rankings, amazonReview)
15
16 params = gs.best_params['rmse']
17 SVDtuned = SVD(n_epochs = params['n_epochs'], lr_all = params['lr_all'], n_factors
18 = params['n_factors'])
19 evaluator.AddAlgorithm(SVDtuned, "SVD - Tuned")
```

Ở đây ta áp dụng kỹ thuật gọi là Grid Search CV. **Grid Search CV** (Cross-Validation) là một kỹ thuật được sử dụng trong Machine Learning để tìm kiếm **bộ siêu tham số tối ưu** cho một mô hình bằng cách thử tất cả các kết hợp có thể trong một tập hợp các giá trị tham số đã định trước. Đây là một phương pháp phổ biến để **tinh chỉnh tham số** (hyperparameter tuning).

Như ảnh trên, ta đang áp dụng Grid Search CV để tìm các tham số: epoch (20 hoặc 30), lr_all: (0.005 hoặc 0.010), n_factors (50 hoặc 100) tối ưu nhất, sao cho điểm số RMSE nhận được là thấp nhất.

Kỹ thuật này sẽ thực hiện kết hợp tất cả các tổ hợp mà ta đã định nghĩa: ví dụ epoch 20 + lr_all 0.005 + n_factors 50, epoch 20 + lr_all 0.005 + n_factors 100,... cứ thế đến khi đủ tổ hợp. Sau đó nó sẽ chạy model với từng tổ hợp và ghi nhận tổ hợp nào có điểm RMSE thấp nhất, đó chính là bộ tổ hợp tham số tối ưu mà ta cần tìm.

Ở phần đầu ta đã chạy SVD với tổ hợp tham số mặc định: epoch 20, n_factors: 100, lr_all: 0.005, lần này ta sẽ thử chạy Grid search cv với kết quả nhận được là:

```

Best RMSE score attained: 1.2741591241422183
{'n_epochs': 30, 'lr_all': 0.01, 'n_factors': 50}

```

Như vậy với bộ 3 tham số epoch: 30, lr_all: 0.01, n_factors: 50 ta sẽ đạt được điểm số RMSE tối ưu nhất.

Kết quả:

Algorithm	RMSE	MAE	HR	cHR	ARHR	Coverage	Diversity	Novelty
SVD - Tuned	1.25	0.9647	0.0047	0.0047	0.002	0.9989	0.9951	154.0847

Có thể thấy điểm số RMSE và MAE đã có cải thiện so với lúc chưa tuning.

Top N result:

- Optimum Care Anti-Breakage Therapy Moisture Replenish Cream Hairdress: 5.00
- Liz Claiborne Curve Eau de Toilette Spray: 5.00
- Creed Spring Flower by Creed for Women - 2.5 oz Millesime Spray: 5.00
- Azulen Supreme Facial Cream 1.66 oz by Dr. Eckstein: 5.00
- Curve Crush FOR WOMEN by Liz Claiborne - 0.50 oz EDT Spray: 5.00
- Exuviance Purifying Cleansing Gel-7.2 oz: 5.00
- Focus 21 Jojoba Shampoo 32oz.: 5.00
- MAC Eye Shadow Steamy 1.5 g / 0.05 oz: 5.00
- Palladio Herbal Eyeliner Pencil: 5.00
- Exuviance Moisture Balance Toner: 5.00
- Pevonia Evolutive Eye Gel "C" (1.0 oz): 5.00
- Revitalash MD - Eyelash Growth - Eyelash Extensions: 5.00
- Osis + Sparkler Shine Spray Hair Styling Serums: 5.00
- by CELLEX-C,: 5.00

- Wigo 1 Inch Digital Ceramic Cruling Iron WG5402: 5.00
- ARAMIS By Aramis For Men COLOGNE 4 OZ (UNBOXED): 5.00
- Palladio Retractable Waterproof Lip Liner: 5.00
- Exuviance - Rejuvenating Treatment Masque: 5.00
- 5 Minute Colorant by Just 5 Hair Color, Darkest Brown - 1 Ea: 5.00
- CoverBlend Multi-Function Concealer SPF 15: 5.00
- Neutrogena Visibly Firm Body Lotion, Active Copper, 8.5 Ounce (Pack of 2): 5.00
- Exuviance Vespera Bionic Serum: 5.00
- EARTH THERAPEUTICS Loofah Body Scrubber 7" Sponge 1 UNIT: 5.00
- Exuviance Evening Restorative Complex: 5.00
- Tresor by Lancome for Women 3.4 oz Refreshing Skin Mist 2007 Summer Edition: 5.00
- Denman Pop Up Brush In Blue: 5.00
- NeoCeuticals Acne Spot Treatment Gel: 5.00
- Perlier Pink Peony Hand Cream 3.4 fl.oz.: 5.00
- NeoStrata Bionic Face Cream PHA 12: 5.00
- NeoStrata Daytime Protection Cream SPF 15: 5.00

Total time for recommending top N: 0.3125131130218506 seconds

Memory usage: 1141.13671875 MB

- **Đánh giá chung**

Nhìn chung có thể thấy SVD sau khi tuning có điểm số tốt hơn đáng kể so với KNN ở điểm RMSE, MSE. Điểm CHR, ARHR ở SVD và Item-KNN có sự tương đồng, tuy nhiên điểm Diversity và Novelty có sự khác biệt:

- Item và User KNN có Diversity cao hơn SVD, điều này chứng tỏ KNN có xu hướng tìm ra những sản phẩm mới lạ nhiều hơn so với sự tương đồng giữa các sản phẩm. Điều này hợp lý vì ở Top N result của KNN đã để lọt

vào sản phẩm nước hoa mặc dù sản phẩm này chưa chắc giống với những sản phẩm đã được đánh giá cao bởi user.

- Item và User KNN có điểm Novelty cao hơn đáng kể, gấp 5 lần so với SVD, điều này chứng tỏ KNN có xu hướng recommend những món đồ nổi tiếng nhiều hơn là so với khuyến nghị của SVD

Các chỉ số Novelty và Diversity này là tốt hay xấu còn tùy thuộc vào nhiều trường hợp do ta quyết định, ví dụ nếu ta muốn user được recommend những sản phẩm mới lạ, thì Diversity cao lại là điều tốt. Ngược lại, nếu ta muốn user được recommend những sản phẩm thật giống nhau, thì Diversity thấp lại phù hợp.

Xét đến top N result, đã có sự cải thiện đáng kể so với KNN, với item ở top 1 là Optimum Care Anti-Breakage Therapy Moisture Replenish Cream Hairdress - đây là một sản phẩm chăm sóc tóc, và nó có vẻ phù hợp với user mà ta đang test này.

Ở top 3 là Azulen Supreme Facial Cream là một loại kem dưỡng da, sản phẩm này cũng khá hợp lý.

Ngoài ra cũng có khá nhiều sản phẩm về tóc và da khác được khuyến nghị.

3.1.4. Thực nghiệm Deep learning method

- Thực nghiệm RBM

RBM (Restricted boltzmann machine) là kỹ thuật khuyến nghị sử dụng Deep Learning, cơ sở lý thuyết của nó đã được trình bày ở phần ...

○ Triển khai code

Với RBM, ta sẽ sử dụng Tensorflow để xây dựng mạng neural như sau:

Đầu tiên, ta định nghĩa class RBM để chứa mạng, class này sẽ nhận các tham số sau đây:

`visibleDimensions`: Số lượng đặc trưng của lớp visible (lớp đầu vào).

`epochs`: Số vòng lặp huấn luyện (mặc định là 20).

`hiddenDimensions`: Số lượng đơn vị ẩn (hidden units).

`ratingValues`: Số lượng giá trị xếp hạng (mặc định là 10).

`learningRate`: Tốc độ học (mặc định là 0.001).

`batchSize`: Kích thước của mỗi batch khi huấn luyện (mặc định là 100).

```
1 class RBM(object):
2
3     def __init__(self, visibleDimensions, epochs=20, hiddenDimensions=50, ratingValues=10,
4                  learningRate=0.001, batchSize=100):
5
6         self.visibleDimensions = visibleDimensions
7         self.epochs = epochs
8         self.hiddenDimensions = hiddenDimensions
9         self.ratingValues = ratingValues
10        self.learningRate = learningRate
11        self.batchSize = batchSize
```

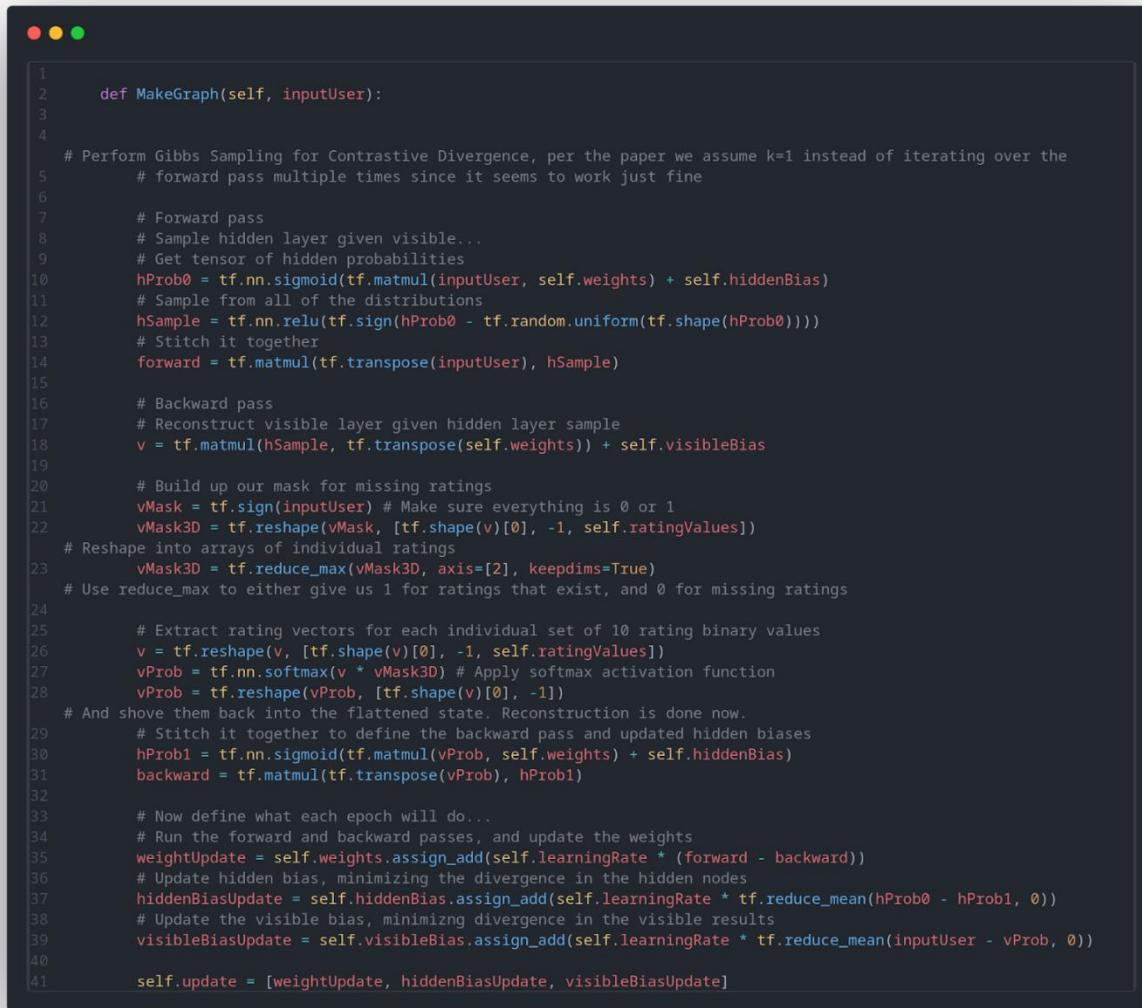
Tiếp đến ta có phương thức **Train** để thực hiện train model:

```
1     def Train(self, X):
2
3
4         # Initialize weights randomly (earlier versions of the code had this block inside MakeGraph, but that was a bug.)
5         maxWeight = -4.0 * np.sqrt(6.0 / (self.hiddenDimensions + self.visibleDimensions))
6         self.weights = tf.Variable(tf.random.uniform([self.visibleDimensions, self.hiddenDimensions], minval=-maxWeight,
7                                                       maxval=maxWeight), tf.float32, name="weights")
8         self.hiddenBias = tf.Variable(tf.zeros([self.hiddenDimensions]), tf.float32, name="hiddenBias")
9         self.visibleBias = tf.Variable(tf.zeros([self.visibleDimensions]), tf.float32, name="visibleBias")
10
11        for epoch in range(self.epochs):
12
13            trX = np.array(X)
14            for i in range(0, trX.shape[0], self.batchSize):
15                epochX = trX[i:i+self.batchSize]
16                self.MakeGraph(epochX)
17
18            print("Trained epoch ", epoch)
```

Phương thức này sẽ khởi tạo trọng số ngẫu nhiên và các bias cho lớp visible và hidden.

Sau đó, qua mỗi epoch, dữ liệu huấn luyện được chia thành các batch nhỏ. Trong mỗi batch, phương thức **MakeGraph** được gọi để tính toán và cập nhật trọng số.

Tiếp đến ta có phương thức MakeGraph thực hiện các bước chính trong **Gibbs Sampling** để huấn luyện mô hình:



```
1  def MakeGraph(self, inputUser):
2
3
4      # Perform Gibbs Sampling for Contrastive Divergence, per the paper we assume k=1 instead of iterating over the
5      # forward pass multiple times since it seems to work just fine
6
7      # Forward pass
8      # Sample hidden layer given visible...
9      # Get tensor of hidden probabilities
10     hProb0 = tf.nn.sigmoid(tf.matmul(inputUser, self.weights) + self.hiddenBias)
11
12     # Sample from all of the distributions
13     hSample = tf.nn.relu(tf.sign(hProb0 - tf.random.uniform(tf.shape(hProb0))))
14
15     # Stitch it together
16     forward = tf.matmul(tf.transpose(inputUser), hSample)
17
18     # Backward pass
19     # Reconstruct visible layer given hidden layer sample
20     v = tf.matmul(hSample, tf.transpose(self.weights)) + self.visibleBias
21
22     # Build up our mask for missing ratings
23     vMask = tf.sign(inputUser) # Make sure everything is 0 or 1
24     vMask3D = tf.reshape(vMask, [tf.shape(v)[0], -1, self.ratingValues])
25
26     # Reshape into arrays of individual ratings
27     vMask3D = tf.reduce_max(vMask3D, axis=[2], keepdims=True)
28
29     # Use reduce_max to either give us 1 for ratings that exist, and 0 for missing ratings
30
31     # Extract rating vectors for each individual set of 10 rating binary values
32     v = tf.reshape(v, [tf.shape(v)[0], -1, self.ratingValues])
33     vProb = tf.nn.softmax(v * vMask3D) # Apply softmax activation function
34     vProb = tf.reshape(vProb, [tf.shape(v)[0], -1])
35
36     # And shove them back into the flattened state. Reconstruction is done now.
37     # Stitch it together to define the backward pass and updated hidden biases
38     hProb1 = tf.nn.sigmoid(tf.matmul(vProb, self.weights) + self.hiddenBias)
39     backward = tf.matmul(tf.transpose(vProb), hProb1)
40
41     # Now define what each epoch will do...
42     # Run the forward and backward passes, and update the weights
43     weightUpdate = self.weights.assign_add(self.learningRate * (forward - backward))
44     # Update hidden bias, minimizing the divergence in the hidden nodes
45     hiddenBiasUpdate = self.hiddenBias.assign_add(self.learningRate * tf.reduce_mean(hProb0 - hProb1, 0))
46     # Update the visible bias, minimizing divergence in the visible results
47     visibleBiasUpdate = self.visibleBias.assign_add(self.learningRate * tf.reduce_mean(inputUser - vProb, 0))
48
49     self.update = [weightUpdate, hiddenBiasUpdate, visibleBiasUpdate]
```

Forward pass: Dự đoán lớp hidden từ lớp visible:

- Sử dụng hàm sigmoid để tính xác suất của các đơn vị ẩn.
- Sau đó lấy mẫu từ các phân phối này.

Backward pass: Dự đoán lại lớp **visible** từ lớp **hidden**:

- Dùng hàm softmax để tính xác suất của các giá trị **visible**.
- Tính lại xác suất của các đơn vị **hidden** từ lớp **visible** đã được tái tạo.

Cập nhật trọng số và bias: Trọng số, bias cho lớp visible và hidden được cập nhật bằng cách sử dụng tốc độ học ([learningRate](#)).

Phương thức [MakeHidden](#) để dự đoán lớp ẩn (hidden) từ đầu vào của người dùng ([inputUser](#))

```
1  def MakeHidden(self, inputUser):
2      hidden = tf.nn.sigmoid(tf.matmul(inputUser, self.weights) + self.hiddenBias)
3      self.MakeGraph(inputUser)
4      return hidden
```

Trong đó, nó sẽ sử dụng hàm sigmoid để tính xác suất của các đơn vị ẩn.

Sau đó gọi [MakeGraph](#) để thực hiện quá trình cập nhật trọng số.

Sau đó ta có hàm [MakeVisible](#) để dự đoán lớp visible từ lớp ẩn ([feed](#)), dùng hàm sigmoid để tính xác suất của các giá trị visible.

```
1     def MakeVisible(self, feed):
2         visible = tf.nn.sigmoid(tf.matmul(feed, tf.transpose(self.weights)) + self.
visibleBias)
3         return visible
```

Cuối cùng, ta có hàm **GetRecommendations** dựa vào dữ liệu đầu vào của người dùng để

- Tính toán lớp **hidden** từ input người dùng thông qua phương thức **MakeHidden**.
- Dự đoán lại lớp **visible** (dự đoán các xếp hạng) từ lớp ẩn bằng phương thức **MakeVisible**.

```
1     def GetRecommendations(self, inputUser):
2
3         feed = self.MakeHidden(inputUser)
4         rec = self.MakeVisible(feed)
5         return rec[0]
```

- Kết quả thu được

Tập 100 record

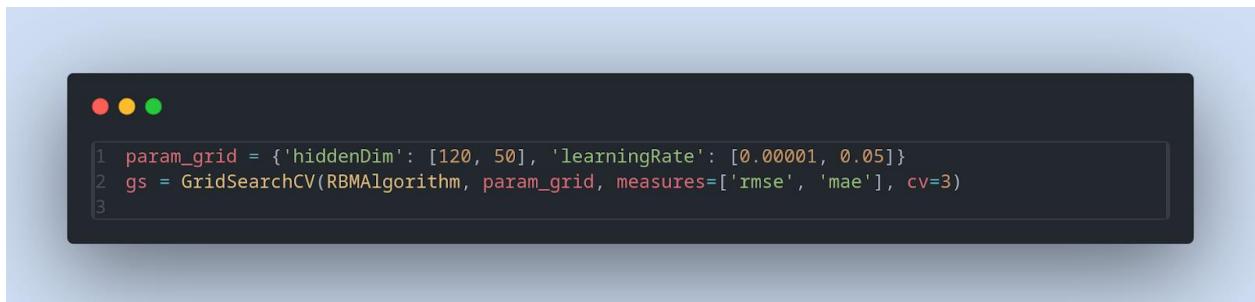
Algorithm	RMSE	MAE

RBM - Untuned	1.608	1.3626
---------------	-------	--------

Tập 10k record

Algorithm	RMSE	MAE
RBM - Untuned	1.376	1.1125

Tiếp đến ta thử áp dụng kỹ thuật Grid Search CV (như đã áp dụng ở phần SVD) để tìm các tham số tối ưu cho model:



```

1 param_grid = {'hiddenDim': [120, 50], 'learningRate': [0.00001, 0.05]}
2 gs = GridSearchCV(RBMAlgorithm, param_grid, measures=['rmse', 'mae'], cv=3)
3

```

Ở đây ta đang chạy GridSearchCV để tìm tổ hợp tối ưu cho các tham số hiddenDim (120, 50), learningRate(0.00001, 0.05). Trong khi các tham số default khi chưa tuning là: hiddenDim = 100, learningRate = 0.01.

Vì việc training model rất tốn tài nguyên, nên ta sẽ thực hiện việc dò tham số này ở trên Google Colab:

```

# evaluator.AddAlgorithm(RBMUntuned, "RBM - Untuned")
# Fight!
evaluator.Evaluate(True)

... Loading Amazon ratings...

Computing product popularity ranks so we can measure novelty later...
Searching for best parameters...
Training new model...
Trained epoch 0
Trained epoch 1
Trained epoch 2
Trained epoch 3
Trained epoch 4
Trained epoch 5
Trained epoch 6
Trained epoch 7
Trained epoch 8
Trained epoch 9
Trained epoch 10
Trained epoch 11
Trained epoch 12
Trained epoch 13
Trained epoch 14
Trained epoch 15
Trained epoch 16
Trained epoch 17
Trained epoch 18
Trained epoch 19
Model saved to rbm_model
Processing user 0

```

Kết quả sau khi dò tham số, ta thu được kết quả:

“Best RMSE score attained: 1.3767759747292028 {'hiddenDim': 120, 'learningRate': 0.05}”.

Vậy ta sẽ thử các tập tham số này để đánh giá lại lần nữa và thu được kết quả:

Algorithm	RMSE	MAE
RBM - Tuned	1.3737	1.1109

Có vẻ điểm số RMSE và MAE đã cải thiện đôi chút.

Kết quả top N result:

- Liz Claiborne Curve Eau de Toilette Spray
- Exuviance Purifying Cleansing Gel-7.2 oz
- Neutrogena Energizing Sugar Body Scrub, Fresh Citrus, 6 Ounce (Pack of 2)

- Exuviance Vespera Bionic Serum
- Exuviance Evening Restorative Complex
- Eau d'Hadrien Perfume by Annick Goutal for women Personal Fragrances
- Soffeet Callus Reducer Kit
- Revlon SkinLights Instant Skin Brightener, SPF 15, Bare Light 04, 1.5 Fluid Ounce (44.3 ml)
- Revlon SkinLights Instant Skin Brightener, SPF 15, Warm Light 03, 1.5 Fluid Ounce (44.3 ml)
- Comodynes Self-tanning Natural & Uniform Color Towelette 8 Pack

Total time for recommending top N: 0.11958909034729004 seconds

Memory usage: 2403.109375 MB

○ Đánh giá chung

Qua điểm số RMSE và MSE của RBM chưa vượt qua được SVD, hay thậm chí là KNN, tuy nhiên, với lượng data không đủ lớn (10k record) thì các phương pháp như Deep Learning có thể không phát huy hiệu quả.

Xét trên top 10 result:

- Top 1 item của RBM giống với User-KNN, đó là **Liz Claiborne Curve Eau de Toilette Spray**: là nước hoa không nằm trong danh sách ưa thích ban đầu, nhưng có thể thu hút nếu người dùng thích mở rộng sang các sản phẩm làm đẹp cá nhân.
- **Exuviance Purifying Cleansing Gel, Exuviance Vespera Bionic Serum, Exuviance Evening Restorative Complex**: Tập trung vào làm sạch, dưỡng ẩm và phục hồi da, phù hợp với sở thích của người dùng.
- **Neutrogena Energizing Sugar Body Scrub**: Tẩy tế bào chết, có mùi hương dễ chịu (Fresh Citrus), khả năng cao phù hợp.

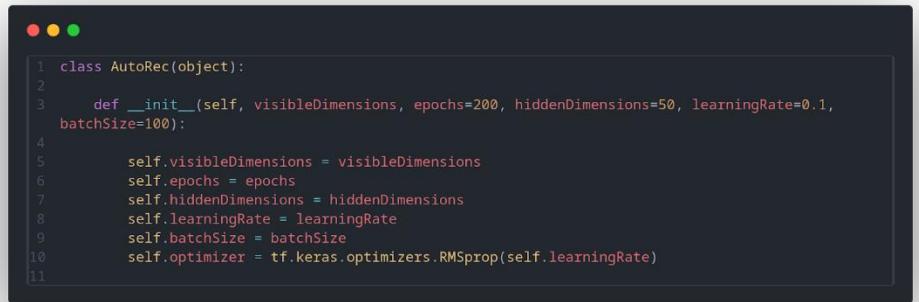
- **Sof'feet Callus Reducer Kit:** Có vẻ xa lạ với các sản phẩm đã đánh giá cao trước đó, nhưng có thể phù hợp nếu người dùng quan tâm đến chăm sóc toàn diện.
- **Revlon SkinLights Instant Skin Brightener, SPF 15:** Sản phẩm làm sáng và bảo vệ da khỏi tia UV, có khả năng phù hợp nếu người dùng quan tâm đến trang điểm tự nhiên.
- **Comodynes Self-tanning Natural & Uniform Color Towelette:** Không thấy dấu hiệu người dùng quan tâm đến sản phẩm tự nhuộm da từ danh sách trước.

- Thực nghiệm Autorec

Autorec cũng là một phương pháp thuộc Deep Learning for Recommender System, đã trình bày lý thuyết ở phần trên.

○ Triển khai Code

Với Autorec, ta sẽ dùng Tensorflow để xây dựng model, đầu tiên ta xây dựng class AutoRec:



```

1  class AutoRec(object):
2
3      def __init__(self, visibleDimensions, epochs=200, hiddenDimensions=50, learningRate=0.1,
4                   batchSize=100):
5          self.visibleDimensions = visibleDimensions
6          self.epochs = epochs
7          self.hiddenDimensions = hiddenDimensions
8          self.learningRate = learningRate
9          self.batchSize = batchSize
10         self.optimizer = tf.keras.optimizers.RMSprop(self.learningRate)
11

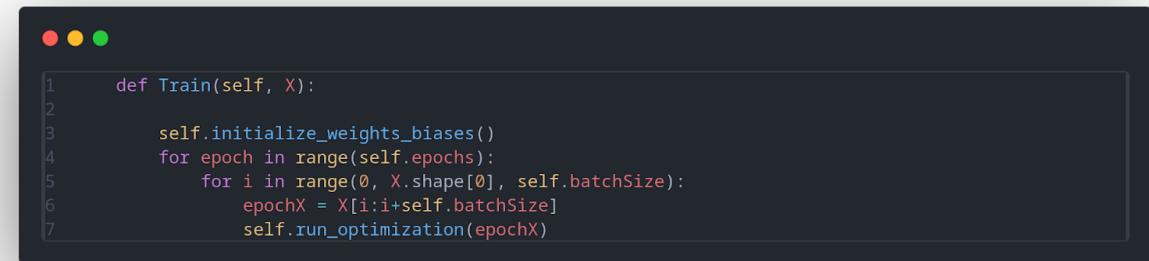
```

Class AutoRec này sẽ nhận các tham số:

- **visibleDimensions:** Số lượng đặc trưng đầu vào (ví dụ: số lượng sản phẩm hoặc đặc điểm).
- **epochs:** Số lần duyệt toàn bộ dữ liệu.

- **hiddenDimensions**: Kích thước của tầng ẩn (số chiều nén dữ liệu ở tầng mã hóa).
- **learningRate**: Tốc độ học để cập nhật weights.
- **batchSize**: Kích thước của từng lô dữ liệu trong quá trình huấn luyện.
- **optimizer**: Sử dụng thuật toán **RMSprop** để tối ưu hóa trọng số.

Tiếp đến, ta có hàm Train để huấn luyện mô hình qua nhiều epochs.



```

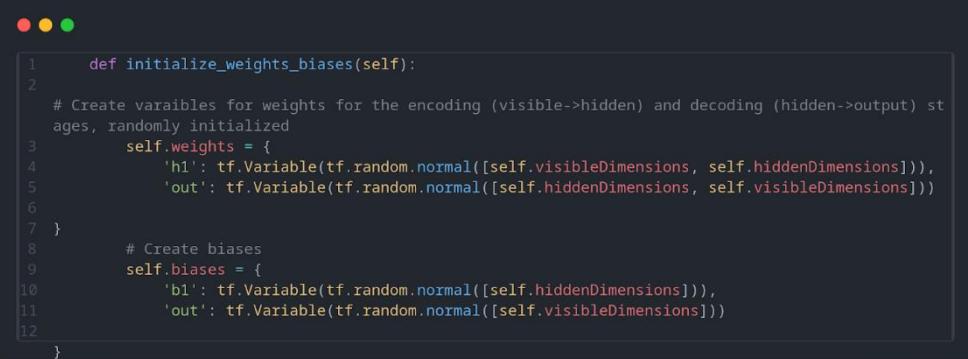
1 def Train(self, X):
2
3     self.initialize_weights_biases()
4     for epoch in range(self.epochs):
5         for i in range(0, X.shape[0], self.batchSize):
6             epochX = X[i:i+self.batchSize]
7             self.run_optimization(epochX)

```

Trong đó:

- **Hàm initialize_weights_biases()**: Khởi tạo trọng số và bias ngẫu nhiên.
- Trong mỗi epoch, chia dữ liệu thành từng lô (batch) có kích thước **batchSize**.
- Với mỗi batch, gọi **run_optimization()** để tối ưu hóa trọng số và bias.

Chi tiết về hàm **initialize_weights_biases()**:



```
1     def initialize_weights_biases(self):
2
3         # Create variables for weights for the encoding (visible->hidden) and decoding (hidden->output) stages, randomly initialized
4         self.weights = {
5             'h1': tf.Variable(tf.random.normal([self.visibleDimensions, self.hiddenDimensions])),
6             'out': tf.Variable(tf.random.normal([self.hiddenDimensions, self.visibleDimensions]))
7         }
8
9         # Create biases
10        self.biases = {
11            'b1': tf.Variable(tf.random.normal([self.hiddenDimensions])),
12            'out': tf.Variable(tf.random.normal([self.visibleDimensions]))
13        }
```

Hàm này có nhiệm vụ khởi tạo:

- Trọng số (weights):
 - h1: Trọng số tầng encode (input → hidden).
 - out: Trọng số tầng decode (hidden → output).
- Bias (biases):
 - b1: Bias cho tầng encode.
 - out: Bias cho tầng decode.

Trọng số và bias được khởi tạo ngẫu nhiên từ phân phối chuẩn.

Tiếp đến ta có hàm `neural_net(self, inputUser)`, dùng để mô phỏng mạng Autoencoder qua các bước encode và decode.

