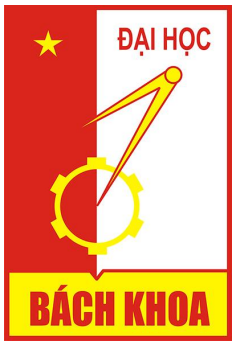
ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



ĐỒ ÁN TỐT NGHIỆP

Thử nghiệm và đánh giá mô hình hệ thống gợi ý

LƯƠNG TUẤN LINH
linh.lt152186@sis.hust.edu.vn

Ngành Công nghệ thông tin

Giảng viên hướng dẫn : PGS.TS Thân Quang Khoát

Chữ ký của GVHD

Bộ môn : Hệ thống thông tin
Viện : Công nghệ Thông tin và Truyền thông

Hà Nội, tháng 6 năm 2020

Thẻ nhiệm vụ tốt nghiệp

Thông tin sinh viên

- **Họ và tên:** Lương Tuấn Linh
- **Email:** linh.lt152186@sis.hust.edu.vn
- **Số điện thoại:** 0912201718
- **Lớp học:** IS1 K60
- **Chương trình:** Kỹ Sư
- **Thời hạn:** Từ tháng 1 năm 2019 đến tháng 6 năm 2020

Mục tiêu chính của đồ án

1. Nghiên cứu về phương pháp xây dựng hệ thống gợi ý
2. Áp dụng các phương pháp xây dựng hệ thống gợi ý và so sánh kết quả

Nhiệm vụ cụ thể của đồ án

1. Tìm hiểu, nghiên cứu và thực hành các phương pháp xây dựng hệ thống gợi ý như content-based, collaborative filtering. Đặc biệt tìm hiểu kĩ về các phương pháp nằm trong nhóm collaborative filtering như neighbor-based, matrix factorization và clustering.
2. Áp dụng 3 phương pháp gợi ý của collaborative fitering là neighbor-based, matrix factorization và clustering vào để giải bài toán xây dựng hệ thống gợi ý sách cho người dùng.
3. Xây dựng các kịch bản thực nghiệm cho mỗi phương pháp, phân tích kết quả, sau đó so sánh điểm mạnh và điểm yếu của mỗi phương pháp được dùng trong thực nghiệm.

Cam kết của sinh viên

Tôi là *Lương Tuấn Linh* cam đoan rằng nội dung trong đồ án này là của tôi dưới sự hướng dẫn của PGS.TS Thân Quang Khoát.

Các đề xuất và kết quả trong đồ án này đều là xác thực và nguyên bản. ·

Hà Nội, Ngày 15 Tháng 6 năm 2020

Tác giả đồ án

Lương Tuấn Linh

Xác nhận về mức độ hoàn thiện đồ án và cho phép đồ án được bảo vệ bởi giáo viên hướng dẫn

. . . . . .
. . . . . .
. . . . . .
. . . . . .
·

Hà Nội, Ngày 15 Tháng 6 năm 2020

Người hướng dẫn

PGS.TS Thân Quang Khoát

Lời cảm ơn

Trong thời gian nghiên cứu đồ án, em đã nhận được nhiều sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của thầy cô và bạn bè.

Em xin gửi lời cảm ơn chân thành đến thầy PGS.TS Thân Quang Khoát - trường Đại học Bách Khoa Hà Nội người đã tận tình hướng dẫn, chỉ bảo em trong suốt quá trình học tập, nghiên cứu đồ án.

Em cũng xin chân thành cảm ơn các thầy cô giáo trong trường Đại học Bách Khoa Hà Nội nói chung, các thầy cô giáo trong Viện Công nghệ Thông Tin và Truyền thông nói riêng đã dạy dỗ cho em kiến thức về các môn học đại cương cũng như các môn chuyên ngành, giúp em có được cơ sở lý thuyết vững vàng và tạo điều kiện giúp đỡ em trong suốt quá trình học tập.

Cuối cùng, em xin chân thành cảm ơn bạn bè đã luôn hỗ trợ, giúp đỡ trong quá trình học tập và nghiên cứu.

Hà Nội, Ngày 15 tháng 4 năm 2020
Lương Tuấn Linh

Tóm tắt

Đồ án này tập trung tìm hiểu và học những kiến thức về hệ thống gợi ý (recommendation system). Hiện nay nhu cầu xây dựng hệ gợi ý ngày càng tăng. Lí do là với sự phát triển của các website thương mai điện tử, nền tảng dịch vụ online như phim ảnh, âm nhạc, việc xây dựng một hệ thống có thể hỗ trợ ra quyết định, cung cấp giải pháp mang tính cá nhân hóa mà không cần trải qua quá trình tìm kiếm phức tạp trở nên rất thiết yếu. Những kiến thức đã tìm hiểu được bao gồm những phần sau. Trong đồ án chỉ ra những vấn đề vẫn còn gặp phải trong việc phát triển hệ thống gợi ý. Đồ án trình bày tổng quan những kiến thức cơ bản về một số phương pháp xây dựng hệ thống gợi ý như content-based, collaborative và phương pháp lai. Sau đó thực nghiệm xây dựng hệ thống gợi ý sử dụng các phương pháp collaborative filtering. Cuối cùng là so sánh và phân tích kết quả.

Đồ án được xây dựng bao gồm các phần như sau:

- **Chương 1** Tổng quan các kiến thức cơ bản về các phương pháp xây dựng hệ thống gợi ý.
- **Chương 2** Giới thiệu về bài toán và đưa ra hướng tiếp cận sử dụng 3 phương pháp collaborative filtering là neighbor-based, matrix factorization và clustering vào để giải bài toán.
- **Chương 3** Trình bày kết quả thực nghiệm, phân tích, so sánh và đánh giá hiệu quả của giải thuật đã đề xuất, đưa ra kết luận.

Lời mở đầu

Hiện nay, hệ thống gợi ý là một phần vô cùng quan trọng trong hệ thống thông tin. Mục đích của hệ thống gợi ý chính là giúp cho người dùng tìm được những thông tin cần thiết, dự đoán sở thích hay xếp hạng mà người dùng có thể đánh giá cho một sản phẩm nào đó mà họ chưa xem xét tới trong quá khứ. Các gợi ý sẽ chính là kết quả được đưa ra dự trên việc thu thập dữ liệu thông tin khi người dùng mua hàng, khi đưa ra các đánh giá cá nhân. Việc thực hiện tính toán được xây dựng trên các thuật toán Machine Learning, đưa ra các dự đoán tốt nhất về sản phẩm mà người dùng có thể thích, giúp tối ưu hóa doanh thu qua up-sale, cross-sale, gợi ý các sản phẩm. Từ đó giúp cải thiện trải nghiệm người dùng, tăng hiệu năng hoạt động bằng tự động hóa, biến khách hàng tiềm năng trở thành khách hàng thật.

Một ví dụ về hệ thống gợi ý như hệ thống bán hàng Amazon, hệ thống của họ sẽ dựa vào nhưng sản phẩm mà người dùng đã mua trong quá khứ, những người dùng đã đánh giá, viết bình luận hay vẫn còn trong giỏ hàng. Hay chẳng hạn như youtube sẽ dựa vào những video mà người dùng đã xem, xem trong bao lâu, dựa vào đánh giá like hay dislike của người dùng để đưa ra gợi ý những video khác để xem. Những ví dụ trên cho thấy hệ thống gợi ý là một hệ thống thiết yếu trong thời đại công nghệ thông tin phát triển như hiện nay.

Đồ án tốt nghiệp này tập trung nghiên cứu các phương pháp giải quyết bài toán xây dựng hệ thống gợi ý sách cho người dùng. Hiện nay, những trang web hay hệ thống bán sách online ngày càng phát triển, do đó hệ thống gợi ý sách cho người dùng cũng rất thiết yếu. Có nhiều phương pháp để tiếp cận bài toán này, tuy nhiên có thể chia thành hai nhóm lớn là content-based và collaborative filtering. Phần tổng quan của đồ án sẽ nói về hai cách tiếp cận trên và những phương pháp nằm trong hai hướng tiếp cận bài toán này. Ngoài ra sau đó sẽ nói về hướng tiếp cận hybrid, lai giữa hai hay nhiều phương pháp khác nhau nhằm đưa ra được kết quả tốt hơn. Chương 2 của đồ án giới thiệu về bài toán đặt ra, môi trường phát triển, bộ dữ liệu được sử dụng và 3 phương pháp được thực nghiệm trong đồ án này, đó là neighbor-based collaborative filtering, matrix factorization và clustering. Chương 3 của đồ án bao gồm các kịch bản thực nghiệm của mỗi phương pháp và kết quả của mỗi kịch bản đó. Sau đó, các kịch bản được phân tích đánh giá và so sánh kết quả giữa các phương pháp. Chương cuối cùng đưa ra kết luận, so sánh ưu điểm, nhược điểm giữa các phương pháp, chỉ ra phương pháp nào nên được sử dụng trong tình huống nào.

Thuật Ngữ

Viết tắt	**Thuật ngữ đầy đủ**
CF	Collaborative Filtering
NBCF	Neighbor Based Collaborative Filtering
MFCF	Matrix Factorization Collaborative Filtering
ILS	Intra-List Similarity

Danh sách bảng

1.1 Content-based Utility Matrix 15

3.1 Kết quả thực nghiệm kịch bản thay đổi số người dùng tương tự user-based 42

3.2 Kết quả thực nghiệm kịch bản thay đổi số sản phẩm tương tự item-based 42

3.3 Kết quả thực nghiệm Matrix factorization collaborative filtering (MFCF) khi thay đổi số tính chất ẩn 44

3.4 Kết quả thực nghiệm MFCF khi thay đổi learning rate 45

3.5 Kết quả thực nghiệm phương pháp clustering 46

3.6 So sánh kết quả thực nghiệm của các phương pháp 47

Danh sách hình vẽ

1.1 Utility Matrix [5] 14

1.2 Mô hình tích 2 ma trận [7] 25

3.1 So sánh hit rate của hai phương pháp user-based và item-based dựa trên thời gian xây dựng danh sách gợi ý. 43

3.2 Các trường hợp tối ưu hàm mất mát theo learning rate [4] 45

Mục lục

Thẻ nhiệm vụ tốt nghiệp **2**

Lời cảm ơn **4**

Tóm tắt **5**

Lời Nói Đầu **6**

Thuật Ngữ **7**

1 Tổng quan **13**

1.1 Giới thiệu về hệ thống gợi ý 13

1.2 Utility Matrix 14

1.3 Hệ thống gợi ý content-based 15

1.3.1 Xây dựng item profile 15

1.3.2 Xây dựng hàm mất mát 16

1.4 Hệ thống gợi ý collaborative filtering 17

1.4.1 Giới thiệu 17

1.4.2 Những vấn đề gặp phải của Collaborative Filtering 18

1.4.3 Neighborhood-based Collaborative Filtering 19

1.4.4 Clustering Collaborative Filtering 22

1.4.5 Matrix factorization collaborative filtering 24

1.5 Hệ thống gợi ý Hybrid 27

1.5.1 Phương pháp hybrid sử dụng trọng số 27

1.5.2 Phương pháp hybrid chuyển đổi giữa các phương pháp (switching hybrid) 28

1.5.3 Phương pháp hybrid hỗn hợp (mixed hybrid) 28

1.5.4 Hybrid Recommender kết hợp tính chất (feature combination) 29

1.6 Đánh giá hệ thống gợi ý 29

1.6.1 Đánh giá mức độ chính xác của giá trị đánh giá dự đoán . . . 29

1.6.2 Đánh giá Top-N recommendation 30

2 Bài toán và các mô hình thực nghiệm 32

2.1 Bài toán 32

2.1.1 Đặt vấn đề 32

2.1.2 Bài toán đặt ra 32

2.2 Công cụ sử dụng 33

2.2.1 Môi trường phát triển 33

2.2.2 Bộ cơ sở dữ liệu 33

2.3 Tiếp cận sử dụng phương pháp Neighbor-based Collaborative Filtering 34

2.3.1 Dự đoán đánh giá của người dùng đối với sách 34

2.3.2 Xây dựng danh sách gợi ý N quyển sách phù hợp với người dùng 34

2.4 Tiếp cận sử dụng phương pháp Matrix Factorization Collaborative Filtering 35

2.4.1 Xây dựng mô hình 35

2.4.2 Dự đoán đánh giá của người dùng với sách 37

2.4.3 Xây dựng danh sách gợi ý N quyển sách phù hợp nhất cho người dùng 37

2.5 Tiếp cận sử dụng phương pháp Clustering Collaborative Filtering . . 37

2.5.1 Xây dựng mô hình 37

2.5.2 Dự đoán đánh giá của người dùng cho sách 39

2.5.3 Xây dựng danh sách gợi ý cho người dùng 39

3 Kết quả thực nghiệm và so sánh **40**

3.1 Thực nghiệm phương pháp neighborhood-based collaborative filtering 41

3.1.1 Kịch bản thay đổi số lượng người dùng tương tự đối với hướng tiếp cận user-based 41

3.1.2 Kịch bản thay đổi số lượng người dùng tương tự đối với hướng tiếp cận item-based 42

3.1.3 So sánh 2 hướng tiếp cận user-based và item-based 43

3.2 Thực nghiệm phương pháp matrix-factorization collaborative filtering 44

3.2.1 Kịch bản thay đổi số lượng tính chất ẩn 44

3.2.2 Kịch bản thay đổi learning rate 44

3.3 Thực nghiệm phương pháp clustering collaborative filtering 45

3.4 So sánh kết quả thực nghiệm các phương pháp 46

Kết Luận **48**

Chương 1

Tổng quan

1.1 Giới thiệu về hệ thống gợi ý

Trong một vài thập kỷ gần đây, với sự phát triển của Youtube, Amazon, Netflix và rất nhiều các web service khác, hệ thống gợi ý đang ngày càng trở nên quan trọng hơn. Từ các dịch vụ e-commerce (gợi ý cho người mua những sản phẩm mà họ có thể thích) cho tới quảng cáo online (gợi ý những quảng cáo phù hợp với từng người), hệ thống gợi ý liên tục xuất hiện trong cuộc sống hàng ngày của mỗi người.

Nói chung, hệ thống gợi ý là những thuật toán nhằm dự đoán mức độ quan tâm của một người dùng đến một sản phẩm (sản phẩm ở đây có thể là phim để xem, những bài báo để đọc, những sản phẩm đang bán hay bất kì cái gì tùy thuộc vào từng lĩnh vực). Hệ thống gợi ý còn nắm vai trò thiết yếu trong một số lĩnh vực vì có khả năng tạo ra một lượng doanh thu vô cùng lớn nhờ vào mức độ chính xác và giúp nổi bật hơn so với các đối thủ cạnh tranh. Một minh chứng cho tầm quan trọng của hệ thống gợi ý là, một vài năm về trước, Netflix đã tổ chức cuộc thi "Netflix prize" với mục đích để tìm được hệ thống gợi ý tốt hơn hệ thống gợi ý hiện có với phần thưởng là một triệu đô.

Việc xây dựng hệ thống gợi ý có rất nhiều phương pháp khác nhau, tuy nhiên có thể chia thành hai hướng tiếp cận chính. Đó là content-based và collaborative filtering. Ngoài ra còn có hướng tiếp cận hybrid, tức là phương pháp lai giữa nhiều phương pháp khác nhau nhằm đưa ra kết quả tốt hơn. Sau đây là phần giới thiệu tổng quan về những phương pháp thuộc các nhóm này.

1.2 Utility Matrix

Trong hệ thống gợi ý có 2 thực thể chính là người dùng và sản phẩm. Mỗi người dùng sẽ một độ quan tâm nhất định tới mỗi sản phẩm. Mức độ quan tâm được gán cho một giá trị ứng với mỗi cặp người dùng - sản phẩm. Giả sử giá trị này được đo bằng giá trị đánh giá của người dùng cho sản phẩm. Khi đó, tập hợp các đánh giá, bao gồm cả những giá trị chưa biết cần được dự đoán, được gọi là utility matrix.

Giả sử như ví dụ của utility matrix như hình 1.1, đánh giá của người dùng cho mỗi phim giá trị từ 1 đến 5, với 5 là rating tốt nhất. Khoảng trống thể hiện những phim mà người dùng vẫn chưa đánh giá. Những giá trị HP1, HP2, HP3 là phim Harry Potter 1, 2, 3, TW là Twilight, SW1, SW2, SW3 là Star Wars 1, 2, 3. Người dùng được thể hiện ở các chữ cái A đến D.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Hình 1.1: Utility Matrix [5]

Thông thường, có rất nhiều người dùng và sản phẩm trong hệ thống, và mỗi người dùng chỉ đánh giá một số lượng ít các sản phẩm hay không đánh giá sản phẩm nào. Do vậy trong utility matrix thường rất nhiều ô trống và những ô được điền là rất ít.Rõ ràng càng nhiều ô được điền thì độ chính xác của hệ thống sẽ càng được cải thiện. Vì vậy các hệ thống luôn hỏi người dùng về sự quan tâm của họ tới sản phẩm, và muốn người dùng đánh giá càng nhiều sản phẩm càng tốt. Việc đánh giá sản phẩm không những giúp các người dùng khác biết được chất lượng sản phẩm mà còn giúp hệ thống biết được sở thích của người dùng, qua đỡ có thể quảng cáo hợp lý.

Nếu không có utility matrix thì gần như không thể gợi ý sản phẩm cho người dùng. Do vậy việc xây dựng utility matrix là việc rất quan trọng. Có hai hướng tiếp cận phổ biến để xác định giá trị đánh giá cho mỗi cặp người dùng-sản phẩm trong utility matrix.

Hướng thứ nhất là nhờ người dùng đánh giá sản phẩm. Ví dụ như Amazon luôn nhờ người dùng đánh giá các sản phẩm của họ bằng việc gửi các email nhắc nhở. Nhiều hệ thống khác cũng áp dụng cách này. Tuy nhiên phương pháp này có một

vài hạn chế vì thường người dùng ít khi đánh giá sản phẩm. Và những đánh giá đó có thể là đánh giá thiên lệch của người dùng.

Hướng thứ hai là dựa vào hành vi của người dùng. Ví dụ, nếu một người xem một bộ phim trên netflix, hay một clip trên youtube hay là mua một sản phẩm trên Amazon chứng tỏ người dùng thích sản phẩm đó. Youtube dựa trên những video mà người dùng đã xem để gợi ý cho người dùng những video khác liên quan tới những video đã xem đó. Thường thì với phương pháp này, chỉ có thể xây dựng utility matrix với giá trị 0 và 1. Trong đó, 0 thể hiện việc chưa có thông tin và 1 thể hiện việc đã xem video (thích sản phẩm). Tuy nhiên, cũng có thể xây dựng ma trận có giá trị cao hơn 1 tùy thuộc vào số lần người dùng xem video đó hay là thời gian xem video đó. Nút dislike cũng mang lại giá trị đánh giá vì thể hiện việc người dùng không thích video, ví dụ có thể gán giá trị -1 trong utility matrix.

1.3 Hệ thống gợi ý content-based

1.3.1 Xây dựng item profile

Content-based dựa trên nội dung của các sản phẩm. Do đó để xây dựng được hệ thống gợi ý sử dụng phương pháp content-based, cần phải xây dựng được bộ profile cho các sản phẩm. Profile được biểu diễn dưới dạng toán học là các vector đặc trưng. Ví dụ như các đặc điểm của một bộ phim có thể là thể loại, diễn viên, đạo diễn, năm sản xuất,... Có nhiều đặc trưng có thể sử dụng, tuy nhiên thể loại có thể khó định nghĩa

Bảng 1.1: Content-based Utility Matrix

Movie	Alice(1)	Bob(2)	Carol(3)	Dave(4)	d_1	d_2
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Sword vs. karate	0	0	5	?	0	0.9

Như ví dụ ở bảng 1.1, có thể đơn giản hóa bằng cách xây dựng vector đặc trưng cho mỗi bộ phim. Chiều thứ nhất của vector là d_1 thể hiện mức độ lãng mạn, chiều thứ 2 là d_2 thể hiện mức độ hành động. Gọi các vector đặc trưng cho mỗi bộ phim là x_1, x_2, x_3, x_4, x_5 ta sẽ có giá trị mỗi vector là $x_1 = [0.9, 0]$, $x_2 = [1.0, 0.01]$, $x_3 = [0.99, 0]$, $x_4 = [0.1, 1.0]$, $x_5 = [0, 0.9]$.

Tương tự, sở thích của mỗi người dùng cũng có thể được mô hình hóa dưới dạng tập các tham số θ. Chẳng hạn như người dùng thích phim lãng mạn và ghét phim

hành động sẽ có vector thể hiện là $\theta_u = [1.0, 0]$. Dữ liệu huấn luyện để xây dựng các vector θ là đánh giá của người dùng và profile của các sản phẩm đã được người dùng đó đánh giá. Việc điền giá trị còn thiếu trong ma trận utility là dự đoán mức độ quan tâm của người dùng với sản phẩm bằng cách lấy tích vô hướng của vector θ thể hiện sở thích của người dùng và vector x thể hiện profile của sản phẩm. Đầu ra có thể viết dưới dạng $f(\theta, x)$.

1.3.2 Xây dựng hàm mất mát

Giả sử số người dùng là N, số phim là M. Ma trận profile là $X = [x_1, x_2, ..., x_m] \in R_{d \times M}$ (d là số feature) và ma trận utility $Y \in R_{M \times N}$ Thành phần ở hàng thứ m, cột thứ n của Y là mức độ quan tâm của người dùng thứ n lên sản phẩm m mà hệ thống thu thập được. Giả sử tìm được một mô hình cho mỗi người dùng minh họa bởi vector $W_n \in R^d$ sao cho độ quan tâm có thể được tính theo công thức:

$$y_{mn} = w_n^T x_m + b_n \tag{1.1}$$

Xét người dùng thứ n, coi tập huấn luyện là tập hợp các thành phần đã được điền của người dùng n trong ma trận utility, có thể xây dựng hàm mất mát tương tự như sau:

$$L_n(w_n, b_n) = \frac{1}{2s_n} \sum_{m:r_{mn}=1} (w^n x_m + b_n - y_{mn})^2 + \frac{\lambda}{2s_n} \|w_n\|_2^2 \tag{1.2}$$

trong đó, thành phần thứ hai là regularization và λ là tham số dương, s_n là số lượng các sản phẩm mà người dùng n đã đánh giá. $\sum_{m:r_{mn}=1}$ thể hiện những sản phẩm m đã được người dùng n tương tác.

Vì ở biểu thức trên s_n là hằng số nên tập trung vào tối ưu phương trình đơn giản hơn dưới đây.

$$\min \frac{1}{2} \sum_{m:r_{mn}=1} (w^n x_m + b_n - y_{mn})^2 + \frac{\lambda}{2} \|w_n\|_2^2 \tag{1.3}$$

Hệ thống gợi ý sử dụng phương pháp content-based có điểm mạnh là không cần có dữ liệu về những người dùng khác vì việc gợi ý chỉ tập trung vào từng người dùng. Điều này giúp cho việc mở rộng hệ thống lên một lượng người dùng lớn trở nên dễ dàng hơn. Hơn nữa, mô hình này có thể nắm bắt được sở thích của người dùng (Ví dụ thể loại phim hay xem, ca sĩ ưa thích, ...) và từ đó có thể gợi ý sản phẩm niche mà ít người dùng thích. Tuy nhiên, hệ thống gợi ý sử dụng phương pháp content-based vẫn còn một số điểm yếu như sau. Vì phương pháp cần phải có đặc

điểm của từng sản phẩm nên vẫn phụ thuộc vào độ chính xác của thông tin các đặc điểm này. Ngoài ra, mô hình này chỉ có thể gợi ý dựa trên sở thích đã biết của người dùng. Nói cách khác, mô hình này không thể gợi ý những sản phẩm nằm ngoài sở thích đã biết của người dùng.

1.4 Hệ thống gợi ý collaborative filtering

1.4.1 Giới thiệu

Hệ thống gợi ý content-based có thể xây dựng mô hình cho mỗi người dùng không phụ thuộc vào các người dùng khác mà chỉ phụ thuộc vào profile của các sản phẩm, tuy nhiên vẫn còn những nhược điểm là không tận dụng được thông tin từ các người dùng khác và có lúc không thể xây dựng được profile cho mỗi sản phẩm. Tiếp theo sẽ là phương pháp có thể giải quyết được 2 vấn đề này, đó là Collaborative Filtering (CF). Phương pháp dự đoán cơ bản của CF là nếu người dùng X và người dùng Y đánh giá n sản phẩm gần giống nhau, hay có hành vi giống nhau (cùng xem nhiều loại phim, cùng mua nhiều loại sản phẩm, ...) thì sẽ đánh giá hay tương tác (xem, mua,...) với các sản phẩm khác giống nhau. Phương pháp CF là dựa trên độ quan tâm của người dùng tới sản phẩm đã biết trước để có thể đưa ra gợi ý các sản phẩm mới mà người dùng chưa biết đến hay chưa sử dụng. Ví dụ, có danh sách gồm m người dùng $u_1, u_2, ...u_m$ và một danh sách n sản phẩm $i_1, i_2, ..., i_n$ và mỗi người dùng u_i có một danh sách các sản phẩm mà người dùng đó đã đánh giá (xem, mua, ...). Những giá trị này có thể là đánh giá, số lần mua, số lần click,... Phương pháp CF sử dụng những thông tin này để đưa ra gợi ý về sản phẩm mà người dùng vẫn chưa có tương tác.