```
1     def neural_net(self, inputUser):
2
3         #tf.set_random_seed(0)
4
5
6         # Initialization of weights and biases was moved out to the initialize_weights_biases function above
7         # This lets us avoid resetting them on every batch of training, which was a bug in earlier versions of
8         # this script.
9
10        # Create the input layer
11        self.inputLayer = inputUser
12
13        # hidden layer
14        hidden = tf.nn.sigmoid(tf.add(tf.matmul(self.inputLayer, self.weights['h1']), self.biases['b1']))
15
16        # output layer for our predictions.
17        self.outputLayer = tf.nn.sigmoid(tf.add(tf.matmul(hidden, self.weights['out']), self.biases['out']))
18
19        return self.outputLayer
```

Ta có hàm `run_optimization(self, inputUser)` để tính toán Sai số giữa đầu vào (`inputUser`) và đầu ra dự đoán (`pred`) sử dụng hàm MSE (Mean Squared Error), sau đó áp dụng gradient descent để cập nhật weight và bias:

```
1     def run_optimization(self, inputUser):
2
3         with tf.GradientTape() as g:
4             pred = self.neural_net(inputUser)
5             loss = tf.keras.losses.MSE(inputUser, pred)
6
7             trainable_variables = list(self.weights.values()) + list(self.biases.values())
8
9             gradients = g.gradient(loss, trainable_variables)
10
11             self.optimizer.apply_gradients(zip(gradients, trainable_variables))
```

Cuối cùng ta có hàm `GetRecommendations(self, inputUser)` để trả về các giá trị dự đoán cho người dùng:

```

1 def GetRecommendations(self, inputUser):
2
3     # Feed through a single user and return predictions from the output layer.
4     rec = self.neural_net(inputUser)
5
6     # It is being used as the return type is Eager Tensor.
7     return rec[0]

```

- Kết quả thu được

Tập 100 records

Chạy với tham số default: epochs=100, hiddenDim=100, learningRate=0.01, batchSize=100, sim_options={ }:

Algorithm	RMSE	MAE	HR	cHR	ARHR	Coverage	Diversity	Novelty
AutoRec	2.0318	1.5776	0	0	0	0.9878	0.934	12.341

Tập 10k records

Algorithm	RMSE	MAE	HR	cHR	ARHR	Coverage	Diversity	Novelty
AutoRec	1.439	1.114	0.0006	0.0006	0.0003	1	0.9758	553.2887

Thử dùng Grid Search CV để tìm tham số tối ưu: param_grid = {'hiddenDim': [120, 50, 200], 'learningRate': [0.00001, 0.05, 0.1]}.

Kết quả:

“Best RMSE score attained: 1.3744000277682027

{'hiddenDim': 200, 'learningRate': 0.05}”

Thử áp dụng bộ tham số tìm được với hiddenDim: 200 và learningRate:0.05:

Algorithm	RMSE	MAE	HR	cHR	ARHR	Coverage	Diversity	Novelty
AutoRecTuned	1.4025	1.0915	0.0012	0.0012	0.0006	1	0.9779	520.5799

Top N result:

Oscar Eau de Toilette for Women by Oscar de La Renta

Optimum Care Anti-Breakage Therapy Moisture Replenish Cream Hairdress

Artec Textureline Smoothing Serum, 8.4-Ounce Pump (Pack of 2)

Michel Design Works Bath Gift Set - Beach:

Liz Claiborne Curve Eau de Toilette Spray

Palladio Herbal Lipstick,: 4.17

SOFT SHEEN Carson Optimum Care Anti-Breakage Therapy Featherlight Hairdress
4oz/113g

Island Essence Lotion

Creed Spring Flower by Creed for Women - 2.5 oz Millesime Spray

Island Essence Lotion

Total time for recommending top N: 813.9953329563141 seconds

Memory usage: 1377.51171875 MB

- **Đánh giá chung**

Đáng ngạc nhiên là mặc dù Autorec là một mạng neural hiện đại hơn RBM, thế nhưng điểm số RMSE lại không tốt hơn, còn MSE thì tương đương nhau. Điều này chứng tỏ một hệ thống khuyến nghị phụ thuộc vào tập data rất nhiều, có thể tập data sử dụng trong đè tài

này không phù hợp với Autorec, hoặc do lượng data không đủ nhiều để Autorec phát huy hết khả năng của nó.

Xét về top 10 result:

- Cũng như KNN và RBM, các sản phẩm nước hoa như **Oscar Eau de Toilette for Women by Oscar de La Renta** và **Creed Spring Flower by Creed** vẫn bị lọt vào.
- **Optimum Care Anti-Breakage Therapy Moisture Replenish Cream Hairdress** và **SOFT SHEEN Carson Optimum Care Anti-Breakage Therapy Featherlight Hairdress**: Phù hợp với sở thích chăm sóc tóc, đặc biệt nếu người dùng quan tâm đến dưỡng ẩm và ngăn ngừa tóc gãy rụng.
- **Artec Textureline Smoothing Serum**: Serum dưỡng tóc với tác dụng làm mượt, phù hợp với xu hướng sử dụng sản phẩm chăm sóc tóc của người dùng.
- **Island Essence Lotion (xuất hiện 2 lần)**: Sản phẩm dưỡng da, phù hợp với sở thích dưỡng ẩm và nuôi dưỡng da của người dùng.
- **Michel Design Works Bath Gift Set - Beach**: Bộ sản phẩm tắm có thể mang lại trải nghiệm spa tại nhà, phù hợp với sở thích thư giãn.
- **Palladio Herbal Lipstick**: Trang điểm nhẹ nhàng với thành phần thảo mộc có thể là điểm mở rộng mới trong sở thích của người dùng.

Kết luận: tuy điểm số RMSE không hơn, nhưng top N result có vẻ phù hợp với user hơn hẳn RBM, do RBM để lọt vào các item không liên quan như: **Comodynes Self-tanning Natural & Uniform Color Towelette**, rõ ràng ta không thấy dấu hiệu người dùng quan tâm đến sản phẩm tự nhuộm da.

- **Thực nghiệm NCF**

- **Triển khai Code**

Với NCF (Neural Collaborative Filtering), ta sẽ sử dụng thư viện Librecommender để triển khai:



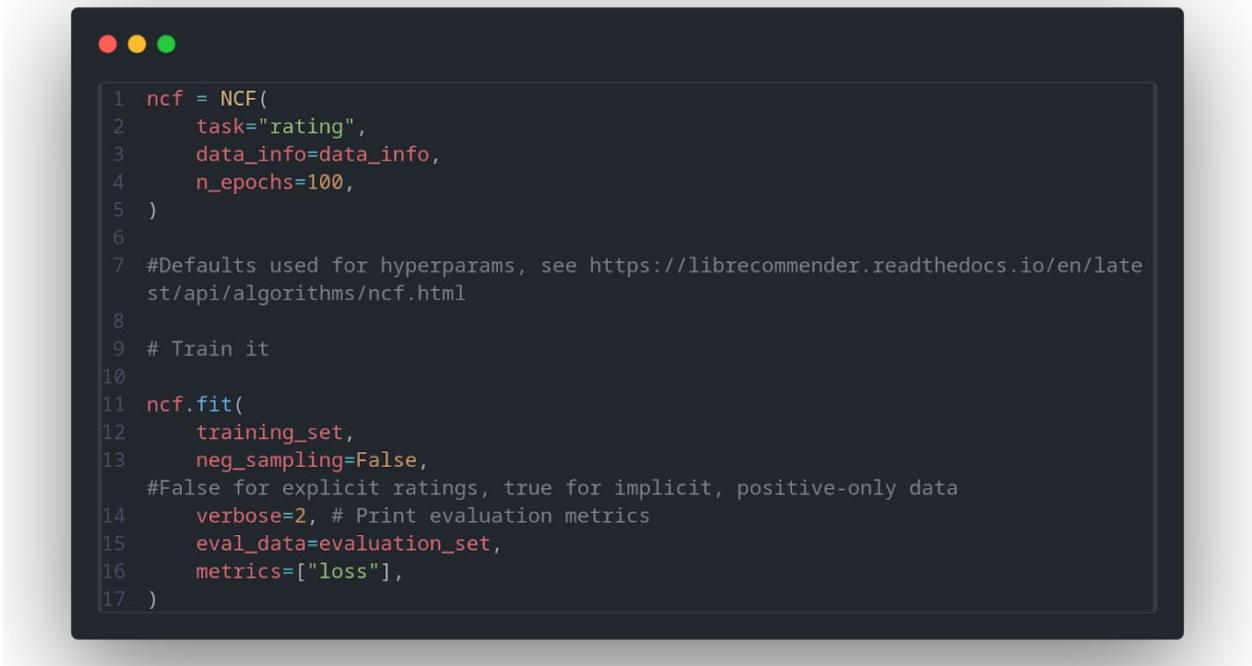
```
1 from libreco.data import random_split, DatasetPure
2 from libreco.algorithms import NCF
3 from libreco.evaluation import evaluate
```

Nhờ có Librecommender mà việc triển khai NCF trở nên khá đơn giản, đầu tiên ta cần chia tập dataset thành các tập training, evaluation, testing với tỉ lệ 80 : 10 : 10.



```
1 training_set, evaluation_set, testing_set = random_split(ratings, multi_ratios=[0.8, 0.1, 0.1])
2
3 # Convert training, evaluation, and test data into format required by LibRecommender ("Pure" collaborative filtering data)
4 training_set, data_info = DatasetPure.build_trainset(training_set)
5 evaluation_set = DatasetPure.build_evalset(evaluation_set)
6 testing_set = DatasetPure.build_testset(testing_set)
```

Tiếp đến ta thực hiện khởi tạo mô hình NCF bằng Librecommender:



```
1 ncf = NCF(
2     task="rating",
3     data_info=data_info,
4     n_epochs=100,
5 )
6
7 #Defaults used for hyperparams, see https://librecommender.readthedocs.io/en/latest/api/algorithms/ncf.html
8
9 # Train it
10
11 ncf.fit(
12     training_set,
13     neg_sampling=False,
#False for explicit ratings, true for implicit, positive-only data
14     verbose=2, # Print evaluation metrics
15     eval_data=evaluation_set,
16     metrics=["loss"],
17 )
```

task="rating":

- Định nghĩa loại bài toán là **explicit feedback** (phản hồi rõ ràng), thường sử dụng dữ liệu đánh giá (ratings) như số sao hoặc điểm số.
- Nếu bài toán là implicit feedback (phản hồi gián tiếp, ví dụ: click, mua hàng), tham số này sẽ khác.

data_info=data_info:

- Cung cấp thông tin về dữ liệu (như số lượng người dùng, số lượng mục sản phẩm, ma trận tương tác).
- **data_info** thường được sinh ra từ các tập dữ liệu huấn luyện và kiểm tra.

n_epochs=100:

- Số lượng epoch (chu kỳ huấn luyện), tức số lần duyệt qua toàn bộ tập dữ liệu.

neg_sampling=False:

- **False:** Sử dụng đánh giá rõ ràng (**explicit ratings**).
- **True:** Áp dụng **negative sampling** cho phản hồi gián tiếp (**implicit feedback**).

Ví dụ: Với dữ liệu chỉ chứa hành động tích cực (like, mua hàng), negative sampling sẽ thêm các trường hợp giả định "không tương tác".

- **Kết quả thu được**

Tập 10k ratings:

Algorithm	RMSE	MAE
NCF	0.748082399	0.486061275

Top 10 result:

Kent OS11 Soft Men's Hairbrush

Angel Cream for Women by Thierry Mugler

CLASSIC COVERMARK-NEUTRAL BEIGE

Island Essence Lotion

Green Tea Eau de Parfum for Women by Elizabeth...

Camille Beckman Hand & Shower Gel

Kenzo Flower Eau de Toilette Spray

Demeter Fragrance Library - Flowers Cologne Co...

Rubee Hand & Body Lotion 16 oz.

CREED ROYAL WATER by Creed EAU DE PARFUM SPRAY...

Total time for recommending top N: 0.019570589065551758 seconds

Memory usage: 546.25 MB

Nhận thấy rằng hiệu suất và mức tiêu thụ RAM của NCF là cực kì tốt với tập 10k ratings so với các phương pháp trước đó, ta sẽ thử test tiếp trên tập lớn hơn: 100k và 200k (full tập).

Tập 100k ratings

Algorithm	RMSE	MAE
NCF	0.916853547	0.450196862

Tập 200k ratings:

Algorithm	RMSE	MAE
NCF	0.928304732	0.48171863

Hiệu suất khi thử recommend top 10 ở tập 200k record:

- Total time for recommending top N: 0.0260622501373291 seconds
- Memory usage: 582.31640625 MB

○ Đánh giá chung

Có thể thấy NCF cho ra kết quả vô cùng tốt ở điểm RMSE ở tập 10k hay kể cả full tập 200k. Không những thế hiệu suất của nó còn vô cùng tốt nếu so với các phương pháp khác.

Tuy nhiên, như đã nói, điểm số RMSE nhiều lúc sẽ không quan trọng bằng top N result trong thực tế, khi mà NCF cho ra result là:

- Kent OS11 Soft Men's Hairbrush ở top 1: đây là sản phẩm dành cho nam, và user của chúng ta không có vẻ gì là nam cả.
- Tiếp đến, NCF đề xuất đến 4 loại nước hoa: Angel Cream for Women by Thierry Mugler, Kenzo Flower Eau de Toilette Spray, Green Tea Eau de Parfum for Women by Elizabeth Arden, CREED ROYAL WATER by Creed EAU DE PARFUM SPRAY. Và như đã trình bày, user chúng ta đang test chưa từng đánh giá nước hoa bao giờ.
- **Island Essence Lotion, Rubee Hand & Body Lotion:** là các sản phẩm dưỡng da, các sản phẩm này có vẻ phù hợp.

Nhìn chung top N result của NCF là không mấy ấn tượng mặc dù điểm RMSE và MSE rất cao.

3.1.5. Thực nghiệm ALS với Spark

- Triển khai Code

PySpark có hỗ trợ ALS - một kỹ thuật thuộc Matrix Factorization.

ALS có thể được triển khai dễ dàng dựa vào thư viện MLlib của PySpark.

Đầu tiên ta sẽ tiền xử lý dữ liệu thành dạng DataFrame (kiểu dữ liệu mà PySpark làm việc):



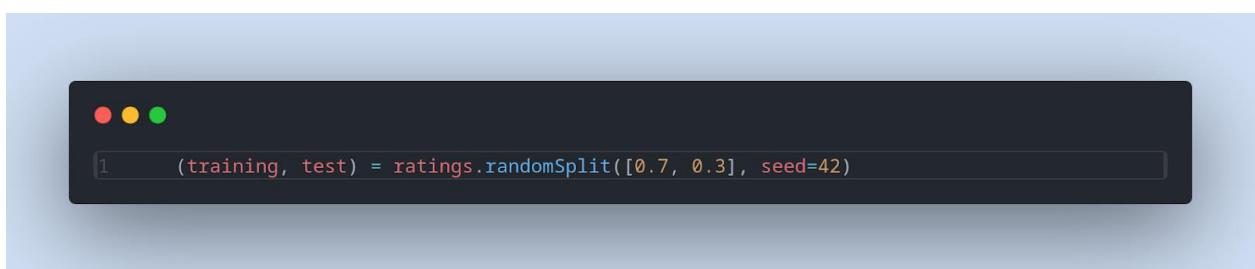
```
1 lines = spark.read.option("header", "true").csv(ratingPath).rdd
2
3 ratingsRDD = lines.map(lambda p: Row(userId=(p[0]), productId=(p[1]),
4                                         rating=float(p[2]), timestamp=int(p[3])))
5
6 ratings = spark.createDataFrame(ratingsRDD)
```

Bởi vì Spark ALS chỉ làm việc với id kiểu số, cho nên ta cần indexing user id và product id kiểu string thành kiểu số trước khi đưa vào model xử lý:



```
1 # Index session_id and post_id columns
2 userIndexer = StringIndexer(inputCol="userId", outputCol="userIdIndex")
3 productIndexer = StringIndexer(inputCol="productId", outputCol="productIdIndex")
4
5 # Fit the indexers
6 ratings = userIndexer.fit(ratings).transform(ratings)
7 ratings = productIndexer.fit(ratings).transform(ratings)
8
```

Tiếp đến ta tạo tập training và test với tỉ lệ 0.7 - 0.3:



```
1 (training, test) = ratings.randomSplit([0.7, 0.3], seed=42)
```

Tiếp đến ta định nghĩa model ALS với các thông số sau:

maxIter=20: Số lượng vòng lặp (iterations) mà thuật toán ALS sẽ thực hiện để tối ưu hóa mô hình. Ở đây, 20 vòng lặp được chỉ định.

regParam=0.1: Tham số điều chỉnh (regularization parameter) giúp tránh overfitting. Giá trị 0.1 cho biết một mức độ điều chỉnh vừa phải.

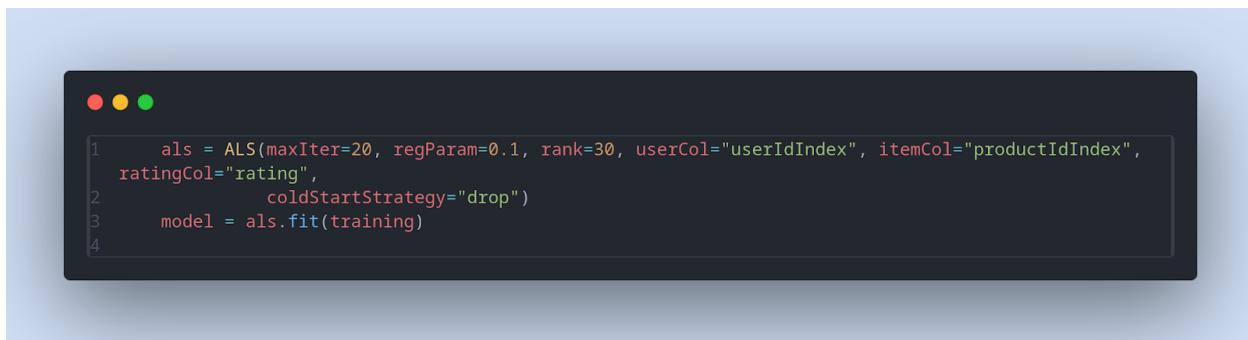
rank=30: Số lượng đặc trưng ẩn (latent factors) được sử dụng để biểu diễn người dùng và sản phẩm trong không gian đặc trưng ẩn. Giá trị này càng lớn, mô hình càng phức tạp.

userCol="userIdIndex": Tên cột đại diện cho người dùng trong tập dữ liệu (ở đây là userIdIndex).

itemCol="productIdIndex": Tên cột đại diện cho sản phẩm trong tập dữ liệu (ở đây là productIdIndex).

ratingCol="rating": Tên cột đại diện cho đánh giá hoặc xếp hạng mà người dùng dành cho sản phẩm (ở đây là rating).

coldStartStrategy="drop": Chiến lược xử lý giá trị không hợp lệ hoặc thiếu dữ liệu (cold start). Với giá trị "drop", các dự đoán không hợp lệ (ví dụ: sản phẩm hoặc người dùng mới chưa từng xuất hiện trong tập huấn luyện) sẽ bị loại bỏ.



```
1     als = ALS(maxIter=20, regParam=0.1, rank=30, userCol="userIdIndex", itemCol="productIdIndex",
2                 ratingCol="rating",
3                 coldStartStrategy="drop")
4     model = als.fit(training)
```

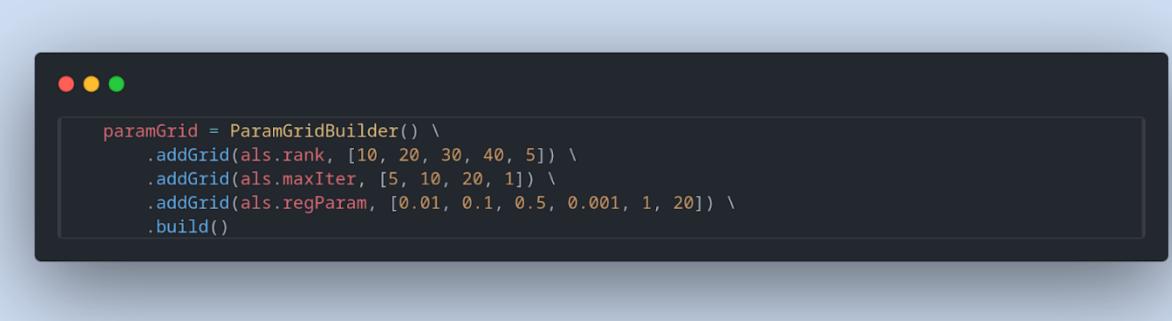
- Kết quả thu được

Tập 1k record:

RMSE	MAE	HR	cHR	ARHR	Coverage	Diversity	Novelty
3.056834	2.557977	0.000527	0.000527	0.000130	0.848101	0.017654	2.23826

Tập 10k record:

Nhận thấy điểm RMSE và MAE không tốt, ta thử tuning lại ALS với Grid Search CV trên tập 10k:



```
paramGrid = ParamGridBuilder() \
    .addGrid(als.rank, [10, 20, 30, 40, 5]) \
    .addGrid(als.maxIter, [5, 10, 20, 1]) \
    .addGrid(als.regParam, [0.01, 0.1, 0.5, 0.001, 1, 20]) \
    .build()
```

Kết quả sau khi chạy Grid Search CV:

“Best parameters: rank=30, maxIter=20, regParam=0.1”. Ta chạy thử lại model với các tham số này và thu được kết quả:

RMSE score	MAE score
1.702207687	0.88789608

Có thể thấy sau khi tuning, model hoạt động tốt hơn đáng kể.

Tập 100k record:

RMSE score	MAE score
2.048896894	1.038689784

Top N result:

- Thymes Bath Salts Envelope (2 oz)
- Bare Escentuals bareMinerals Blush

- Venom Gloss
- Derma e Complete E Cranberry Creme, Intense Moisturizing Formula-Creme Hydrante, 2 oz (Pack of 2)
- Barielle Intensive Nail Renewal Oil .50 Fl.Oz.
- Komenuka Bijin 10-Product Trial/Sample Set from Rice Bran
- Ahava Mens Deep Cleansing Gel, 3.4oz
- Ahava Mens Protective Moisturizing Fluid SPF 15, 1.7oz
- LACOSTE POUR HOMME For Men By LACOSTE Eau de Toilette Spray
- Escada Sentiment By Escada For Women. Eau De Toilette Spray 2.5 Ounces

Memory usage: 102.5234375 MB

Total time for recommending top N: 3.6173043251037598 seconds

Tập 200k record (full tập):

RMSE score	MAE score
2.22569019	1.327666885

Total time for recommending top N: 37.612921476364136 seconds

Memory usage: 114.25 MB

- Đánh giá chung

Các điểm số như RMSE và MSE không bằng các thuật toán khác, và không bằng một thuật toán khác cũng thuộc Matrix Factorization là SVD.

Xét về top N:

Thymes Bath Salts Envelope (2 oz): Sản phẩm muối tắm thư giãn phù hợp với sở thích trước đây về các trải nghiệm spa tại nhà. Có vẻ như rất phù hợp.

Derma e Complete E Cranberry Creme, Intense Moisturizing Formula: Sản phẩm dưỡng ẩm mạnh mẽ, chứa thành phần tự nhiên như chiết xuất quả nam việt quất, hoàn toàn phù hợp với sở thích dưỡng ẩm và chăm sóc da của người dùng.

Ahava Mens Deep Cleansing Gel, 3.4oz: Sữa rửa mặt cho nam, không hoàn toàn phù hợp vì người dùng trước đó không cho thấy quan tâm đặc biệt đến sản phẩm dành cho nam.

Ahava Mens Protective Moisturizing Fluid SPF 15, 1.7oz: Kem dưỡng da cho nam, tương tự như trên, có thể không phù hợp.

Bare Escentuals bareMinerals Blush: Phấn má nhẹ nhàng, với công thức tự nhiên. Nếu người dùng quan tâm đến trang điểm nhẹ nhàng, đây có thể là lựa chọn tốt. Tuy nhiên, điều này chưa rõ ràng trong sở thích trước đó.

Venom Gloss: Sản phẩm trang điểm dạng son bóng, chưa rõ mức độ phù hợp vì người dùng trước đó không cho thấy sở thích mạnh với các sản phẩm trang điểm.

LACOSTE POUR HOMME For Men By LACOSTE Eau de Toilette Spray: Nước hoa nam, không phù hợp với người dùng đã đánh giá các sản phẩm trước đó là nữ.

Lý do có thể do dataset đưa vào ALS không phù hợp, bởi vì nhiều nghiên cứu cho thấy ALS hoạt động rất tốt.

- Đổi dataset

Vậy nên ta sẽ thử đổi một bộ dataset khác là bộ Video_games_rating, cũng được trích từ tập Amazon Rating 2024. Bộ dataset có kích thước lớn hơn, với 463,669 reviews.

Ta sẽ chọn một user từ tập này để test:

User id: AJKWF4W7QD4NS

productId	Title	Rating	Timestamp
B000EHPQUS	King of Fighters 2006	3	1178496000

B00006IKBG	Sega Sports NHL 2K3	5	1039392000
B00006IKBL	The Lord of the Rings: Fellowship of the Ring	1	1035763200
B000MUXLOK	Burnout Paradise	5	1203465600
B0009XILEA	Kasumi Ninja	2	1133136000
B000021Y5S	Final Doom	5	1026172800
B00004TBGT	Spider-Man	5	1007424000
B0000V48MA	Transformers	4	1090108800
B0000TSRA6	Resident Evil: Outbreak	3	1082073600

B000A3ON14	Spiderman: Mysterio's Menace / X-2 Wolverines Revenge	5	1131235200
------------	--	---	------------

Có thể thấy:

- Người dùng đánh giá cao các tựa game như **Burnout Paradise** và **Spider-Man** (đều nhận được điểm 5), điều này cho thấy họ có sở thích với các trò chơi đầy kịch tính và hành động, đặc biệt là các trò chơi đua xe và siêu anh hùng.
- **Sega Sports NHL 2K3** cũng nhận được điểm 5, cho thấy người dùng có thể thích các trò chơi thể thao chiến lược, nơi họ có thể điều khiển và quản lý các đội tuyển thể thao.
- Các trò chơi như **Kasumi Ninja** và **King of Fighters 2006** đều thuộc thể loại đối kháng, cho thấy sở thích của người dùng với các trò chơi có yếu tố cạnh tranh cao, nhưng **King of Fighters 2006** lại chỉ nhận được điểm 3, có thể do trò chơi không đáp ứng đủ kỳ vọng.
- **The Lord of the Rings: Fellowship of the Ring** nhận được điểm 1, điều này có thể chỉ ra rằng người dùng không thích thể loại trò chơi phiêu lưu, ít nhất là đối với tựa game này.
- Các trò chơi như **Final Doom** nhận điểm cao, cho thấy người dùng có thể ưa thích những tựa game cổ điển hoặc có yếu tố hoài cổ.

KẾT QUẢ CHẠY FULL TẬP 460K RECORDS:

RMSE score	MAE score
1.206452773	0.916372698

Khác hẳn với dataset cũ, đây là một số điểm ẩn tượng.

Xét tới top N result:

- Turok 3: Shadows Of Oblivion
- Carmageddon
- Soviet Strike (Sega Saturn)
- Streets of Rage 2
- Doom II
- Midnight Outlaw Illegal Street Racing
- Herzog Zwei
- Spiderman: Mysterio's Menace / X-2 Wolverines Revenge (2-in-1 Game Cartridge)
- Rock N' Roll Racing
- Cabela's Big Game Hunter 2005

Total time for recommending top N: 84.2462785243988 seconds

Memory usage: 134.50390625 MB

Đây là một top N tốt, bởi vì:

- **Turok 3: Shadows of Oblivion:** Là một game hành động và bắn súng, phù hợp với sở thích của user với những trò chơi có yếu tố kịch tính và chiến đấu như **Final Doom** và **Spider-Man**.

- **Carmageddon:** Đây là một trò chơi đua xe với yếu tố bạo lực, rất phù hợp với sở thích của người dùng khi user này đã đánh giá cao **Burnout Paradise** (đua xe) và các game hành động.
- **Soviet Strike (Sega Saturn):** Trò chơi này thuộc thể loại bắn súng, rất phù hợp với sở thích của user đối với các trò chơi chiến đấu và hành động.
- **Streets of Rage 2:** Một trò chơi đối kháng cổ điển, rất hợp với sở thích của người dùng khi user đó đã đánh giá cao các trò chơi như **Kasumi Ninja** và **Spiderman: Mysterio's Menace**.
- **Doom II:** Trò chơi hành động bắn súng cổ điển, rất phù hợp với sở thích của user với **Final Doom** và các game bắn súng khác.
- **Midnight Outlaw Illegal Street Racing:** Trò chơi đua xe đường phố, phù hợp với sở thích của user về các trò chơi đua xe như **Burnout Paradise**.
- **Herzog Zwei:** Đây là một trò chơi chiến thuật thời gian thực, mặc dù không phải là thể loại yêu thích chủ yếu của user, nhưng có thể thích hợp nếu người dùng muốn thử nghiệm các trò chơi chiến lược, mặc dù điểm **King of Fighters 2006** chỉ đạt điểm 3 có thể cho thấy sở thích ít hơn đối với chiến thuật.
- **Spiderman: Mysterio's Menace / X-2 Wolverines Revenge:** Game này được người dùng đánh giá cao trước đó, và sẽ chắc chắn phù hợp.
- **Rock N' Roll Racing:** Một trò chơi đua xe với yếu tố giải trí cao, rất phù hợp với sở thích của user về các trò chơi đua xe và hành động.
- **Cabela's Big Game Hunter 2005:** Đây là một trò chơi săn bắn, có thể phù hợp với sở thích của người dùng nếu họ thích các trò chơi hành động có yếu tố săn bắn giống như **Cabela's Big Game Hunter**.

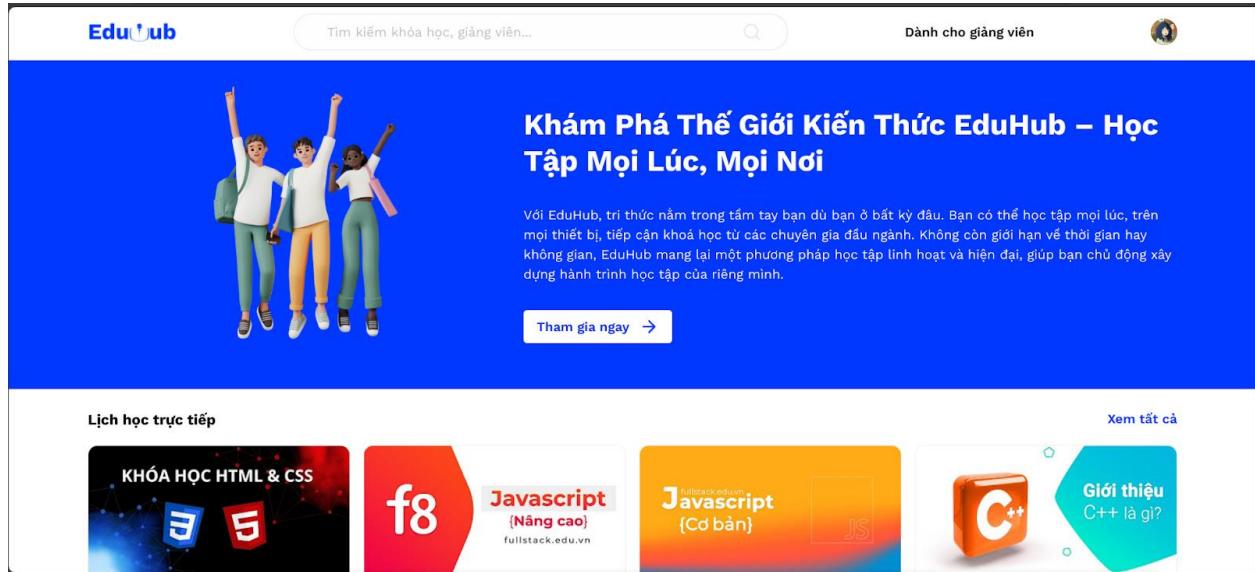
Tổng kết các đánh giá:

Algorithm	RMSE	MAE	HR	cHR	ARHR	Coverage	Diversity	Novelty
Content KNN	1.3886	1.0322						
User KNN	1.3079	1.035	0.0011	0.0011	0.0005	1	0.9778	516.2702
Item KNN	1.2922	1.0048	0.0049	0.0049	0.0047	1	0.9784	515.884
SVD Untuned	1.2555	0.9857	0.005	0.005	0.0022	0.9998	0.9927	102.437
SVD Tuned	1.25	0.9647	0.0047	0.0047	0.002	0.9989	0.9951	154.0847
RBM Untuned	1.376	1.1125						
RBM Tuned	1.3737	1.1109						
AutoRec	1.439	1.114	0.000 6	0.0006	0.0003	1	0.9758	553.2887
AutoRecTun ed	1.4025	1.0915	0.001 2	0.0012	0.0006	1	0.9779	520.5799
NCF	0.7480823 99	0.4860612 75						
ALS	1.7022076 87	0.8878960 8						

3.2. Xây dựng ứng dụng minh họa

3.2.1. Ứng dụng mua bán khóa học online

- Mô tả ứng dụng



Eduhub.io.vn là một trang web mua bán khóa học online, giúp kết nối giảng viên và học viên, cho phép giảng viên đăng ký tài khoản và đăng khóa học, học viên có thể mua khóa học và bắt đầu quá trình học tập qua video, bài trắc nghiệm, tài liệu,...

Đáng chú ý, dưới mỗi khóa học sẽ có phần “Đánh giá khóa học” và “Khóa học gợi ý”:

★ 4.0 xếp hạng khóa học • 1 đánh giá

★★★★★

Write your comment here...

Đánh giá

 **Thịnh Văn Hà 1**
★★★★☆ 11:51 18/12/2024

html css

Khóa học gợi ý



Devops cho người mới bắt đầu
Tác giả
5 ★★★★★ (1 đánh giá)
đ140,000



Javascript cơ bản
Tác giả
5 ★★★★★ (1 đánh giá)
đ100,000



Giới thiệu C++ là gì?
Lập trình C++ từ cơ bản đến nâng cao
Tác giả
4 ★★★★☆ (4 đánh giá)
đ120,000

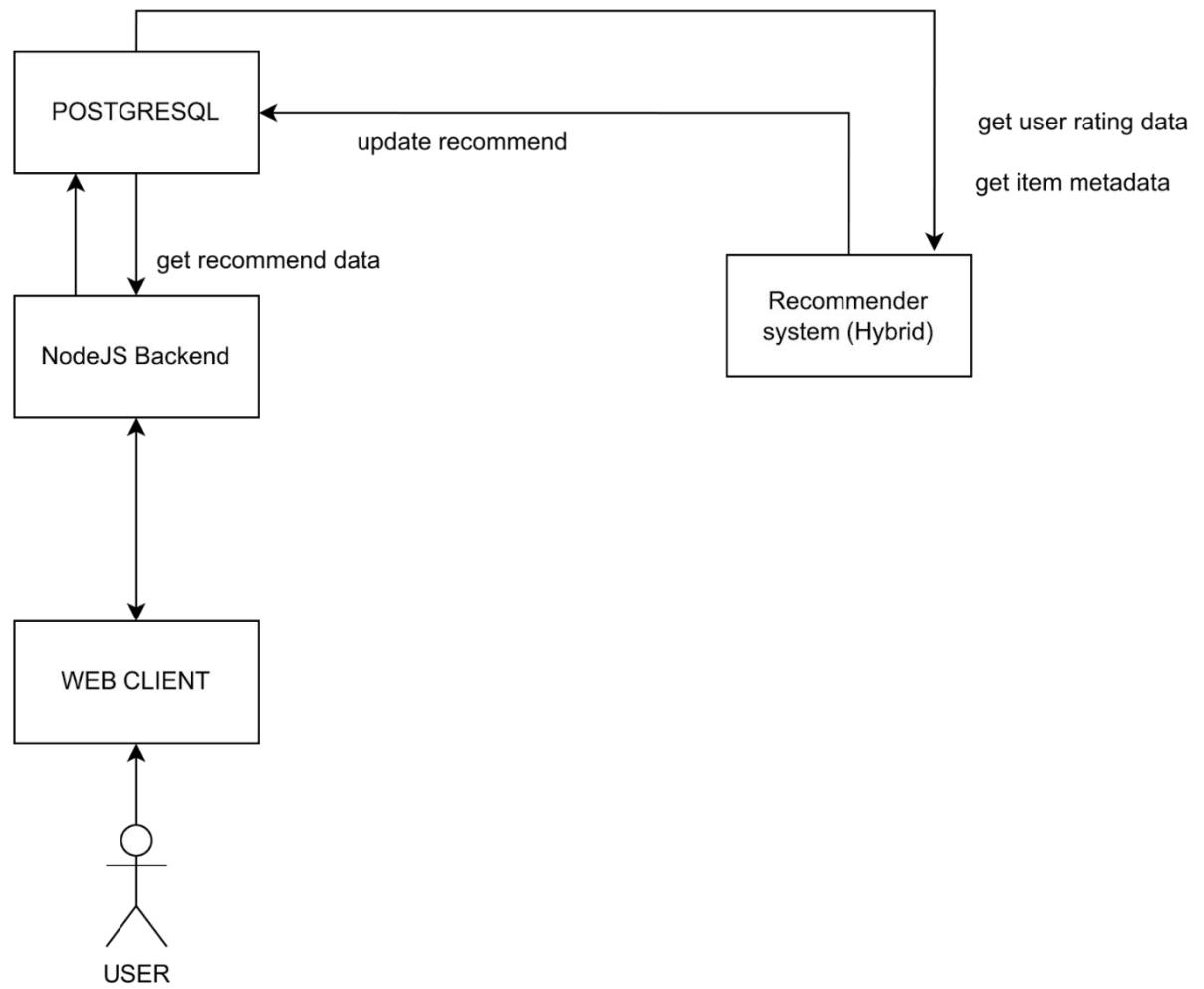
Vì vậy, ta sẽ thực hiện tích hợp Recommender System cho phần “Khóa học gợi ý” này từ dữ liệu của “Đánh giá khóa học”.

- Ý tưởng

Phương pháp recommend được chọn: Hybrid (kết hợp giữa RBM và Content-based filtering).

Lý do là bởi vì trang web hiện tại có quá ít người dùng do mới triển khai trong chưa đầy 1 tháng, với số lượng user <50 người. Do đó, việc áp dụng thuận các phương pháp thuộc Collaborative filtering rất khó khăn do mắc phải cold-start problem.

Tuy nhiên, ta có thể giải quyết vấn đề này bằng cách kết hợp Collaborative filtering (cụ thể là RBM) với Content based-filtering (cụ thể là Content KNN). Nhờ đó, hệ khuyến nghị vẫn có thể đề xuất khóa học cho người dùng mặc dù chưa có dữ liệu nhờ vào metadata của khóa học.



Ảnh: mô tả luồng hoạt động của Eduhub Recommender system

Các thành phần chính:

- Web client: frontend nơi người dùng cuối tương tác
- NodeJS Backend: backend xử lý các request và truy cập database để trả ra recommender data cho người dùng.
- PostgreSQL: cơ sở dữ liệu chính, lưu các dữ liệu của Eduhub và cả recommender data.

- Recommender System (Hybrid): một script viết bằng Python, sẽ được execute bởi một cron job mỗi 1 tiếng (có thể thay đổi tùy thuộc vào ý muốn của người triển khai).

Mô tả data:

Bảng course_ratings:

course_ratings	
create_at	timestamp
update_at	timestamp
create_by	varchar
update_by	varchar
delete_at	timestamp
course_id	uuid
rating_point	integer
comment	varchar
student_id	uuid
liked	integer
unliked	integer
id	uuid

Chúng ta sẽ sử dụng các trường: course_id (mã khóa học), rating_point (điểm đánh giá từ 0-5), student_id (mã học viên tạo đánh giá), create_at (thời điểm tạo đánh giá).

Bảng courses:

courses	
create_at	timestamp
update_at	timestamp
create_by	varchar
update_by	varchar
delete_at	timestamp
name	varchar(150)
thumbnail	text
duration	numeric(10,1)
difficulty_level	courses_difficulty_level_enum
start_date	date
end_date	date
category_id	uuid
lecturer_id	uuid
is_approved	boolean
name_en	varchar(150)
original_price	numeric
sell_price	numeric
short_description	text
introduction	text
participants	text
course_targets	text
welcome_join	varchar(100)
video_sale	text
course_materials	text
lowest_price	numeric
socialGroupLink	varchar(100)
courseLink	varchar(100)
tags	text
is_free_course	boolean
start_free_date	date
end_free_date	date
status	courses_status_enum
total_students	integer
total_reviews	integer
average_rating	integer
id	uuid

Đây là bảng chứa toàn bộ thông tin của một khóa học, ta sẽ sử dụng các cột sau:

- Id: mã khóa học
- Name: Tên khóa học
- Difficulty_level: độ khó khóa học (easy, medium, hard)
- Category_id: mã danh mục khóa học
- Lecturer_id: mã giảng viên

Bảng course_recommendations:

course_recommendations	
create_at	timestamp
update_at	timestamp
create_by	varchar
update_by	varchar
delete_at	timestamp
courses	text
student_id	uuid

Bảng này sẽ lưu recommend course cho từng user. Với cột courses là một mảng string chứa mảng các course_id.

Ý tưởng là ta sẽ lấy dữ liệu từ bảng course_rating để train cho RBM, và dữ liệu metadata của khóa học từ bảng courses để thực hiện Content based filtering.

Luồng thực hiện:

1. User tương tác với Web Client
2. Web client gửi dữ liệu tương tác (thêm đánh giá mới, xem danh sách gợi ý,...) đến Backend
3. Backend nhận gợi ý mới và thêm vào bảng course_ratings
4. Recommender system chạy cron job mỗi 1 giờ để lấy dữ liệu từ course_ratings và thực hiện khuyến nghị trên tập dữ liệu, sau đó sẽ cập nhật lại dữ liệu vào bảng course_recommend.
5. Backend lấy dữ liệu từ bảng course_recommend và trả cho client.

- **Triển khai**

Đầu tiên ta cần xây dựng class CourseRecommendation, class này chịu trách nhiệm giao tiếp với database PostgreSQL và thực hiện các bước tiền xử lý dữ liệu, convert data ra dạng có thể đọc được bởi các thư viện recommend.