Phương pháp CF gặp phải nhiều thử thách. Thuật toán CF cần phải có khả năng xử lý với bộ dữ liệu thưa (bộ dữ liệu mà nhiều người dùng, nhiều sản phẩm nhưng lại ít đánh giá), có thể xử lý được khi số lượng người dùng và sản phẩm tăng, đáp ứng cho khoảng thời gian ngắn, và nhiều vấn đề khác. Điểm khác biệt lớn nhất giữa hệ thống gợi ý CF và content-based là CF chỉ sử dụng đánh giá của người dùng cho sản phẩm còn content-based thì phụ thuộc vào đặc điểm của sản phẩm để dự đoán. Cả hai phương pháp đều có những giới hạn nhất định. Do vậy một số phương pháp hybrid cũng đã được phát triển như content-boosted CF hay personality diagnosis nhằm vượt qua được những giới hạn của mỗi phương pháp.

1.4.2 Những vấn đề gặp phải của Collaborative Filtering

Dữ liệu thưa: Trong thực tế, rất nhiều hệ thống gợi ý phải xử lý một lượng sản phẩm rất lớn. Utility matrix thường vô cùng thưa nên việc tối ưu hiệu năng của hệ thống gợi ý là rất quan trọng. Đối với vấn đề này, có 1 phương pháp được áp dụng là phương pháp giảm số chiều dữ liệu ví dụ như Singular Value Decomposition (SVD), loại bỏ những người dùng, sản phẩm không tiêu biểu hay không cần thiết nhằm giảm số chiều của Utility matrix.

Cold start: Khi có người dùng mới hay sản phẩm mới vào trong hệ thống, chưa từng có lịch sử đánh giá hay tương tác gì của người dùng, sản phẩm này, việc gợi ý sản phẩm phù hợp người dùng có thể trở nên khó khăn. sản phẩm mới không được gợi ý nếu không có ai đánh giá và người dùng cũng không nhận được gợi ý tốt vì chưa có lịch sử đánh giá hay mua bán gì. Giải pháp như hybrid có thể hỗ trợ vấn đề này. Với những sản phẩm chưa được đánh giá hay mua bao giờ, nếu có được các đặc điểm của sản phẩm, vẫn có thể gợi ý được cho người dùng.

Khả năng mở rộng: Khi số lượng người dùng và sản phẩm tăng quá nhanh, phương pháp CF bình thường sẽ gặp phải vấn đề scalability, yêu cầu tính toán vượt quá mức có thể đáp ứng. Ví dụ như khi có hàng chục triệu người dùng và hàng triệu sản phẩm thì độ phức tạp của thuật toán CF là $O(n)$ sẽ là quá lớn. Chưa nói đến việc các hệ thống cần phải phản hồi ngay lập tức để đưa ra gợi ý hợp lý. Phương pháp giảm số chiều dữ liệu như SVD có thể dùng để giải quyết vấn đề scalability tuy nhiên vẫn phải trải qua bước matrix factorization. Ngoài ra còn có phương pháp khác như clustering, phương pháp này giải quyết vấn đề bằng cách tìm người dùng để đánh giá trong 1 cụm nhỏ có độ tương đồng cao thay vì trên toàn bộ cơ sở dữ liệu.

Gray Sheep: Gray sheep nói đến những người dùng có quan điểm không nhất quán, không tương đồng với nhóm người dùng nào nên không có nhiều giá trị đối với phương pháp CF. Có phương pháp được áp dụng đến giải quyết vấn đề này là lấy trung bình có tỉ lệ của phương pháp content-based và CF. Tỉ lệ của content-based và CF sẽ được xác định tùy theo mỗi người dùng, cho phép hệ thống tối ưu hóa việc kết hợp content-based và CF cho mỗi người dùng, giúp giải quyết vấn đề gray sheep.

Shilling Attacks: Có nhiều trường hợp đánh giá sản phẩm, người dùng đưa ra đánh giá rất tốt về sản phẩm của họ và đánh giá xấu về những sản phẩm của đối thủ. Để tối ưu được hệ thống thì cần tránh được vấn đề này. Phương pháp item-based CF cho thấy bị ít ảnh hưởng bởi shilling attack hơn là user-based CF. Ngoài ra phương pháp lai cũng là một giải pháp cho shilling attack vì ít bị phụ thuộc vào đánh giá của người dùng khác hơn so với phương pháp CF đơn thuần.

1.4.3 Neighborhood-based Collaborative Filtering

Một phương pháp CF được áp dụng khá phổ biến là Neighborhood-based collaborative filtering (NBCF). Ý tưởng của phương pháp này là dự đoán mực độ quan tâm của người dùng đối với sản phẩm dựa trên những người dùng khác tương đồng với người dùng ban đầu mà đã có quan tâm tới sản phẩm này.

Tính toán độ tương đồng

Việc quan trọng nhất phải làm trong CF là xác định được sự giống nhau (similarity) giữa hai người dùng hay hai sản phẩm. Ví dụ trong item-based CF, việc tính toán độ giống nhau giữa sản phẩm i và sản phẩm j là cần tìm ra những người dùng đã đánh giá cả hai sản phẩm và áp dụng hàm tính toán độ giống nhau. Còn đối với user-based CF, tính toán độ giống nhau giữa hai người dùng u và người dùng v mà đã cùng đánh giá nhiều sản phẩm. Những hàm tính toán độ giống nhau bao gồm cosine similarity, euclidean distance, pearson correlation.
Cosine similarity là hàm được sử dụng nhiều nhất. Công thức của cosine similarity là dưới đây:

$$similarity = \cos u_1, u_2 = \frac{u_1^T u_2}{\|u_1^T\|.\|u_2\|} \tag{1.4}$$

Trong đó $u_{1,2}$ là vector tương ứng với người dùng 1, 2 đã được chuẩn hóa.
Độ similarity (giống nhau) của hai vector là một số trong đoạn $[-1, 1]$. Giá trị bằng 1 thể hiện hai vector hoàn toàn tương đồng với nhau. Hàm số cos của một góc bằng 1 nghĩa là góc giữa hai vector bằng 0, tức một vector bằng tích của một số dương với vector còn lại. Giá trị bằng -1 thể hiện hai vector này hoàn toàn trái ngược nhau. Điều này cũng hợp lý , tức khi hành vi của hai người dùng là hoàn toàn ngược nhau thì độ tương đồng giữa hai vector đó là thấp nhất.

Chuẩn hóa dữ liệu

Khi thực hiện tính similarity, cần phải gán một giá trị tạm thời nào đó cho những ô đánh giá trống trong Utility matrix. Nếu gán 0 thì không được vì giá trị quá thấp, sẽ giống như người dùng ghét sản phẩm đó, nếu gán trị trung bình ví dụ như 3 (đối với đánh giá 1 đến 5) thì vẫn có thể gặp vấn đề như sau. Nếu với những người dùng đánh giá khắt khe, thường chỉ đánh giá điểm 1, 2, 3 thì giá trị 3 ở đây có thể là quá cao, còn với người dùng dễ tính, thường đánh giá 3, 4, 5 thì giá trị này là có thể là quá thấp. Do vậy, cần phải chuẩn hóa dữ liệu của Utility matrix.
Bước đầu tiên của chuẩn hóa là tính trung bình các đánh giá của người dùng. Giá trị này cao sẽ tương ứng với người dùng dễ tính và ngược lại. Tiếp tục trừ từ mỗi đánh

giá đi giá trị này và thay giá trị chưa biết bằng không thì ta được Utility matrix đã chuẩn hóa. Bước chuẩn hóa này quan trọng cũng vì lý do sau đây. Thứ nhất, việc trừ đi trung bình cộng của mỗi cột khiến trong trong mỗi cột có những giá trị dương và âm. Những giá trị dương tương ứng với việc người dùng thích sản phẩm, những giá trị âm tương ứng với việc người dùng không thích sản phẩm. Những giá trị bằng 0 tương ứng với việc chưa xác định được liệu người dùng có thích sản phẩm hay không. Lý do thứ hai là vì số chiều của Utility matrix thường rất lớn, có thể lên tới hàng chục triệu người dùng và hàng triệu sản phẩm, việc lưu toàn bộ Utility matrix sẽ tốn rất nhiều bộ nhớ. Thấy rằng số lượng đánh giá biết trước thường rất nhỏ so với kích thước của utility matrix nên cách tốt hơn là lưu dưới dạng sparse matrix, tức chỉ lưu giá trị khác không và vị trí của chúng. Tức là những đánh giá chưa biết nên được thay bằng 0. Việc này giúp tối ưu bộ nhớ rất nhiều.

Điền các giá trị khuyết trong utility matrix

Việc dự đoán mức độ quan tâm của người dùng với một sản phẩm dựa trên các người dùng giống với người dùng này nhất rất giống với phương pháp k-nearest neighbors (KNN) với hàm khoảng cách là cosine similarity. Tương tự như KNN, NBCF cũng dùng thông tin của k người dùng gần nhất để dự đoán. Để đánh giá độ quan tâm của người dùng đối với một sản phẩm, chỉ cần quan tâm tới các người dùng lân cận đã từng đánh giá sản phẩm đó. Tuy nhiên, trong KNN thì giá đo khoảng cách là các giá trị không âm còn similarity trong NBCF thì có thể âm. Công thức dự đoán đánh giá của người dùng u cho sản phẩm i là:

$$\widehat{y_{i,u}} = \frac{\sum_{u_j \in N(u,i)} \overline{y_{i,u}} sim(u,u_j)}{\sum_{u_j \in N(u,i)} |sim(u,u_j)|} \tag{1.5}$$

Trong $N(u,i)$ là tập hợp k người dùng có similarity cao nhất với người dùng u mà đã đánh giá sản phẩm i. Để quy đổi giá trị đánh giá dự đoán về thang 5, cộng giá trị dự đoán với giá trị đánh giá trung bình của mỗi người dùng.
Việc thực hiện gợi ý có thể thực hiện theo nhiều cách. Có thể sắp xếp những sản phẩm chưa được đánh giá theo giá trị giảm dần của đánh giá dự đoán để gợi ý cho người dùng, hoặc gợi ý tất cả sản phẩm có đánh giá dự đoán đã chuẩn hóa lớn 0.

Item-based collaborative filtering

Phương pháp user-based vẫn còn một số hạn chế như sau. Đầu tiên là, số lượng người dùng thường lớn hơn số lượng sản phẩm rất nhiều. Do đó kích thước ma trận similarity là rất lớn. Việc lưu trữ ma trận nhiều khi không khả thi. Thứ hai là, ma trận utility thường rất thưa và số lượng người dùng lại rất lớn so với số lượng sản

phẩm dẫn đến nhiều cột của utility matrix có rất ít hoặc không có giá trị biết trước. Do đó, khi người dùng thay đổi đánh giá trước đó hay đánh giá thêm sản phẩm, trung bình cộng đánh giá hay vector chuẩn hóa thay đổi rất nhiều. Dẫn đến việc tính toán ma trận similarity, vốn đã tốn nhiều thời gian và bộ nhớ, cần phải thực hiện lại.

Cách tiếp cận khác là thay vì tìm sự giống nhau giữa các người dùng, chuyển sang tìm sự giống nhau giữa các sản phẩm. Nếu người dùng thích một sản phẩm thì hệ thống nên gợi ý những sản phẩm tương tự sản phẩm đó cho người dùng đó. Phương pháp này có những ưu điểm sau. Vì số sản phẩm nhỏ hơn nhiều số sản phẩm nên ma trận similarity nhỏ hơn rất nhiều giúp tính toán hiệu quả hơn. Thứ hai là vì số lượng sản phẩm ít hơn người dùng mà tổng đánh giá không đổi, dẫn đến việc số đánh giá sản phẩm có được sẽ nhiều hơn số đánh giá mà người dùng đánh giá. Từ đó dẫn đến việc tính độ tương đồng đáng tin cậy hơn, giá trị trung bình mỗi hàng thay đổi ít hơn khi có đánh giá bị thay đổi dẫn tới việc cập nhật ma trận similarity được thực hiện ít hơn.

Top-N recommendation

Top-N recommendation là việc gợi ý một tập hợp gồm N sản phẩm được xếp hạng mà người dùng có thể thích. Ví dụ như khi sử dụng youtube, nếu người dùng truy cập youtube với tài khoản youtube của mình, ở trang chủ, người dùng sẽ được gợi ý một số video phù hợp với mỗi cá nhân. Phương pháp Top-N recommendation phân tích utility matrix để tìm ra sự liên hệ giữa các người dùng hay sản phẩm khác nhau và dựa trên đó tìm những gợi ý hợp lý. Ta có thể chia Top-N recommendation thành 2 phương pháp là user-based top-N recommendation và item-based top-N recommendation.

Với phương pháp user-based top-N recommendation, bước đầu tiên là tìm ra k người dùng giống nhất với người dùng đang muốn được recommendation nhất. Sau khi đã tìm ra k người dùng đó, những sản phẩm đã được k người dùng này đánh giá (mua, xem) được tập hợp thành một tập C. Trong tập C, những sản phẩm được đánh giá cao hay xuất hiện nhiều mà người dùng chưa mua sẽ được ưu tiên gợi ý.
Còn với phương pháp item-based top-N recommendation, đầu tiên tìm tập k sản phẩm giống nhất với mỗi sản phẩm mà người dùng đã mua. Sau đó tìm ra tập C bằng cách loại bỏ những sản phẩm mà người dùng đã mua ra khỏi tập k sản phẩm giống nhất với các sản phẩm đã mua. Những sản phẩm còn lại sẽ được xếp hạng dựa trên độ giống của sản phẩm với sản phẩm mà người dùng đã mua.

Lợi thế và khó khăn khi ứng dụng neighborhood-based collaborative filtering

Việc ứng dụng NBCF có những lợi thế khá rõ như dễ áp dụng, không phụ thuộc vào thông tin của sản phẩm, không phải thực hiện công đoạn huấn luyện, ngoài ra so với phương pháp khác như content-based thì có độ chính xác cao hơn. Tuy nhiên, phương pháp NBCF vì là phương pháp memory-based nên vẫn có vấn đề lớn về khả năng mở rộng. Khi mà số người dùng hay số sản phẩm trở nên quá lớn, ma trận similarity sẽ tốn rất nhiều bộ nhớ. Hơn nữa, với số người dùng hay sản phẩm càng lớn thì việc tìm ra K hàng xóm gần nhất sẽ tốn kếm rất nhiều tài nguyên tính toán.

1.4.4 Clustering Collaborative Filtering

Model-based Collaborative Filtering

Phương pháp NBCF ở trên là phương pháp memory-based CF. Phương pháp memory-based CF sử dụng toàn bộ hay một phần của bộ dữ liệu để thực hiện việc dự đoán. Phương pháp này có một số ưu điểm như là dễ áp dụng, dữ liệu mới có thể dễ dàng được thêm vào và sử dụng luôn để đánh giá. Tuy vậy memory-based CF vẫn còn một số nhược điểm như hiệu suất thấp khi bộ dữ liệu thưa và đặc biệt là ít có khả năng mở rộng quy mô khi gặp bộ dữ liệu quá lớn do tốn rất nhiều bộ nhớ. Những vấn đề này có thể được giải quyết với Model-based CF. Clustering CF là một trong những phương pháp tiếp cận của Model-based CF.

Giới thiệu Clustering Collaborative Filtering

Nội dung dưới đây được đươc viết dựa theo tài liệu [3].

Clustering CF dựa trên giả định rằng người dùng trong cùng một nhóm có cùng sở thích, vì vậy những người dùng này đánh giá các mặt hàng tương tự nhau. Do đó, người dùng được phân chia thành các nhóm được gọi là các cluster được định nghĩa là một nhóm người dùng tương tự nhau.

Giả sử mỗi người dùng được biểu diễn dưới dạng vector xếp hạng ký hiệu $u_i = (r_{i1}, r_{i2},..., r_{in})$. Thước đo khác nhau giữa hai người dùng là khoảng cách giữa họ. Chúng ta có thể sử dụng khoảng cách Minkowski, khoảng cách Euclidian khoảng cách Manhattan hay độ tương đồng cosine.

$$distance_{Minkowski}(u_1, u_2) = \sqrt[q]{\sum_j (r_{1j} - r_{2j})^q}$$

$$distance_{Euclidian}(u_1,u_2) = \sqrt{\sum_j (r_{1j}-r_{2j})^2}$$

$$distance_{Manhattan}(u_1,u_2) = \sum_j |r_{1j}-r_{2j}|$$

$$similarity_{cosine}(u_1,u_2) = \cos u_1,u_2 = \frac{u_1^T u_2}{\|u_1^T\|.\|u_2\|} = \frac{\sum_j r_{1j}r_{2j}}{\sqrt{\sum_j r_{1j}^2}\sqrt{\sum_j r_{2j}^2}}$$

Khoảng cách giữa u_1 và u_2 càng ngắn hay độ tương đồng càng cao thì u_1 và u_2 càng giống nhau. Để thực hiện Clustering CF cần thực hiện hai bước sau:

- Phân người dùng thành các cụm và mỗi cụm luôn chứa các giá trị đánh giá. Ví dụ, Mỗi cụm đều là kết quả từ thuật toán k-mean và một giá trị trung bình là vector đánh giá giống như vector người dùng.
- Người dùng cần được gợi ý được đưa vào một trong các cụm và đánh giá của người đó sẽ giống như đánh giá của cụm đó. Việc đưa người dùng vào cụm nào sẽ dựa trên khoảng cách của người dùng đới với cụm.

Vậy nên bước quan trọng nhất là làm sao để phân người dùng vào các cụm. Có nhiều phương pháp clustering như k-mean và k-centroid. Phương pháp phổ biến nhất là k-mean, bao gồm ba bước sau đây:

- Ta chọn ngẫu nhiên k người dùng, mỗi người ban đầu đại diện cho tâm của một cụm. Từ đó, chúng ta có k tâm cụm. Mỗi tâm cụm này được coi là đại diện của người dùng của một cụm. Có tổng cộng k cụm.
- Đối với mỗi người dùng, khoảng cách giữa mỗi người dùng với k tâm cụm được tính toán. Người dùng đó sẽ thuộc về cụm gần nhất. Nói cách khác, nếu người dùng u_i thuộc về cụm c_v, khoảng cách giữa u_i và tâm m_v của cụm c_v, được ký hiệu là $distance(u_i,m_v)$ là nhỏ nhất trong tất cả các cụm.
- Sau đó, tâm của tất cả các cụm được tính toán lại. Nếu điều kiện dừng được thỏa mãn thì dừng lại, nếu không thì lặp lại bước thứ hai.

Quá trình này được lặp lại cho đến khi điều kiện dừng được thỏa mãn. Có hai điều kiện dừng điển hình cho thuật toán k-mean:

- k tâm của các cụm không thay đổi. Nói cách khác, k cụm không thay đổi. Điều kiện này thể hiện đã hoàn thành phân cụm.
- Cách khác là tiêu chí lỗi nhỏ hơn một ngưỡng được xác định trước.

Nếu như điều kiện dừng là tiêu chí lỗi nhỏ hơn một ngưỡng được xác định trước, tiêu chí lỗi sẽ được tính nhau sau:

$$error = \sum_{v=1}^{k} \sum_{u_i \in c_v} distance(u_i, m_v)$$

Trong đó c_v và m_v lần lượt là cụm v và tâm của nó.

Lợi thế và vấn đề gặp phải khi áp dụng Clustering Cf

Phương pháp clustering CF có lợi thế như có thể dễ dàng áp dụng để chia bộ dữ liệu phức tạp thành nhiều cụm khác nhau;, phù hợp cho bộ dữ liệu lớn; có tính linh hoạt cao, điều chỉnh tâm các cụm có thể thay đổi kết quả đầu ra; có thời gian chạy tuyến tính với số đối tượng. Tuy nhiên clutering CF còn có một số vấn đề như không thể biết được số lượng cụm tối ưu do đó số cụm phải được chọn từ trước; nếu các cụm ban đầu được chọn ngẫu nhiên thì kết quả có thể thiếu tính nhất quán.

1.4.5 Matrix factorization collaborative filtering

Nội dung dưới đây được đươc viết dựa theo tài liệu [7].

Giới thiệu

MFCF là một hướng tiếp cận khác cho collaborative filtering dựa trên bài toán phân tích ma trận thành nhân tử (matrix factorization hoặc matrix decomposition).

Trong hệ thống gợi ý content-based, mỗi sản phẩm đều có sản phẩm profile, được mô tả bằng vector x. Ta cần tìm vector hệ số w tương ứng với mỗi người dùng. Từ đó đánh giá của người dùng đó cho sản phẩm xấp xỉ với:

$$y \approx w^T x = x^T w$$

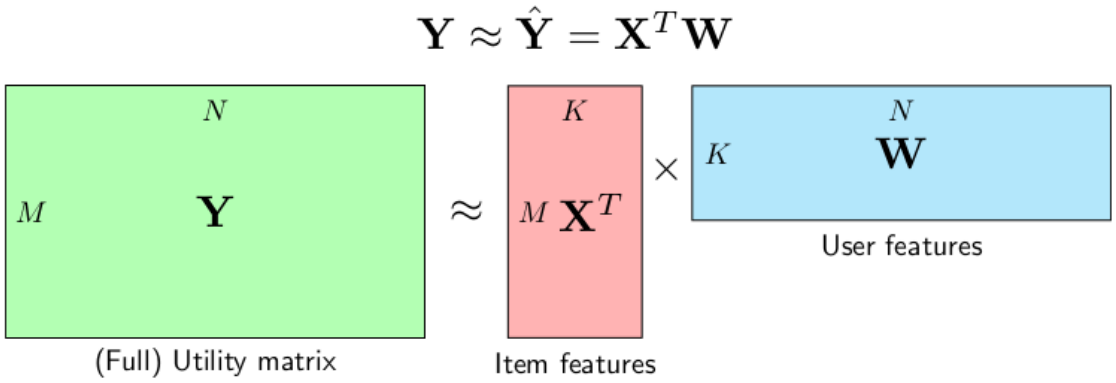
Khi đó ma trận utility sẽ xấp xỉ với:

$$Y \approx X^T W$$

Vector x được xây dựng dựa trên thông tin đặc điểm của sản phẩm. Như vậy việc xây dựng sản phẩm profile đóng vai trò quan trọng, ảnh hưởng trực tiếp đến hiệu năng của mô hình.

Giả sử, không cần phải xây dựng trước các vector x dựa trên thông tin sản phẩm và vector này có thể được huấn luyện đồng thời với vector hệ số của mỗi người dùng.

Có nghĩa là biến số trong bài toán tối ưu là cả X và W. X là ma trận của toàn bộ sản phẩm profile, mỗi cột tương ứng với một sản phẩm, W là ma trận của toàn bộ vector hệ số của người dùng, mỗi cột tương ứng với một người dùng.



Hình 1.2: Mô hình tích 2 ma trận [7]

Với cách làm này, ta cố gắng xấp xỉ ma trận utility $Y \in R^{M \times N}$ bằng tích của hai ma trận $X \in R^{M \times K}$ và $W \in R^{K \times N}$ tương tự như hình 1.2. Thông thường, K được chọn là số nhỏ hơn rất nhiều so với M, N. Khi đó, cả hai ma trận X và W đều có rank không vượt quá K. Do vậy, phương pháp này còn được gọi là low-rank matrix factorization.

Ý tưởng đằng sau MFCF chính là sự tồn tại của các tính chất ẩn mô tả sự liên quan giữa các sản phẩm và người dùng. Ví dụ trong hệ thống gợi ý sách, những tính chất ẩn ở đây có thể là các thể loại sách trinh thám, lịch sử, giáo dục,... hoặc cũng có thể là kết hợp nào đó của các thể loại này. Mỗi sản phẩm sẽ mang tính chất ẩn ở một mức độ nhất định nào đó, điều này sẽ được thể hiện ở vector x của sản phẩm đó, hệ số này càng cao thể hiện sản phẩm mang cần nhiều tính chất đó. Mỗi người dùng cũng tương tự như vậy. Mỗi người dùng có thể có sở thích đọc sách trinh thám, lịch sử, giáo dục,... hoặc thích thể loại được kết hợp bởi những thể loại này, điều này sẽ được thể hiện ở vector w của người dùng đó. Hệ số của 1 tính chất ẩn nào đó càng cao càng chứng tỏ người dùng càng thích sản phẩm chứa nhiều tính chất ẩn đó. Từ đó giá trị $x^T w$ sẽ thể hiện được hứng thú của người dùng với sản phẩm nhất định. Hệ số tính chất ẩn của sản phẩm càng lớn và hệ số thể hiện độ thích của người dùng với tính chất đó càng lớn, giá trị trên sẽ càng lớn.

Xây dựng và tối ưu hàm mất mát

Ta có giá trị đánh giá của người dùng n cho sản phẩm m được tính theo công thức $y_{mn} = x_m \times y_n$. Xây dựng hàm mất mát của MFCF cũng tương tự như việc xây dựng hàm mất mất của content-based. Điểm khác biết là biến tối ưu là cả X và W.

Hàm mất mát của MFCF có thể được viết như sau:

$$L(X,W) = \frac{1}{2s}\sum_{n=1}^{N}\sum_{m:r_{mn}=1}(y_{mn} - x_m w_n)^2 + \frac{\lambda}{2}(\|X\|_F^2 + \|W\|_F^2) \tag{1.6}$$

r_{mn} sẽ bằng 1 khi người dùng n đã đánh giá cho sản phẩm m, s là toàn bố số đánh giá có trong bộ huấn luyện.