```
class CourseRecommendation:
    courseID_to_name = {}
    name_to_courseID = {}
    courseID_to_details = {} # Store details: difficulty_level, category_id, lecturer_id

    def __init__(self):
        self.db_connection = DatabaseConnection()

    def _connect_db(self): ...

    def loadCourseData(self): ...

    def getUserRatings(self, user): ...

    def getPopularityRanks(self): ...

    def getCourseName(self, courseID): ...

    def getCourseID(self, courseName): ...

    def getDifficultyLevel(self, courseID): ...

    def getCategoryID(self, courseID): ...

    def getLecturerID(self, courseID): ...

    # Save recommend course ids to course_recommendations table
    def saveRecommendations(self, user, courses): ...

    def saveRecommendForEveryUser(self, recommendForEveryUser): ...

    # Load all users that have ratings
    def loadUsers(self): ...
```

Tiếp đến ta định nghĩa 2 thuật toán RBM và ContentKNN (2 class này đã được xây dựng ở phần thực nghiệm RBM và Content based filtering, giờ đây ta có thể tái sử dụng lại):

```
#Simple RBM  
SimpleRBM = RBMAgorithm(epochs=40)  
#Content  
ContentKNN = ContentKNNAlgorithm(10, {}, coursesData)
```

Tiếp đến, ta sẽ kết hợp chúng lại, với trọng số của RBM là 0.2 và ContentKNN là 0.8. Điều này có nghĩa chúng ta sẽ chú trọng vào kết quả của ContentKNN hơn vì tình trạng web hiện tại đang có quá ít dữ liệu để thực hiện RBM hiệu quả.

```
1 Hybrid = HybridAlgorithm([SimpleRBM, ContentKNN], [0.2, 0.8])
```

Cuối cùng, ta thực hiện recommend cho mọi user trong hệ thống và update lại nó xuống database:

```
1 recommendForEveryUser = evaluator.RecommendForEachUser(coursesData, users)
```

- **Kết quả**

Algorithm	RMSE	MAE	HR	cHR	ARHR	Coverage	Diversity	Novelty
Hybrid	0.7559	0.5714	0	0	0	0.35	0.8476	8.0143

Điểm số thu được, đặc biệt là RMSE và MSE cho kết quả rất tốt.

Top N result:



Search icon


Thịnh Văn Hà 1
★★★★★ 11:51 18/12/2024

Edit icon

html css

Edit icon

Khóa học gợi ý



C# Basics #1
HowKteam.com

Lập trình C# cơ bản

Tác giả

4 ★★★★★ (2 đánh giá)

đ0



**node & Express
FULL COURSE**

Lập trình NodeJS Backend
cho người mới bắt đầu

Tác giả

3 ★★★★★ (3 đánh giá)

đ100,000



f8
Javascript {Nâng cao}
fullstack.edu.vn

Javascript nâng cao

Tác giả

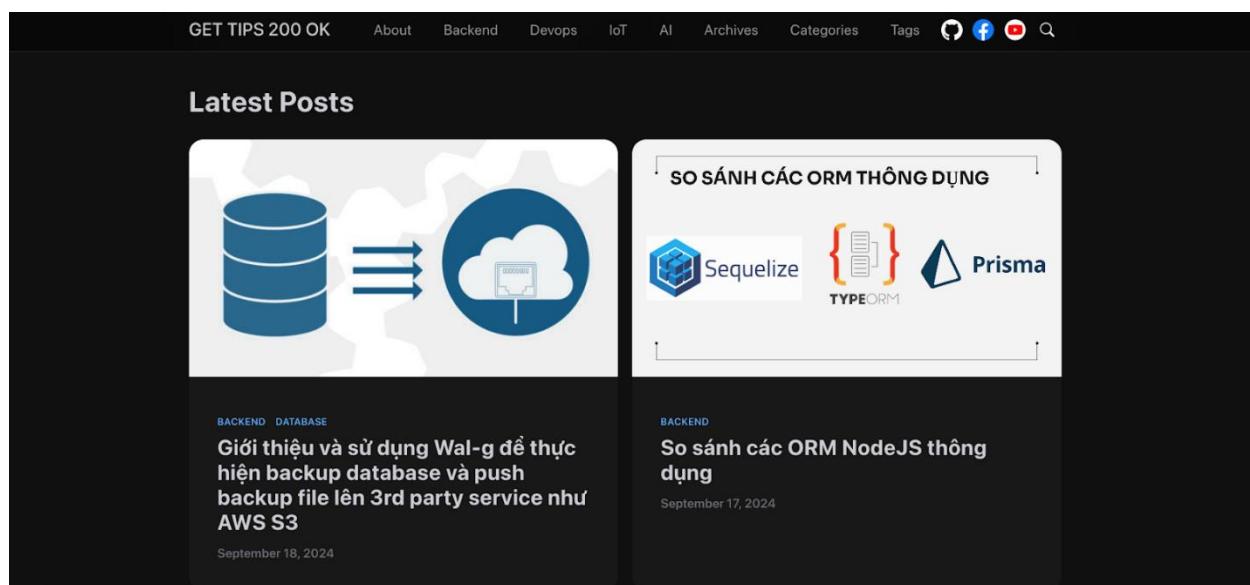
3 ★★★★★ (1 đánh giá)

đ100,000

3.2.2. Ứng dụng blog cá nhân

- Mô tả ứng dụng

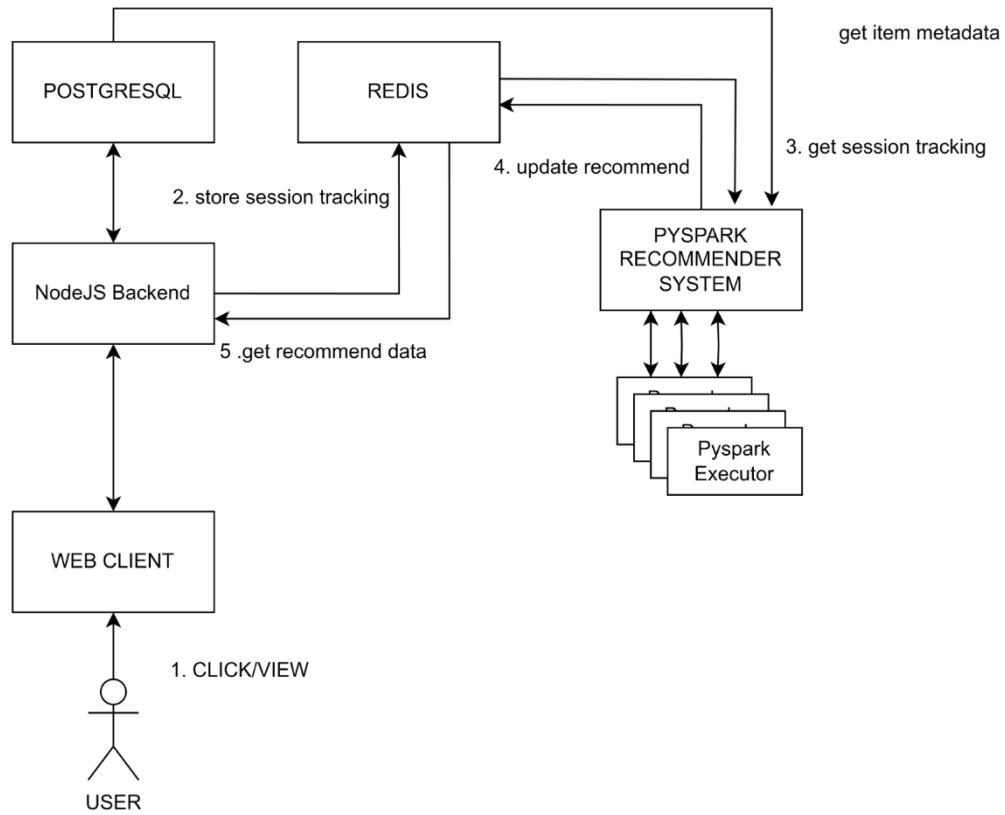
GET TIPS 200 OK là một trang blog cá nhân, với các bài viết được đăng bởi chủ website. Trang blog này hiện tại chưa có chức năng quản lý người dùng, tức là chưa có tính năng tạo tài khoản, đăng nhập tài khoản người đọc.



Vậy thách thức đặt ra là, nếu hệ thống không có dữ liệu về người dùng (vì không có chức năng đăng nhập), vậy ta sẽ tích hợp recommender system như thế nào?

- Ý tưởng

Đối với những ứng dụng không có quản lý người dùng, ta có thể tracking lại session của người truy cập web mà không cần người dùng phải đăng nhập. Cụ thể, mỗi khi có người dùng truy cập web, ta sẽ thực hiện ghi nhận lại những tương tác của người dùng đó với web ví dụ như click, view, scroll,... Tất cả những dữ liệu này hoàn toàn có thể dùng để xây dựng được một recommender system tốt.



Ảnh: sơ đồ mô tả luồng xử lý

Các thành phần chính:

- Web client: frontend nơi người dùng cuối tương tác
- NodeJS Backend: backend xử lý các request và truy cập database để trả ra recommender data cho người dùng.
- PostgreSQL: cơ sở dữ liệu chính (lưu metadata của bài đăng).
- PySpark Recommender System: sử dụng PySpark để xử lý data song song, giúp tối ưu hiệu suất, nó sẽ đưa data cho các Pyspark executor xử lý.
- Redis: lưu session tracking data và recommender data. Chọn Redis để lưu những dữ liệu này bởi vì các session data như view/click luôn được thực hiện liên tục, nếu như sử dụng các database đơn thuần như PostgreSQL sẽ

gây nghẽn hiệu suất. Do đó Redis (một cơ sở dữ liệu lưu data vào RAM) là một lựa chọn hợp lý.

Luồng xử lý sẽ như sau:

1. User tương tác (click/view) với Web client, Web client sẽ giao tiếp với Backend để gửi session tracking data
2. Backend sẽ lưu session tracking data vào Redis.
3. PySpark recommender system sẽ thu thập session tracking data từ Redis, sau đó nó sẽ thực hiện tạo recommend mới
4. Pyspark recommender system sẽ cập nhật recommend mới vào session recommend trong Redis.
5. Backend lấy recommend data và trả về cho client.

Mô tả data:

Bảng post:

post		
content	text	
tags	text	
categories	text	
title	text	
date	timestamp	
views	integer	
id	varchar(150)	

Bảng này sẽ chứa các metadata của bài viết, ở đây ta sẽ chú ý đến id (mã bài), title, date (thời điểm đăng bài) và views (số lượt xem bài viết).

Cấu trúc trong Redis:

Schema SESSION_KEY:

	SESSION_KEY	Length	TTL
1	2c14f6e5-a5df-4473-8c88-64c130c1dc90	36	134 B
2	2e7942f8-9a95-4191-bca3-a929dab3fd2	36	134 B
3	3a6b221b-c16e-47f0-9e19-b10a3bc16e07	36	134 B
4	3d09f97c-cc31-41e4-b8d7-c4b67749eac9	36	134 B
5	3e6eba06-d3f4-47d8-8fde-c14812bdf2dd	36	134 B
6	3e97310f-b0b9-43bc-9fc0-2dea312a6581	36	134 B
7	04c368c9-5fb0-4fb6-9fac-ce7b81ce9534	36	134 B
8	6a0a82dc-246c-411a-aff7-fa103e896c35	36	134 B
9	6a57bcd9-5116-43c4-ad62-dffcd3ac7f5f6	36	134 B
10	6b29b9c2-57dd-4002-8af5-c20b7ec27c4e	36	134 B
11	6b3996ef-d43e-48ed-8595-c2dae8dfa341	36	134 B
12	7b75d96a-c1c6-4ad4-a051-e4941c252fd1	36	134 B

Schema này có nhiệm vụ lưu một session id duy nhất trong Redis, tránh trường hợp bị trùng session id.

Schema Tracking:

	TRACKING	Length	TTL
1	2c14f6e5-a5df-4473-8c88-64c130c1dc90	No limit	288 B
2	3a6b221b-c16e-47f0-9e19-b10a3bc16e07	No limit	10 KB
3	22f792fa-3265-4d3d-8409-7fe8cba0f223	No limit	1 KB
4	47abc925-025b-4d29-9a96-568613da549b	No limit	4 KB
5	403cee32-53ff-48c8-b1fa-d7b844375f5d	No limit	256 B
6	49625t7-1a9c-4d7c-96be-52cc88d43813	No limit	352 B
7	965405a3-8da9-4c00-a8ea-c347b8017014	No limit	992 B
8	9378998f-ec0f-4d96-9ad2-b711cdefad8e	No limit	12 KB
9	b0a5bed6-7508-4989-87d8-7b644ee54dd0	No limit	224 B
10	bef7f64-b0e0-472e-b751-9ea6e3f5a2bf	No limit	224 B
11	c39842cf-1a9d-47d7-b2c6-4391764eb4da	No limit	608 B
12	ea8944f0-74b7-4648-bdbb-0929967eea6a	No limit	224 B
13	ebfcf436-7e83-49b2-acaa-72ff2cb24171	No limit	2 KB

Schema này có nhiệm vụ lưu lại sesion tracking data của người dùng, với cấu trúc như sau:

- Nếu là action click:

```
{  
    "type": "click",  
    "post_id": "/2024/01/27/Tips-choose-linux-distro/",  
    "timestamp": "2024-12-23T10:17:59.824Z"  
},
```

Trong đó:

- type: là loại action (click/view)
- post_id: mã bài đăng
- timestamp: thời điểm tracking
- Nếu là action view:

```
{  
    "type": "view",  
    "post_id": "/2024/05/01/bien-dien-thoai-thanh-sms-gateway-tan-dung-  
    lam-service-send-otp/",  
    "viewTime": 9,  
    "timestamp": "2024-12-23T10:17:56.150Z"  
},
```

Trong đó:

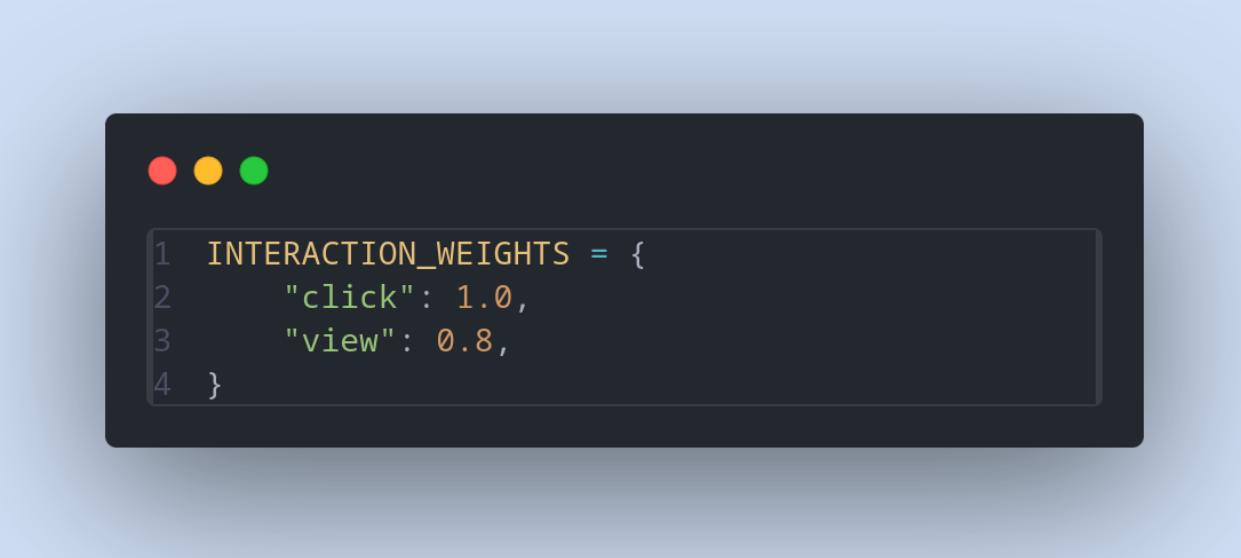
- type: là loại action (click/view)
- post_id: mã bài đăng
- timestamp: thời điểm tracking

- viewTime: thời gian user ở lại trong trang để xem bài viết này (đơn vị là giây).
- **Triển khai**

Ta sẽ sử dụng thuật toán ALS do PySpark cung cấp để triển khai recommender system cho ứng dụng này.

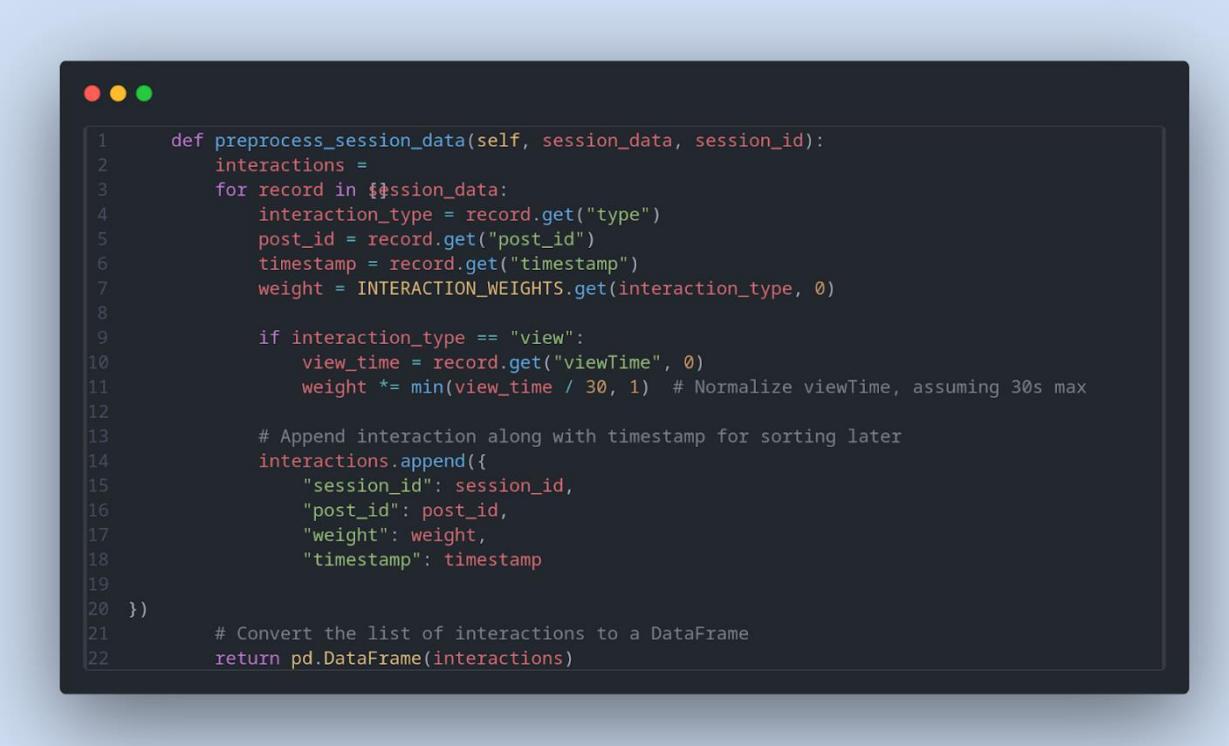
Thông thường với dạng session-based recommender system như thế này thì việc lựa chọn các mô hình RNN, ví dụ như GR4Rec (đã trình bày ở phần cơ sở lý thuyết) là hợp lý. Tuy nhiên việc áp dụng GRU4Rec thực tế rất phức tạp và khi chạy cũng tốt rất nhiều tài nguyên do đặc thù của mạng RNN. Do đó ở khuôn khổ đề tài này chúng ta sẽ áp dụng ALS, một kỹ thuật thuộc Matrix Factorization để giải quyết bài toán.

Ta sẽ xem giá trị “viewTime” hoặc số lần click thu thập được từ session tracking data như là “rating” trong một bài toán Matrix Factorization thông thường, chỉ khác ở đây ta sẽ gán cho nó những trọng số khác nhau tùy thuộc vào mức độ quan trọng:



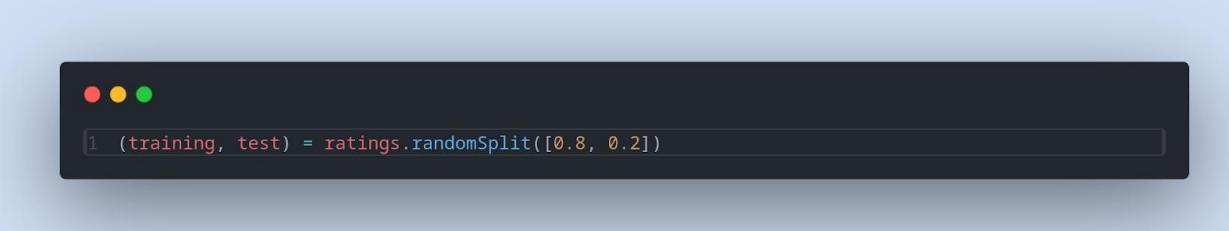
```
1 INTERACTION_WEIGHTS = {
2     "click": 1.0,
3     "view": 0.8,
4 }
```

Ở đây ta đang gán cho trọng số của click là 1.0 và view là 0.8. Việc để view là 0.8 là hợp lý vì thực tế ta sẽ còn nhân trọng số của view với viewTime để ra được kết quả.



```
1  def preprocess_session_data(self, session_data, session_id):
2      interactions =
3      for record in session_data:
4          interaction_type = record.get("type")
5          post_id = record.get("post_id")
6          timestamp = record.get("timestamp")
7          weight = INTERACTION_WEIGHTS.get(interaction_type, 0)
8
9          if interaction_type == "view":
10              view_time = record.get("viewTime", 0)
11              weight *= min(view_time / 30, 1) # Normalize viewTime, assuming 30s max
12
13      # Append interaction along with timestamp for sorting later
14      interactions.append({
15          "session_id": session_id,
16          "post_id": post_id,
17          "weight": weight,
18          "timestamp": timestamp
19      })
20  })
21  # Convert the list of interactions to a DataFrame
22  return pd.DataFrame(interactions)
```

Ở đây ta triển khai một hàm làm nhiệm vụ tiền xử lý dữ liệu: thực hiện tính toán weight cho hợp lý, nếu là click thì nó sẽ bằng đúng trọng số, còn nếu là view thì bằng trọng số x viewTime.



```
1 (training, test) = ratings.randomSplit([0.8, 0.2])
```

Tiếp đến ta thực hiện chia training và test theo tỉ lệ 80 và 20.

```

● ● ●
1     als = ALS(maxIter=5, regParam=0.01, userCol="session_id_index", itemCol="post_id_index",
2                 ratingCol="weight",
3                 coldStartStrategy="drop")
4     model = als.fit(training)

```

Sau đó ta thực hiện định nghĩa ALS với maxIter=5, regParam=0.01.