Để tối ưu hàm mất mát ở trên, ta thực hiện các bước như sau. Đầu tiên cố định X, thực hiện tối ưu W. Sau đó cố định W, thực hiện tối ưu X. Lặp lại 2 bước trên cho tới khi điều kiện dừng được thỏa mãn.
Khi cố định X, việc tối ưu W tương đướng với tối ưu hàm sau:

$$L(W) = \frac{1}{2s}\sum_{n=1}^{N}\sum_{m:r_{mn}=1}(y_{mn} - x_m w_n)^2 + \frac{\lambda}{2}\|W\|_F^2 \tag{1.7}$$

Khi cố định W, việc tối ưu X tương đướng với tối ưu hàm sau:

$$L(X) = \frac{1}{2s}\sum_{n=1}^{N}\sum_{m:r_{mn}=1}(y_{mn} - x_m w_n)^2 + \frac{\lambda}{2}\|X\|_F^2 \tag{1.8}$$

Mỗi hàm ở trên đều có thế được tối ưu bằng gradient descent. Áp dụng bài toán gradient descent như ở phần content-based, ta có công thức cập nhật mỗi cột của W là:

$$w_n = w_n - \eta(-\frac{1}{s}\hat{X}_n^T(\hat{y}_n - \hat{X}_n w_n) + \lambda w_n)$$

trong đó $\hat{X}_n$ là ma trận được tạo bởi các hàng tương ứng với các sản phẩm đã được đánh giá bởi bởi người dùng n, $\hat{y}_n$ là những giá trị đánh giá tương ứng.
Công thức cập nhật mỗi hàng của X là:

$$x_m = x_m - \eta(-\frac{1}{s}\hat{W}_m^T(\hat{y}_m - x_m\hat{W}_m) + \lambda x_m)$$

trong đó $\hat{W}_m$ là ma trận được tạo bởi các cột tương ứng với người dùng đã đánh giá sản phẩm m, $\hat{y}_m$ là những giá trị đánh giá tương ứng.

Lợi thế và vấn đề của phương pháp Matrix factorization

Trong thực tế, số lượng người dùng và số lượng sản phẩm thường rất lớn. Phương pháp như content-based rất dễ để áp dụng và không cần phải thực hiện huấn luyện nhưng khi dự đoán đánh giá lại phải thực hiện việc tìm ra ma trận similarity và việc tìm ra K đối tượng giống nhất, việc này tốn rất nhiều bộ nhớ và khối lượng

tính toán khi số lượng người dùng và sản phẩm lớn. Phương pháp MFCF tuy phức tạp khi thực hiện huấn luyện vì phải lặp đi lặp lại việc tối ưu một ma trận trong khi cố định ma trận còn lại, tuy vậy khi dự đoán đánh giá chỉ cần phải lấy tích vô hướng của hai vector $x^T w$, mỗi vector đều có độ dài là K, nhỏ hơn rất nhiều so với số người dùng và sản phẩm. Vậy nên quá trình dự đoán đánh giá không đòi hỏi khả năng tính toán quá cao. Hơn nữa, việc lưu lại các vector x và w trong bộ nhớ chỉ bao gồm $K(M+N)$ phần tử còn với phương pháp content-based thì phải lưu M^2 hay N^2 phần tử với M, N là số lượng người dùng và sản phẩm.

Vấn đề gặp phải của MFCF là tuy thời gian dự đoán đánh giá rất nhanh nhưng thời gian huấn luyện lại rất lâu đối với tập dữ liệu lớn. Trong thực tế ma trận utility liên tục thay đổi do có thêm người dùng mới, sản phẩm mới, đánh giá mới của người dùng hay người dùng thay đổi đánh giá của họ. Điều này dẫn tới việc phải liên tục huấn luyện dẫn đến tốn thời gian.

1.5 Hệ thống gợi ý Hybrid

Nội dung dưới đây được đươc viết dựa theo tài liệu [2].

Phương pháp gợi ý Hybrid, hay phương pháp gợi ý lai thường là kết hợp của hai phương pháp hoặc nhiều hơn các phương pháp gợi ý nhằm có được độ chính xác cao hơn và để giải quyết một số vấn đề xảy ra khi chỉ dùng một phương pháp nhất định. Thông thường, phương pháp collaborative filtering được kết hợp với phương pháp khác để giải quyết một số vấn đề.

1.5.1 Phương pháp hybrid sử dụng trọng số

Phương pháp hybrid sử dụng trọng số là giá trị đánh giá cho sản phẩm được tính toán từ kết quả của các phương pháp gợi ý có trong hệ thống. Ví dụ như một phương pháp hybrid đơn giản là kết hợp tuyến tính của của các giá trị đánh giá. Hệ thống P-Tango sử dụng phương pháp này. Hệ thống này ban đầu cho trọng số của collaborative filtering và content-based bằng nhau, sau đó điều chỉnh trọng số này để tăng độ chính xác.

Điểm mạnh của phương pháp Hybrid có trọng số là có thể sử dụng toàn bộ khả năng của hệ thống một cách đơn giản và nhanh chóng, có thể dễ dàng điều chỉnh các trọng số. Tuy nhiên phương pháp này đồng nghĩa về việc giá trị đánh giá của các phương pháp khác nhau phải đồng nhất trong không gian của item. Tuy nhiên trong thực tế thì không phải như thế, chẳng hạn như phương pháp collaborative

filtering sẽ khó đánh giá chính xác khi không có nhiều đánh giá có sẵn.

1.5.2 Phương pháp hybrid chuyển đổi giữa các phương pháp (switching hybrid)

Phương pháp switching hybrid sử dụng một số tiêu chí để chuyển đổi giữa các phương pháp gợi ý khác nhau. Hệ thống DailyLearner [1] sử dụng phương pháp hybrid bao gồm content-based và collaborative filtering. Trong hệ thống này phương pháp content-based được áp dụng trước. Nếu không có đủ thông tin để có thể đưa ra gợi ý có tính tin cậy cao từ phương pháp content-based thì sẽ chuyển sang sử dụng phương pháp collaborative filtering. Phương pháp switching hybrid này không thể hoàn toàn xử lý được vấn đề vì cả content-based và collaborative filtering đều gặp vấn đề với cold start với người dùng mới. Tuy vậy, phương pháp content-based của DailyLearner là nearest-neighbor, không đòi hỏi dữ liệu lớn để có thể phân loại chính xác.

Trong switching hybrid, phương pháp collaborative filtering cung cấp khả năng thực hiện gợi ý những sản phẩm không cùng thể loại, có ngữ nghĩa không giống với với sản phẩm được đánh giá cao trước đó, nhưng vẫn có liên quan. Phương pháp switching hybrid làm tăng thêm độ phức tạp của hệ thống gợi ý vì cần phải lựa chọn tiêu chí chuyển đổi giữa các phương pháp, đòi hỏi có thêm biến số. Tuy nhiên, lợi ích của hệ thống này là có thể tận dụng tối ưu ưu điểm của các phương pháp gợi ý thành phần của nó.

1.5.3 Phương pháp hybrid hỗn hợp (mixed hybrid)

Khi cần thực hiện một lượng lớn gợi ý, có thể áp dụng phương pháp sử dụng cùng lúc các kết quả gợi ý từ nhiều phương pháp khác nhau. Ví dụ như một hệ thống gợi ý chương trình TV có thể sử dụng phương pháp này. Đưa ra gợi ý bằng content-based dựa trên thông tin về nội dung của các chương trình và sử dụng collaborative filtering dựa trên lịch sử xem của người dùng khác. Kết hợp của cả hai kết quả này có thể dùng để đưa ra kết quả gợi ý cuối cùng.

Phương pháp mixed hybrid có thể giúp tránh được vấn đề cold start của item. Kết quả từ phương pháp như content-based có thể đưa ra được gợi ý mặc dù item đó chưa được người dùng đánh giá. Tuy vậy, phương pháp này không giúp giải quyết vấn đề cold start của người dùng vì cả phương content-based và collaborative filtering đều cần dữ liệu liên qua tới người dùng để thực hiện gợi ý.

1.5.4 Hybrid Recommender kết hợp tính chất (feature combination)

Phương pháp này là phương pháp kết hợp giữa content-based và collaborative filtering dựa trên việc coi thông tin đánh giá của người dùng cho item như là thông tin tính chất của item, sau đó thực hiện phương pháp content-based cho toàn bộ dữ liệu này. Phương pháp feature combination hybrid này cho phép hệ thống sử dụng thông tin của cả content-based và collaborative filtering chứ không chỉ phụ thuộc vào một phương pháp nào. Do đó hệ thống sẽ không chỉ phụ thuộc vào việc người dùng có đánh giá item hay không.

1.6 Đánh giá hệ thống gợi ý

1.6.1 Đánh giá mức độ chính xác của giá trị đánh giá dự đoán

Độ chính xác của hệ thống gợi ý có thể được kết luận dựa trên kết quả đánh giá. Việc đánh giá hệ thống gợi ý tùy theo mỗi loại hệ thống. Có một số phương pháp đánh giá như Mean Absolute Error (MAE), Normalized Mean Absolute Error (NMAE), Root Mean Squared Error (RMSE).

Mean Absolute Error (MAE) và Normalized Mean Absolute Error (NMAE)

Phương pháp được sử dụng tương đối phổ biến trong việc đánh hệ thống gợi ý là MAE, tính toán trung bình của giá trị đối của hiệu giữa đánh giá dự đoán và đánh giá thực. MAE càng nhỏ cho thấy kết quả đánh giá dự đoán càng chính xác.

$$MAE = \frac{\sum_{\{i,j\}} |p_{i,j} - r_{i,j}|}{n} \tag{1.9}$$

Trong đó n là tổng số đánh giá của tất cả người dùng, $p_{i,j}$ là giá trị đánh giá dự đoán của người dùng i đối với sản phẩm j còn $r_{i,j}$ là giá trị đánh giá thực của người dùng i đối với sản phẩm j. Giá trị MAE càng thấp chứng tỏ độ chính xác của dự đoán càng cao.
Hệ thống gợi ý khác có thể dùng một thang đo khác để đánh giá. Normalized Mean Absolute Error (NMAE) giúp tính ra xem độ sai lệch (error) chiếm bao nhiêu phần trăm của toàn bộ thang đánh giá. NMAE càng nhỏ cho thấy mức độ sai lệch trong dự đoán so với toàn bộ trường giá trị dự đoán càng thấp.

$$NMAE = \frac{MAE}{r_{\max} - r_{\min}} \tag{1.10}$$

$r_{\max}$ là giá trị lớn nhất và $r_{\min}$ là giá trị nhỏ nhất có thể đánh giá.

Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) trở nên nổi tiếng vì nó là phương pháp đánh giá cho gợi ý phim của Netflix prize. RMSE dùng để đánh giá mức độ sai lệch của giá trị đánh giá được dự đoán so với giá trị đánh giá thực, RMSE càng thấp tương đương với độ chính xác càng cao.

$$RMSE = \sqrt{\frac{1}{n}\sum_{\{i,j\}}(p_{i,j} - r_{i,j})^2} \tag{1.11}$$

Trong đó n là tổng số lượng đánh giá của tất cả người dùng, $p_{i,j}$ của người dùng i đối với sản phẩm j còn $r_{i,j}$ là giá trị đánh giá thực của người dùng i đối với sản phẩm j.

1.6.2 Đánh giá Top-N recommendation

Hit Rate

Để đánh giá top N sản phẩm gợi ý cho người dùng có thể sử dụng hit rate. Mỗi khi người dùng đánh giá 1 sản phẩm trong top N sản phẩm gợi ý, coi đó là 1 hit. Các bước tính ra hit rate của 1 người dùng như sau. Đầu tiên, tìm tất cả những sản phẩm mà người dùng đã đánh giá. Sau đó loại bỏ đi một trong những sản phẩm này (đây còn gọi là leave-one-out cross validation). Sử dụng tất cả những sản phẩm còn lại để tìm ra top N recommendation. Nếu như sản phẩm đã bị loại bỏ xuất hiện trong top N recommendation thì đó là 1 hit. Hit rate sẽ được tính bằng công thức sau.

$$HR = \frac{total\ hit}{total\ user} \tag{1.12}$$

Giá trị này cho thấy được tỉ lệ sản phẩm đã bị loại bỏ xuất hiện trong top N gợi ý là bao nhiêu. Giá trị này càng cao chứng tỏ hệ thống Top-N recommendation càng tốt. Vấn đề của phương pháp này là khó để có thể gợi ý đúng 1 sản phẩm duy nhất trong top N gợi ý. Do đó giá trị của hit rate với phương pháp leave-one-out cross validation thường khá nhỏ trừ phi có tập dữ liệu lớn.

Rating Hit Rate (rHR)

Một cách khác để tính hit rate là chia ra thành các hit rate của mỗi giá trị đánh giá. Những giá trị này cho thấy được những sản phẩm được gợi ý được người dùng thích đến mức độ nào vì mục đích của Top-N recommendation là gợi ý những sản phẩm mà người dùng thực sự thích. Vậy nên ở đây ta quan tâm đến những rating có giá trị cao hơn là rating có giá trị thấp.

Cumulative Hit Rate (cHR)

Vì ta chỉ quan tâm tới những rating cao, ta có thể bỏ qua rating thấp và chỉ tính toán hit rate của giá trị cao. Ví dụ như chỉ tính toán hit rate của của rating >= 4. Nếu gặp hit nhưng rating lại < 4 thì vẫn không được tính là hit.

Average Reciprocal Hit Ranking (ARHR)

Phương pháp này không chỉ tính hit rate mà còn đánh giá cả việc hit đó xuất hiện ở vị trí nào trong xếp hạng top N sản phẩm được gợi ý. Hit ở vị trí càng cao trong bảng xếp hạng này thì có giá trị càng lớn. ARHR có thể được tính theo công thức sau đây.

$$ARHR = \frac{\sum_{i=1}^{n} \frac{1}{rank_i}}{total\,user} \tag{1.13}$$

Với $rank_i$ là vị trí của hit i trong top N recommendation. Ví dụ như hit ở vị trí 1 sẽ có giá trị là 1 nhưng hit ở vị trí 3 thì sẽ chỉ có giá trị là $\frac{1}{3}$.

Mức độ đa dạng

Ngoài việc chỉ đo lường mức độ chính xác của danh sách gợi ý, còn có tính chất khác của danh sách gợi ý cần được đo lường. Một tính chất điển hình đó là mức độ đa dạng của danh sách gợi ý. Trong một hệ thống gợi ý, việc quan trọng không chỉ là gợi ý cho người dùng sản phẩm mà người dùng sẽ thích mà việc có thể gợi ý được nhiều loại sản phẩm có đặc điểm, thể loại khác nhau cũng rất quan trọng. Mức độ đa dạng thể hiện việc những sản phẩm trong danh sách được dùng để gợi ý cho người dùng khác nhau đến mức độ nào. Sự tương đồng giữa các sản phẩm có thể được quyết định dựa trên các đặc điểm của sản phẩm hoặc dựa trên việc các sản phẩm có được đánh giá giống nhau hay không. Một phương pháp để đo độ đa dạng là Intra-List Similarity (ILS) [8]. Công thức ILS tính độ tương đồng giữa các sản phẩm trong danh sách gợi ý cho người dùng sử dụng công thức tính độ tương đồng như cosine similarity. Công thức ILS cho một người dùng u:

$$ILS_u = \frac{1}{2} \sum_{i_j \in L} \sum_{i_k \in L; i_j \neq i_k} sim(i_k, i_j) \tag{1.14}$$

L là danh sách những sản phẩm được gợi ý cho người dùng u. $sim(i_k, i_j)$ là phép tính độ tương đồng giữa hai sản phẩm i_k và i_j sử dụng cosine similarity.

Kết quả ILS càng lớn chứng tỏ mức độ đa dạng càng thấp. Từ kết quả ILS cho mỗi người dùng có thể tính được trung bình ILS của toàn bộ người dùng trong hệ thống gợi ý.

Chương 2

Bài toán và các mô hình thực nghiệm

2.1 Bài toán

2.1.1 Đặt vấn đề

Hiện nay, ngành công nghệ thông tin đang ngày càng phát triển, kèm theo đó là sự phát triển của các dịch vụ online như youtube, netflix, amazon,... Và đối với những dịch vụ này, hệ gợi ý đóng vai trò vô cùng quan trọng trong các dịch vụ này. Chẳng hạn như Amazon khẳng định 35% doanh thu của họ đến từ hệ thống gợi ý của dịch vụ này [6]. Trong hệ gợi ý, một bài toán được sử dụng rất nhiều đó chính là Top-N recommendation.

Bài toán này nhằm tìm ra N sản phẩm mà mỗi người dùng sẽ có xu hướng hứng thú nhất dựa trên những sản phẩm mà người dùng đã tương tác. Những ví dụ tiêu biểu của bài toán này là những video gợi ý cho người xem ở trang chủ youtube sau khi người dùng đăng nhập, những sản phẩm được gợi ý khi mua hàng trên amazon, những phim được gợi ý khi đăng nhập vào netflix,... Trong phần ứng dụng sẽ triển khai hệ thống gợi ý áp dụng các phương pháp khác nhau để đưa ra gợi ý cho người dùng sau đó so sánh kết quả.

2.1.2 Bài toán đặt ra

Đồ án này tập trung vào việc giải quyết bài toán xây dựng hệ thống gợi ý sách cho người dùng. Dựa trên những dữ liệu đã có sẵn, áp dụng một số phương pháp gợi ý khác nhau để gợi ý cho người dùng những quyển sách phù hợp với sở thích. Sau đó đánh giá và so sánh kết quả giữa các phương pháp.

2.2 Công cụ sử dụng

2.2.1 Môi trường phát triển

Google Colab

Google colab là môi trường Jupyter Notebook miễn phí, không yêu cầu cài đặt và chạy hoàn toàn ở trên cloud. Sử dụng Google Colab, ta có thể thực thi viết và thực thi code, lưu và chia sẻ nghiên cứu, và có thể sử dụng nguồn tài nguyên tính toán mạnh.

Điểm mạnh của Google colab là đã tích hợp sẵn rất nhiều thư viện Python. Đối với người bắt đầu học python thì rất nhanh và tiện lợi do không cần cài đặt gì. Google colab cho phép sử dụng GPU miễn phí. Người dùng có thể thực thi code không bị gián đoạn trong 12 giờ liên tục, được sử dụng RAM lên tới 25GB. Notebook luôn được lưu ở trên google drive nên có thể chỉnh sửa, chia sẻ và thực thi ở mọi nơi.

Tuy vậy, Google colab vẫn có 1 điểm bất lợi như sau. Người dùng vẫn cần phải cài đặt tất cả thư viện không có sẵn và phải thực hiện điều này trong tất cả các session làm việc. Google drive luôn là nơi lấy dữ liệu và lưu trữ dữ liệu nhưng đôi khi cũng có thể là trên local, việc này có thể dẫn đến việc tiêu tốn bandwidth rất nhiều khi bộ dữ liệu lớn. Điểm bất lợi lớn nhất của Google colab là vì sử dụng dữ liệu từ Google drive nên rất khó để sử dụng bộ dữ liệu quá lớn vì Google drive chỉ cung cấp 15 GB miễn phí.

2.2.2 Bộ cơ sở dữ liệu

Goodbooks 10k

Bộ dữ liệu này chứa đánh giá của người dùng cho mười nghìn cuốn sách phổ biến, với 6 triệu đánh giá. Nguồn của những đánh giá này được tìm thấy trên internet. Nói chung có khoảng 100 đánh giá cho mỗi cuốn sách, tuy nhiên một số có ít đánh giá hơn. Có thể đánh giá từ một đến năm.
Cả ID sách và ID người dùng đều liền kề nhau. Đối với sách, chúng là 1 đến 10000, đối với người dùng là 1 đến 53424. Tất cả người dùng đã thực hiện ít nhất hai xếp hạng. Số xếp hạng trung bình cho mỗi người dùng là 8.
Ngoài ra còn có dữ liệu những cuốn sách được đánh dấu để đọc bởi người dùng, thông tin cơ bản của sách như tác giả, năm,... và thể loại sách.

2.3 Tiếp cận sử dụng phương pháp Neighbor-based Collaborative Filtering

Vì phương pháp NBCF là phương pháp memory-based do đó không cần phải thực hiện bước huấn luyện mô hình. Trước khi thực hiện việc dự đoán đánh giá của người dùng cho sách hay xây dựng danh sách gợi ý N quyển sách cho người dùng, chỉ cần thực hiện 2 bước là chuẩn hóa dữ liệu và xây dựng ma trận similarity.

- Việc chuẩn hóa dữ liệu được thực hiện như sau. Đối với hướng tiếp cận user-based, tất cả đánh giá của một người dùng bị trừ đi giá trị trung bình cộng của các giá trị đánh giá đã biết của người dùng đó. Còn đối với hướng tiếp cận item-based, tất cả giá trị đánh giá của người dùng cho một item bị trừ đi giá trị bằng trung bình cộng những giá trị đó.
- Ma trận similarity lưu độ tượng đồng giữa những người dùng hay những quyển sách. Ma trận similarity được xây dựng dựa trên similarity.

$$similarity_{cosine}(u_1, u_2) = \cos u_1, u_2 = \frac{u_1^T u_2}{\|u_1^T\|.\|u_2\|}$$

2.3.1 Dự đoán đánh giá của người dùng đối với sách

Đối với hướng tiếp cận user-based, việc dự đoán đánh giá của người dùng u cho quyển sách i được thực hiện bằng cách tìm ra k người dùng giống với người dùng i nhất đã từng đánh giá quyển sách i dựa trên ma trận similarity. Khi tìm ra k người dùng này, để dự đoán đánh giá người dùng u cho quyển sách i, dùng công thức tính trung bình có trọng số như sau:

$$\widehat{y_{i,u}} = \frac{\sum_{u_j \in N(u,i)} \overline{y}_{i,u} sim(u, u_j)}{\sum_{u_j \in N(u,i)} |sim(u, u_j)|}$$

Trong $N(u,i)$ là tập hợp k user có similarity cao nhất với người dùng u mà đã đánh giá quyển sách i. Vì giá trị similarity có thể âm nên ở mẫu số sử dụng giá trị tuyệt đối.

2.3.2 Xây dựng danh sách gợi ý N quyển sách phù hợp với người dùng

User-Based Top-N recommendation

Phương pháp này dựa trên phương pháp user-based top-N recommendation được ghi ở trên nhưng đã được điều chỉnh để phù hợp cho bộ dữ liệu goodbooks

10k. Đối với phương pháp này, ma trận similarity được xây dựng giữa người dùng. Đầu tiên, tìm k người dùng giống nhất với người dùng muốn được gợi ý (active user) sử dụng ma trận similarity. Sau đó, tìm set C bao gồm tất cả những quyển sách mà k người dùng này đã đánh giá với giá trị sau khi chuẩn hóa lớn hơn 0. Sau đó thống kê số lần xuất hiện của mỗi quyển sách trong set C, sau đó sắp xếp theo thứ tự giảm dần của số lần xuất hiện của mỗi quyển sách trong set C. Lấy N giá trị đầu tiên, kết quả có N quyển sách được dùng để gợi ý cho active user.

Item-Based Top-N recommendation

Phương pháp này dựa trên phương pháp item-based top-N recommendation đã được để cập ở phần tổng quan, đã được điều chỉnh để phù hợp cho bộ dữ liệu goodbooks 10k. Ở phương pháp này, không xây dựng ma trận similarity lưu độ tương đồng giữa những người dùng mà xây dựng ma trận similarity lưu độ tương đồng giữa các quyển sách. Phương pháp này có ưu điểm hơn so với phương pháp user-based ở điểm là trong thực tế thường số lượng sách trong dữ liệu thường nhỏ hơn nhiều số lượng người dùng. Như trong bộ dữ liệu goodbooks 10k thì chỉ có 10000 quyển sách trong khi đó có hơn 53 nghìn người dùng. Các bước thực hiện phương pháp này như sau. Đầu tiên, tìm set C, là hợp của các tập k quyển sách giống nhất với những quyển sách mà active user thích (có đánh giá > 0 sau khi đã chuẩn hóa) dựa trên ma trận similarity. Sau đó xóa những sản phẩm mà active user đã đánh giá, thống kê số lần xuất hiện của những quyển sách còn lại trong set C. Sắp xếp theo số lần xuất hiện giảm dần, lấy N item đầu tiên. Kết quả được N quyển sách được gợi ý cho active user.

2.4 Tiếp cận sử dụng phương pháp Matrix Factorization Collaborative Filtering

2.4.1 Xây dựng mô hình

Việc tiếp cận bài toán sử dụng phương pháp MFCF được thực hiện theo các bước như sau:

Bước đầu tiên cần phải thực hiện chính là chuẩn hóa dữ liệu, việc chuẩn hóa sẽ được thực hiện bằng cách trừ mỗi hàng của ma trận utility đi trung bình cộng của các giá trị đã biết của hàng đó hoặc trừ đi một cột giá trị trung bình cộng của các giá trị đã biết trong cột đó. Sau đó cần phải khởi tạo hai ma trận $X \in \mathbb{R}^{M \times K}$ và $W \in \mathbb{R}^{K \times N}$. Trong đó thì M là số lượng quyển sách và N là số lượng người dùng.