```

● ● ●
1     # Save the recommend for all user to redis
2     for row in userRecs.collect():
3         recommendations = row['recommendations']
4
5         # Create a list to hold the post IDs
6         post_ids =
7         for rec in recommendations:
8             original_post_id = ratings.filter(ratings['post_id_index'] == rec['post_id_index']).select(
9                 'post_id').collect()[0]['post_id']
10            post_ids.append(original_post_id)
11
12            original_session_id = ratings.filter(ratings['session_id_index'] == row.session_id_index).select(
13                'session_id').collect()[0]['session_id']
14
15            postLoad.save_data_to_redis(key = original_session_id, values= post_ids)

```

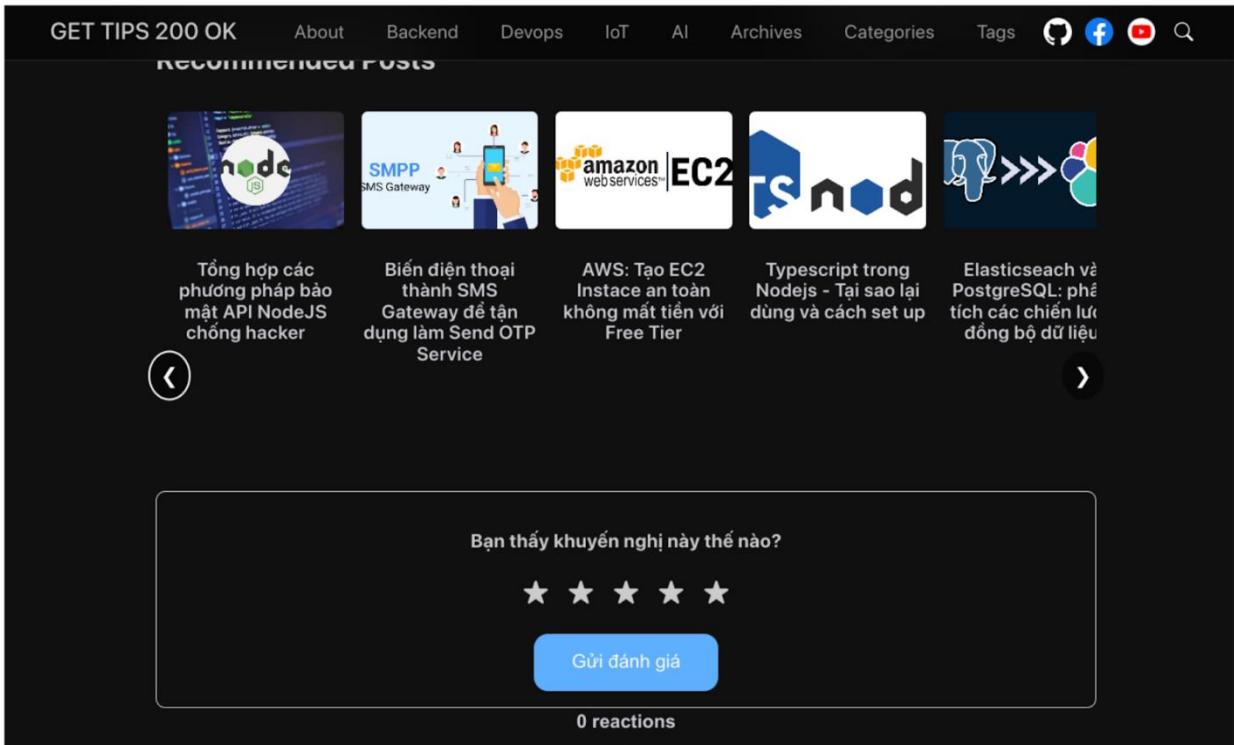
Cuối cùng ta thực hiện tính recommend và cập nhật recommend lại vào redis.

- **Kết quả**

RMSE	MAE	HR	cHR	ARHR	Coverage	Diversity	Novelty
0.2496562	0.1554869	0.1333333	0.1333333	0.06107473	0.80952	0.14167	2

Kết quả thu được có điểm số RMSE và MAE tốt.

Đây là kết quả của top N result:



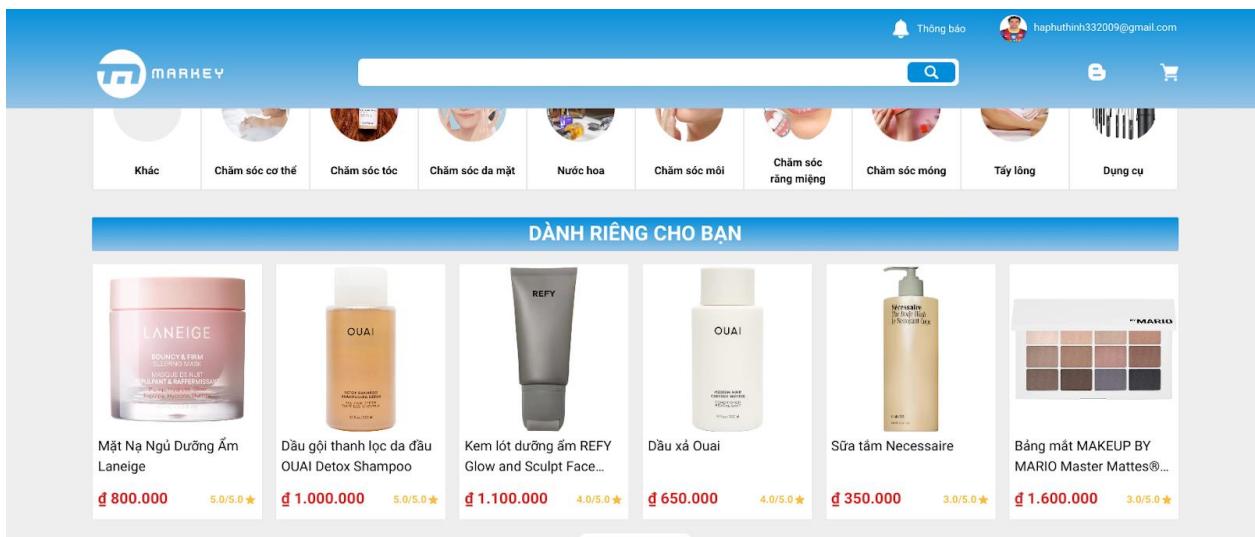
3.2.3. Ứng dụng thương mại điện tử

- Mô tả ứng dụng

Market website là một trang web bán mỹ phẩm, cho phép kết nối giữa người mua và người bán. Người bán có thể tạo tài khoản và đăng sản phẩm, trong khi người dùng có thể mua những sản phẩm họ muốn.

Cũng giống như mọi website thương mại điện tử, Markey cũng có phần “Sản phẩm gợi ý” và “Đánh giá sản phẩm”:

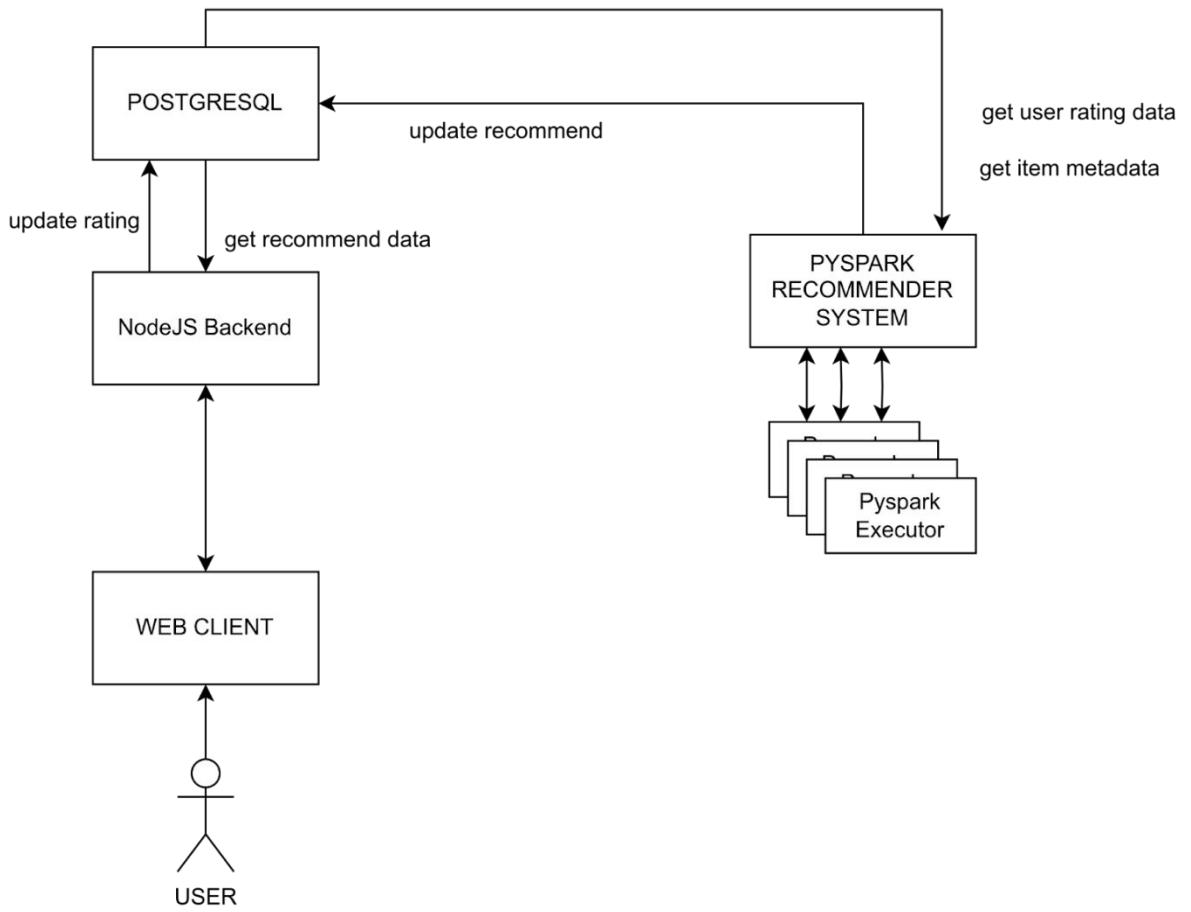
Ảnh: phần đánh giá sản phẩm



Ảnh: phần sản phẩm gợi ý

Như vậy chúng ta hoàn toàn có thể thu thập dữ liệu đánh giá sản phẩm từ người dùng để xây dựng recommender system cho phần sản phẩm gợi ý.

- Ý tưởng



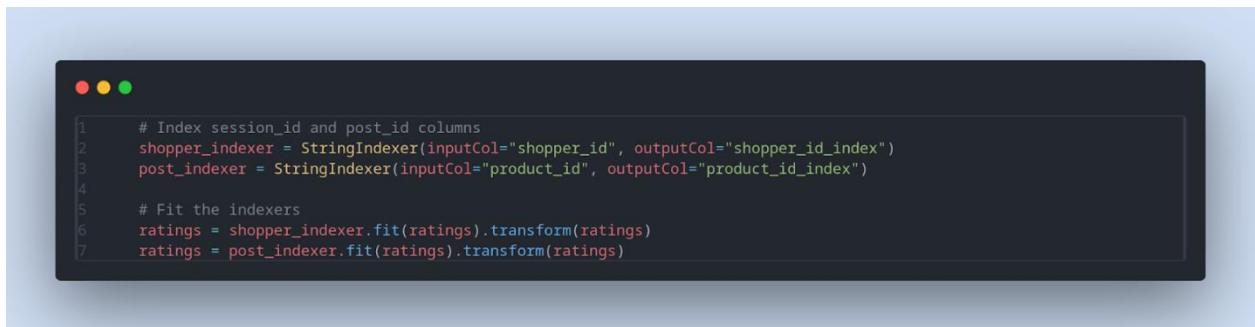
Các thành phần chính:

- Web client: frontend nơi người dùng cuối tương tác
- NodeJS Backend: backend xử lý các request và truy cập database để trả ra recommender data cho người dùng.
- PostgreSQL: cơ sở dữ liệu chính (lưu metadata của bài đăng).
- PySpark Recommender System: sử dụng PySpark để xử lý data song song, giúp tối ưu hiệu suất, nó sẽ đưa data cho các Pyspark executor xử lý. Lựa chọn sử dụng Spark bởi vì nó sẽ thích hợp cho các hệ thống ecommerce khi lượng người dùng ngày càng tăng, cần scaling sau này.

- Triển khai

Tương tự như khi triển khai với website GET TIPS 200 OK đã trình bày trước đó, ta sẽ sử dụng ALS với PySpark.

Đầu tiên, ta cần convert shopper id và post id thành kiểu số, bởi vì Spark ALS không chấp nhận id kiểu string:



```
# Index session_id and post_id columns
shopper_indexer = StringIndexer(inputCol="shopper_id", outputCol="shopper_id_index")
post_indexer = StringIndexer(inputCol="product_id", outputCol="product_id_index")
# Fit the indexers
ratings = shopper_indexer.fit(ratings).transform(ratings)
ratings = post_indexer.fit(ratings).transform(ratings)
```

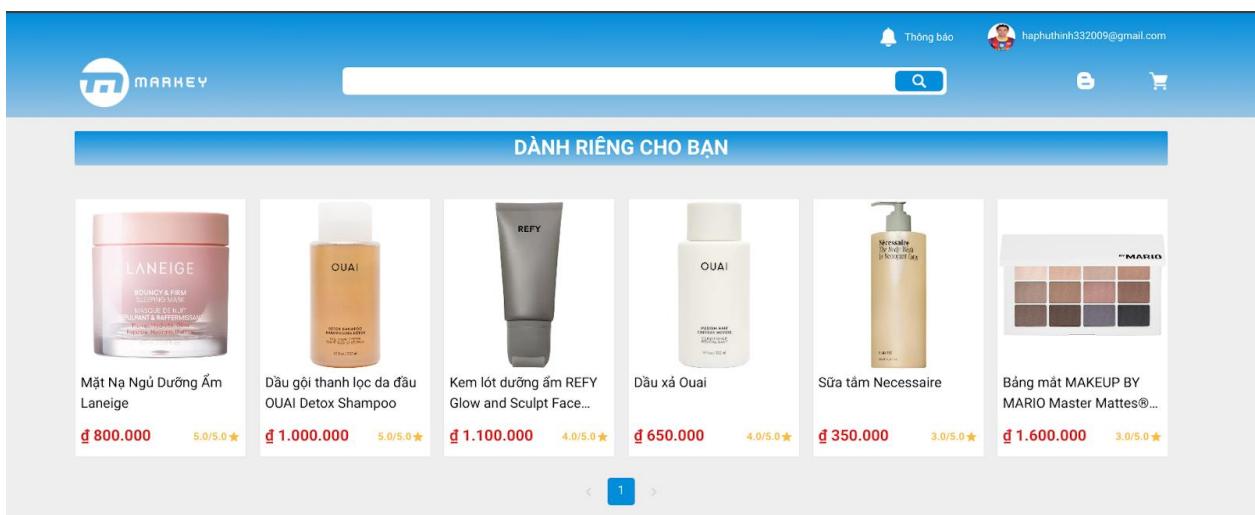
Tiếp đến, ta thực hiện chia training và test với tỉ lệ 80 20. Sau đó tạo model ALS với maxIter=5, regParam=0.01 và coldStartStrategy= drop.



```
(training, test) = ratings.randomSplit([0.8, 0.2])
als = ALS(maxIter=5, regParam=0.01, userCol="shopper_id_index", itemCol="product_id_index", ratingCol="rating",
          coldStartStrategy="drop")
model = als.fit(training)
```

- Kết quả

Top N result:



DÀNH RIÊNG CHO BẠN

Mặt Nạ Ngủ Dưỡng Ẩm Laneige	Dầu gội thanh lọc da đầu OUAI Detox Shampoo	Kem lót dưỡng ẩm REFY Glow and Sculpt Face...	Dầu xả Ouai	Sữa tắm Necessaire	Bảng mắt MAKEUP BY MARIO Master Mattes®...
đ 800.000 5.0/5.0 ★	đ 1.000.000 5.0/5.0 ★	đ 1.100.000 4.0/5.0 ★	đ 650.000 4.0/5.0 ★	đ 350.000 3.0/5.0 ★	đ 1.600.000 3.0/5.0 ★

4. KẾT LUẬN

4.1. Kết quả đạt được

- Qua các thử nghiệm thực tế, nhóm đã so sánh hiệu quả của các phương pháp như Content-based filtering, Neighborhood-based collaborative filtering, Matrix factorization, Deep learning và Hybrid method. Các phương pháp này cho thấy khả năng khuyến nghị khá tốt, với sự khác biệt rõ rệt về độ chính xác và tốc độ thực hiện.
- Việc xây dựng các ứng dụng minh họa (mua bán khóa học online, blog cá nhân và thương mại điện tử) đã chứng minh tính khả thi và hiệu quả của các phương pháp khuyến nghị trong các hệ thống thực tế. Các ứng dụng này không chỉ mang lại những khuyến nghị chính xác cho người dùng mà còn giúp nâng cao trải nghiệm người dùng trên nền tảng.

4.2. Hạn chế

- Mặc dù các phương pháp đã được thử nghiệm và cho kết quả tốt, nhưng vẫn còn một số hạn chế về độ chính xác khi áp dụng vào các hệ thống thực tế, đặc biệt là đối với những dữ liệu có tính chất phân tán hoặc không đầy đủ.
- Các phương pháp phức tạp như Deep learning hoặc Hybrid method đòi hỏi lượng tài nguyên tính toán lớn, điều này có thể gây khó khăn trong việc triển khai trên môi trường có giới hạn tài nguyên.
- Các hệ thống khuyến nghị còn khá nhạy cảm với dữ liệu thiếu hoặc dữ liệu có lỗi, điều này ảnh hưởng đến chất lượng khuyến nghị và có thể dẫn đến kết quả không chính xác.

4.3. Hướng phát triển

- Trong tương lai, cần tiếp tục nghiên cứu và cải tiến các thuật toán để tăng độ chính xác và khả năng ứng dụng rộng rãi, đặc biệt là trong các hệ thống với dữ liệu không đầy đủ hoặc phân tán.

- Tìm hiểu thêm các phương pháp học sâu (Deep learning) để cải thiện khả năng khuyến nghị trong những hệ thống phức tạp, giúp các mô hình nhận diện được các yếu tố tiềm ẩn trong dữ liệu người dùng.
- Cần nghiên cứu các phương pháp hiệu quả hơn để xử lý các vấn đề liên quan đến dữ liệu thiếu, sai sót hoặc không chính xác trong các hệ thống khuyến nghị.

5. TÀI LIỆU THAM KHẢO

[1] Recommender Systems: The Textbook (Charu C. Aggarwal, 2016)

[2] Khóa học **Building Recommender Systems with Machine Learning and AI** của Udemy: [Building Recommender Systems with Machine Learning and AI](https://www.udemy.com/building-recommender-systems-with-machine-learning-and-ai/)

[3] Bài viết về Content-based Recommender System:

<https://machinelearningcoban.com/2017/05/17/contentbasedrecordersys/>

[4] Bài viết về Content-based Recommender System:

<https://viblo.asia/p/tim-hieu-ve-content-based-filtering-phuong-phap-goi-y-dua-theo-noi-dung-phan-1-V3m5WGBg5O7>

[5] Bài viết về Neighborhood-Based Collaborative Filtering

<https://machinelearningcoban.com/2017/05/24/collaborativefiltering>

[6] Bài viết về Matrix Factorization Collaborative Filtering:

<https://viblo.asia/p/tim-hieu-ve-content-based-filtering-phuong-phap-goi-y-dua-theo-noi-dung-phan-1-V3m5WGBg5O7>

[7] Bài viết về Neghborhood based Collaborative Filtering:

<https://medium.com/fnplus/neighbourhood-based-collaborative-filtering-4b7caedd2d11>

[8] Bài viết về Neghborhood based Collaborative Filtering :

<https://medium.com/@aisagescribe/neighborhood-based-collaborative-filtering-1cb5dc0d3d8b>

[9] Tài liệu về Case Study Amazon:

<https://ieeexplore.ieee.org/abstract/document/1167344>

[10] Tài liệu về Case Study Netflix:

<https://pantelis.github.io/cs301/docs/common/lectures/recommenders/netflix/>

[11] Tài liệu về RBM: <https://vikas2pain.medium.com/restricted-boltzmann-machine-as-collaborative-filtering-28f22adf5155>

[12] Tài liệu về Autorec: <https://medium.com/@hwangdb/pytorch-implementation-of-autoencoder-based-recommender-system-9aff6c3d1b02>

[13] Tài liệu về Deep Learning For Recommender System:

<https://medium.com/sciforce/deep-learning-based-recommender-systems-b61a5ddd5456>