Từ đó thấy được X là ma trận của toàn bộ các quyển sách, mỗi hàng của X tương ứng với một quyển sách, còn W là ma trận của toàn bộ người dùng, mỗi cột của W tương ứng với một người dùng. Việc cần làm tiếp theo là cố gắng xấp xỉ ma trận Utility $Y \in \mathbb{R}^{M \times N}$ bằng tích của hai ma trận $X \in \mathbb{R}^{M \times K}$ và $W \in \mathbb{R}^{K \times N}$.
Khi đó hàm mất mát sẽ có dạng như sau:

$$L(X,W) = \frac{1}{2s}\sum_{n=1}^{N}\sum_{m:r_{mn}=1}(y_{mn} - x_m w_n)^2 + \frac{\lambda}{2}(\|X\|_F^2 + \|W\|_F^2) \tag{2.1}$$

Việc tối ưu hàm mất mát trên sẽ được thực hiện bằng cách cố định W và tối ưu X sau đó thực hiện cố định X và tối ưu W. Hai bước này được lặp lại cho tới khi điều kiện dừng được thỏa mãn. Áp dụng gradient descent để tối ưu hàm mất mát.
Công thức cập nhật mỗi cột của W là:

$$w_n = w_n - \eta(-\frac{1}{s}\hat{X}_n^T(\hat{y}_n - \hat{X}_n w_n) + \lambda w_n)$$

Công thức cập nhật mỗi hàng của X là:

$$x_m = x_m - \eta(-\frac{1}{s}\hat{W}_m^T(\hat{y}_m - x_m \hat{W}_m) + \lambda x_m)$$

Cập nhật X và W

Đối với phương pháp này, mỗi một vòng lặp sẽ cố định W và thực hiện cập nhật tất cả các hàng của X theo công thức ở trên. Sau đó cố định X và cập nhật tất cả các cột của W. Hai việc trên được lặp đi lặp lại cho đến khi điều kiện dừng được thỏa mãn.

Điều kiện dừng

Khi điều kiện dừng được thỏa mãn, việc tối ưu hàm mất mát sẽ được dừng lại. Trong hướng tiếp cận này có hai điều kiện dừng chính. Điều kiện dừng thứ nhất là khi đạt đến số vòng lặp tối đa. Số vòng lặp tối đa sẽ được chọn từ trước. Điều kiện thứ hai là giá trị của hàm mất mát giảm đi sau n vòng lặp nhỏ hơn một giá trị nhất định. Giá trị này cũng sẽ được chọn từ trước. Sau mỗi n vòng lặp, giá trị của hàm mất mát sẽ được tính toán lại, sau đó tính toán hiệu số của giá trị mới được tính này với giá trị hàm mất mát được tính n vòng lặp trước đó. Nếu giá trị này nhỏ hơn 1 giá trị nhất định cho trước, việc tối ưu hàm mất mát sẽ được dừng lại, nếu không thì việc tối ưu hàm mất mát sẽ tiếp tục. Trong hai điều kiện trên, điều kiện, bất kì điều kiện nào được thỏa mãn trước thì việc tối ưu hàm mất mát đều sẽ được dừng

lại. Đối với phương pháp cập nhật chỉ 1 cột hay 1 hàng của W và X, số vòng lặp tối đa hay giá trị n thường sẽ lớn hơn rất nhiều so với phương pháp cập nhật toàn bộ X và W.

2.4.2 Dự đoán đánh giá của người dùng với sách

Sau khi đã tối ưu hàm mất mát, có thể sử dụng X và W để dự đoán đánh giá của một người dùng đối với một quyển sách. Ví dụ như khi muốn tìm đánh giá của người dùng n cho quyển sách m, đơn giản chỉ cần tính tích vô hướng của 2 vector x_m và w_n, trong đó x_m là hàng thứ m trong ma trận X và w_m là cột thứ n trong ma trận W. Vì khối lượng tính toán ít và đơn giản nên có thể đưa ra kết quả rất nhanh mà không tốn nhiều tài nguyên tính toán.

2.4.3 Xây dựng danh sách gợi ý N quyển sách phù hợp nhất cho người dùng

Để tìm ra N quyển sách có thể dùng để gợi ý cho người dùng m, việc đầu tiên cần làm là điền toàn bộ giá trị còn khuyết của người dùng m trong ma trận utility. Sau đó sắp xếp tất cả những quyển sách mới được dự đoán đánh giá theo giá trị giảm dần của đánh giá của người dùng m cho quyển sách đó. Chọn N quyển sách đầu tiên, sẽ có được những quyển sách có thể dùng để gợi ý cho người dùng m.

2.5 Tiếp cận sử dụng phương pháp Clustering Collaborative Filtering

2.5.1 Xây dựng mô hình

Mục đích của hướng tiếp cận này là phân người dùng vào các cụm khác nhau để khi thực hiện việc dự đoán dựa trên đánh giá của những người dùng tương tự có thể thực hiện chỉ trong không gian cụm chứ không phải toàn bộ dữ liệu. Với hướng tiếp cận sử dụng phương pháp clustering, việc áp dụng được thực hiện theo các bước dưới đây.

Bước đầu tiên cần làm là thực hiện chuẩn hóa dữ liệu của ma trận utility bằng cách trừ mỗi hàng hoặc mỗi cột đi trung bình cộng của các giá trị đã biết trong hàng hay cột đó. Sau đó cần thực hiện phân người dùng vào các cụm và tìm ra tâm của mỗi cụm theo các bước sau.

Đầu tiên cần chọn số lượng cụm muốn chia (k). Giá trị k này sẽ được chọn từ trước. Sau đó, cần chọn ra k tâm cụm cho các cụm. Việc này có thể thực hiện theo 2 cách. Hoặc là k tâm cụm này sẽ được chọn từ trước, hoặc là có thể chọn bằng cách chọn ngẫu nhiên k người dùng khác nhau để làm k tâm cụm. Tuy nhiên nếu dùng cách chọn ngẫu nhiên từ người dùng, cần phải chọn người dùng đã từng có lịch sử đánh giá sách để tránh việc tâm của các cụm trùng nhau và đều bằng 0.
Sau khi đã chọn được tâm cụm. Việc tiếp theo là phân các người dùng vào các cụm. Người dùng thuộc về cụm nào sẽ được quyết định bằng cách tìm xem người dùng đó gần tâm cụm nào nhất, khi đó sẽ gán nhãn người dùng đó thuộc vào cụm đó. Trong phần ứng dụng, việc đó khoảng cách giữa người dùng và tâm cụm sẽ được tính sử dụng cosine similarity.

$$similarity_{cosine}(u_1, u_2) = \cos u_1, u_2 = \frac{u_1^T u_2}{\|u_1^T\|.\|u_2\|}$$

Giá trị cosine similarity càng lớn chứng tỏ người dùng càng gần tâm cụm. Sau khi đã phân người dùng vào các cụm, việc tiếp theo cần làm là tính toán lại tâm cụm. Tâm cụm được tính toán bằng cách lấy trung bình cộng của tọa độ tất cả các người dùng thuộc về cụm đó trong không gian. Sau đó kiểm tra điều kiện hội tụ, nếu điều kiện hội tụ không được thỏa mãn, lặp lại hai bước phân cụm và tính toán lại tâm cụm, nếu điều kiện hội tụ được thỏa mãn thì dừng việc phân cụm trong người dùng lại.

Điều kiện hội tụ được áp dụng trong phần ứng dụng là tâm cụm không thay đổi so với lần tính toán lại tâm cụm trước đó. Nếu như sau khi tính toán tâm cụm, xếp lại người dùng vào các cụm rồi tính toán lại các tâm cụm một lần nữa mà tâm cụm vẫn không có thay đổi gì sẽ được coi như là đã hội tụ.

Xây dựng danh sách gợi ý cho mỗi cụm

Bước tiếp theo là xây dựng danh sách những quyển sách có thể gợi ý cho người dùng ở trong cụm. Khi xây dựng được danh sách này rồi thì khi muốn tìm ra N quyển sách để gợi ý cho người dùng m, chỉ cần lấy danh sách những quyển sách gợi ý của cụm, loại bỏ những quyển sách mà người dùng m đã đánh giá sau đó lấy N quyển sách đầu tiên. Để xây dựng danh sách gợi ý cho cụm k, đầu tiên xây dựng set C gồm tất cả những quyển sách được người dùng thuộc cụm k đánh giá với giá trị lớn hơn 0 (sau khi đã chuẩn hóa). Sau đó xây dựng danh sách những quyển sách thuộc set C cùng với số lần xuất hiện của những quyển sách đó trong set C. Sắp xếp danh sách trên theo giá trị từ cao đến thấp của số lần xuất hiện của quyển sách đó trong set C, từ đó có được danh sách gợi ý cho cụm k.

2.5.2 Dự đoán đánh giá của người dùng cho sách

Để dự đoán đánh giá của người dùng m cho quyển sách n, cần thực hiện những bước sau đây. Đầu tiên là xác định cụm k mà người dùng m thuộc về. Coi tâm cụm k như một người dùng, khi đó đánh giá của người dùng này cho quyển sách n sẽ chính là giá trị đánh giá của người dùng m cho quyển sách n. Do đó, có thể đưa ra được dự đoán đánh giá của người dùng rất nhanh mà không cần phải qua tính toán gì.

2.5.3 Xây dựng danh sách gợi ý cho người dùng

Để xây dựng danh sách gợi ý N quyển sách phù hợp với người dùng m, việc đầu tiên cần phải làm là xác định người dùng m thuộc cụm nào và lấy danh sách gợi ý của cụm đó. Loại bỏ những quyển sách người dùng m đã đánh giá ra khỏi danh sách trên, sau đó lấy N quyển sách đầu tiên trong danh sách. Từ đó được danh sách N quyển sách có thể gợi ý cho người dùng.

Chương 3

Kết quả thực nghiệm và so sánh

Môi trường thực hiện thực nghiệm là Google Colab. Ngôn ngữ sử dụng là Python. Với mỗi phương pháp sẽ có kịch bản thực nghiệm khác nhau, sau đó sẽ so sánh kết quả giữa các phương pháp này. Trong phần ứng dụng, việc xây dựng danh sách gợi ý cho người dùng luôn đưa ra danh sách 10 quyển sách phù hợp nhất với người dùng. Thực nghiệm được thực hiển trên cơ sở dữ liệu goodbooks 10k. Tuy nhiên do vấn đề giới hạn bộ nhớ của google colab, chỉ có thể sử dụng một phần của bộ dữ liệu goodbooks, bao gồm 2 triệu đánh giá, 10000 quyển sách và 48334 người dùng.

Một số chỉ số được sử dụng để được sử dụng để so sánh, đánh giá các mô hình được thực nghiệm là Root Mean Squared Error (*RMSE*), Hit Rate, Average Reciprocal Hit Ranking (*ARHR*), Intra-List Simlarity (*ILS*).

- Root Mean Square Error dùng để đánh giá sự sai lệch giữa đánh giá được dự đoán và đánh giá thực. *RMSE* càng nhỏ chứng tỏ dự đoán càng chính xác.

$$RMSE = \sqrt{\frac{1}{n}\sum_{\{i,j\}}(p_{i,j} - r_{i,j})^2} \tag{3.1}$$

  Trong đó n là tổng số lượng đánh giá của tất cả người dùng, $p_{i,j}$ là giá trị đánh giá được dự đoán của người dùng i đối với sản phẩm j, còn $r_{i,j}$ là giá trị đánh giá thực của người dùng i đối với sản phẩm j.

- Hit rate dùng để đánh giá danh sách gợi ý có phù hợp với người dùng không. Hit rate càng lớn chứng tỏ danh sách gợi ý càng chính xác. Thực hiện bằng cách loại bỏ đi một đánh giá tốt của người dùng ra khỏi tập huấn luyện, nếu quyển sách đó xuất hiện trong danh sách gợi ý thì được coi là một hit.

$$HR = \frac{total\ hit}{total\ user} \tag{3.2}$$

- ARHR giống như hit rate tuy nhiên đánh giá thêm việc quyển sách được người dùng thích có nằm ở vị trí gần đầu trong danh sách gợi ý hay không.

$$ARHR = \frac{\sum_{i=1}^{n} \frac{1}{rank_i}}{total\ user} \tag{3.3}$$

  Với $rank_i$ là vị trí của hit i trong top N recommendation. Ví dụ như hit ở vị trí 1 sẽ có giá trị là 1 nhưng hit ở vị trí 3 thì sẽ chỉ có giá trị là $\frac{1}{3}$.

- Intra-List Simlarity tính mức độ giống nhau của các quyển sách trong danh sách gợi ý. *ILS* càng lớn tức các quyển sách trong danh sách gợi ý càng giống nhau, tương đương với mức độ đa dạng càng thấp.

$$ILS_u = \frac{1}{2} \sum_{i_j \in L} \sum_{i_k \in L; i_j \neq i_k} sim(i_k, i_j) \tag{3.4}$$

  L là danh sách những sản phẩm được gợi ý cho người dùng u. $sim(i_k, i_j)$ là phép tính độ tương đồng giữa hai sản phẩm i_k và i_j sử dụng cosine similarity.

Trong bảng của phần thực nghiệm, *HR* là hit rate, *ARHR* là average reciprocal hit ranking, t_1 là thời gian xây dựng mô hình, t_2 là thời gian dự đoán 400000 đánh giá của tập dữ liệu kiểm tra, t_3 là thời gian xây dựng 23451 danh sách gợi ý sách cho người dùng, *ILS* là chỉ số Intra-List Similarity. Thời gian trong bảng có giá trị là giây.

3.1 Thực nghiệm phương pháp neighborhood-based collaborative filtering

Các kịch bản thực nghiệm:

- Thay đổi số lượng người dùng tương tự cần thiết để thực hiện dự đoán đối với phương pháp user-based
- Thay đổi số lượng quyển sách tương tự cần thiết để thực hiện dự đoán đối với phương pháp item-based

3.1.1 Kịch bản thay đổi số lượng người dùng tương tự đối với hướng tiếp cận user-based

Kết quả thực nghiệm của kịch bản thay đổi số lượng người dùng tương tự đối với hướng tiếp cận user-based được thể hiện trong bảng 3.1. Trong bảng này, k là số lượng người dùng tương tự cần được tìm.

Theo như bảng 3.1 có thể thấy khi k càng lớn thì hit rate và ARHR càng lớn, RMSE có xu hướng giảm. Tuy vậy thì thời gian đưa ra dự đoán đánh giá và xây dựng danh sách gợi ý tăng theo, đặc biệt là thời gian xây dựng danh sách gợi ý tăng lên rất nhanh. Nguyên nhân ở đây là do khi số lượng người dùng tương tự cần tìm càng lớn thì danh sách những quyển sách ứng cử có thể gợi ý càng dài, từ đó việc sắp xếp danh sách gợi ý này càng tốn nhiều thời gian.

Bảng 3.1: Kết quả thực nghiệm kịch bản thay đổi số người dùng tương tự user-based

k	*RMSE*	*HR*	*ARHR*	t_1	t_2	t_3	*ILS*
10	0.8721	0.1692	0.0928	161.92	1372.13	436.16	2.2528
20	0.8554	0.2151	0.1165	151.6	1398.96	839.48	3.2896
30	0.8504	0.2333	0.1247	162.16	1449.08	1156.49	3.5745
40	0.8485	0.2414	0.1293	156.0	1402.95	1507.72	3.9813
50	0.8475	0.2494	0.133	161.78	1442.34	1871.14	4.3441
80	0.8471	0.254	0.1369	161.89	1448.0	2869.66	4.9292
100	0.8473	0.2554	0.1375	160.15	1461.87	3800.48	5.1417
150	0.8484	0.2561	0.1381	163.49	1504.09	5651.21	5.4921
200	0.8494	0.2556	0.1377	179.02	1534.89	7287.2	5.7295

3.1.2 Kịch bản thay đổi số lượng người dùng tương tự đối với hướng tiếp cận item-based

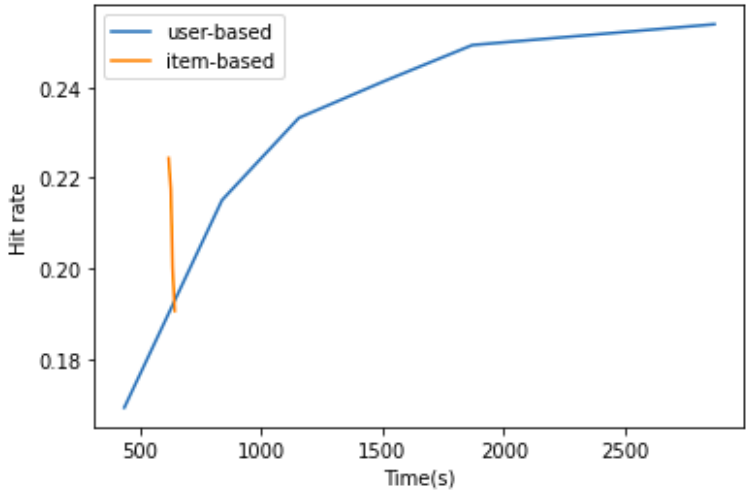
Kết quả thực nghiệm của kịch bản thay đổi số lượng người dùng tương tự đối với hướng tiếp cận user-based được thể hiện trong bảng 3.2. Trong bảng này, k là số lượng người dùng tương tự cần được tìm.

Bảng 3.2: Kết quả thực nghiệm kịch bản thay đổi số sản phẩm tương tự item-based

k	*RMSE*	*HR*	*ARHR*	t_1	t_2	t_3	*ILS*
10	0.8223	0.2244	0.1041	9.17	405.41	619.19	2.5325
20	0.8178	0.2211	0.0979	9.36	407.01	623.8	3.448
30	0.819	0.2175	0.0972	9.26	404.89	628.07	3.7371
40	0.8206	0.2079	0.0889	9.26	408.5	632.4	3.694
50	0.822	0.2004	0.0844	9.2	405.48	635.57	3.589
80	0.8254	0.1906	0.0779	9.34	408.2	644.18	3.3732
100	0.8271	0.1903	0.0778	9.24	406.95	655.67	3.2073
150	0.8286	0.1795	0.0721	9.32	409.33	672.5	3.0125
200	0.8286	0.175	0.0679	9.37	414.23	688.15	2.9357

3.1.3 So sánh 2 hướng tiếp cận user-based và item-based

Theo như bảng 3.1 và 3.2, thấy được RMSE của hướng tiếp cận item-based thấp hơn so với hướng tiếp cận user-based. Tuy vậy, hướng tiếp cận user-based lại cho hit rate và ARHR cao hơn khi k lớn hơn 20.



Hình 3.1: So sánh hit rate của hai phương pháp user-based và item-based dựa trên thời gian xây dựng danh sách gợi ý.

Ngoài ra, có thể thấy được về vấn đề thời gian xây dựng mô hình, thời gian dự đoán đánh giá của người dùng cho sách và thời gian xây dựng danh sách gợi ý thì hướng tiếp cận item-based có thời gian ngắn hơn rất nhiều. Theo như hình 3.1, khi xây dựng danh sách gợi ý, hướng tiếp cận user-based tuy có hit rate cao hơn nhưng thời gian xây dựng danh sách gợi ý lại lớn hơn rất nhiều, đặc biệt là khi k lớn. Xét đến ILS, khi k không quá lớn, chỉ nằm trong khoảng 10 đến 30 thì ILS của hai phương pháp không quá khác nhau. Tuy nhiên khi k lớn thì ILS của phương pháp user-based ngày càng tăng tức mức độ đa dạng của những quyển sách gợi ý càng giảm, còn phương pháp item-based thì *ILS* còn giảm đi tương đương với mức độ đa dạng tăng thêm.

So sánh hai phương pháp có thể thấy phương pháp item-based có ứng dụng thực tế tốt hơn. Lí do là vì độ chính xác của phương pháp item-based không thua kém của phương pháp user-based, tuy vậy trong thực tế, lượng người dùng lớn hơn rất nhiều so với 50000 người dùng của bộ cơ sở dữ liệu được dùng trong thực nghiệm, từ đó dẫn đến phương pháp user-based sẽ tốn bộ nhớ hơn rất nhiều. Ngoài ra trong thực tế, hiệu năng hệ thống là điều rất quan trọng, đòi hỏi tốc độ đưa ra kết quả nhanh.

3.2 Thực nghiệm phương pháp matrix-factorization collaborative filtering

Các kịch bản thực nghiệm:

- Thay đổi số lượng tính chất ẩn k
- Thay đổi learning rate

3.2.1 Kịch bản thay đổi số lượng tính chất ẩn

Kết quả thực nghiệm của kịch bản thay đổi số lượng tính chất ẩn được thể hiện ở bảng 3.3. Trong đó k là số lượng tính chất ẩn. Tham số learning rate trong kịch bản này là 10.

Theo như bảng 3.3 có thể thấy việc thay đổi số lượng tính chất ẩn không tạo ra nhiều thay đổi tới RMSE, hit rate, ARHR hay ILS. Tuy nhiên việc tăng số lượng tính chất ẩn dẫn đến việc tăng thời gian huấn luyện cũng như việc lưu trữ hai ma trận X và W sẽ tốn nhiều bộ nhớ hơn.

Bảng 3.3: Kết quả thực nghiệm MFCF khi thay đổi số tính chất ẩn

k	$RMSE$	HR	$ARHR$	t_1	t_2	t_3	ILS
5	0.9178	0.1749	0.036	2919.83	1.45	880.69	2.0227
8	0.9178	0.1749	0.036	2976.32	1.47	912.09	2.0227
10	0.9175	0.1749	0.036	3329.01	1.56	935.25	2.0227
12	0.9175	0.1749	0.036	3350.96	1.58	928.35	2.0227
15	0.9175	0.1749	0.036	3502.87	1.6	931.18	2.0227
17	0.9175	0.1749	0.036	3223.93	1.56	929.02	2.0227
20	0.9175	0.1749	0.036	3227.33	1.58	937.55	2.0227
30	0.9196	0.1749	0.036	3342.56	1.6	942.14	2.0227

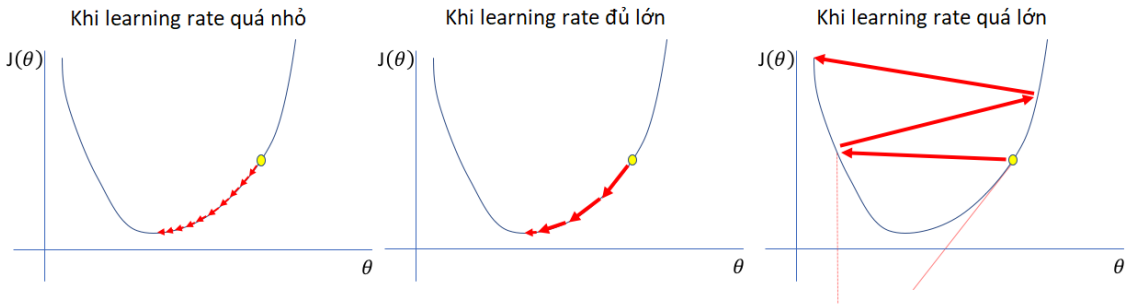
3.2.2 Kịch bản thay đổi learning rate

Kịch bản này sử dụng mô hình tương tự như kịch bản trên tuy nhiên thay vì thay đổi số lượng tính chất ẩn, kịch bản này xét trường hợp thay đổi learning rate. Kết quả của kịch bản được thể hiện trong bảng 3.4. Trong kịch bản này, số lượng tính chất ẩn được chọn bằng 10, lr trong bảng 3.4 là learning rate.

Bảng 3.4: Kết quả thực nghiệm MFCF khi thay đổi learning rate

lr	$RMSE$	HR	$ARHR$	t_1	t_2	t_3
20	1.9568	0.1066	0.0242	1909.56	2.29	970.48
15	0.9189	0.1749	0.036	5466.12	1.62	960.97
12	0.9175	0.1749	0.036	4002.02	1.6	923.18
10	0.9171	0.1749	0.036	3873.36	1.63	978.33
8	0.9171	0.1749	0.036	4594.68	1.63	992.76]
4	0.9171	0.0017	0.0004	6654.43	1.73	1005.55
2	0.9184	0.0024	0.0008	8738.37	1.63	989.08

Qua bảng 3.4 có thể thấy learning rate ảnh hưởng rất nhiều đến quá trình tối ưu hàm mất mát. Nếu learning rate quá cao chẳng hạn như trường hợp learning rate bằng 20, dẫn đến trường hợp giá trị của hàm mất mát tăng thêm chứ không hề giảm đi. Hay nếu như learning rate nhỏ như trường hợp learning rate bằng 4 hay bằng 2, điều kiện dừng có thể được thỏa mãn trước khi hàm mất mát được tối ưu, từ đó dẫn đến RMSE cao hơn, Hit rate và ARHR rất thấp. Các trường hợp trên được thể hiện ở hình 3.2.



Hình 3.2: Các trường hợp tối ưu hàm mất mát theo learning rate [4]

3.3 Thực nghiệm phương pháp clustering collaborative filtering

Các kịch bản thực nghiệm:

- Thay đổi số lượng cụm

Theo như kịch bản thay đổi số lượng cụm, kết quả của thay đổi của RMSE, thời gian xây dựng mô hình và thời gian dự đoán đánh giá theo kịch bản này được thể hiện ở bảng 3.5. Các tâm cụm ban đầu được chọn ngẫu nhiên từ các người dùng.

Bảng 3.5: Kết quả thực nghiệm phương pháp clustering

k	*RMSE*	*HR*	*ARHR*	t_1	t_2	t_3	*ILS*
20	0.908	0.1749	0.0792	248.94	1.03	74.51	0.8577
50	0.9073	0.1749	0.0792	260.9	1.01	63.01	0.8596
80	0.9074	0.175	0.0793	263.52	1.02	57.39	0.8533
100	0.9068	0.175	0.0793	257.16	1.01	55.92	0.8306
150	0.9064	0.175	0.079	270.07	1.02	53.27	0.8382
200	0.9063	0.1743	0.0791	272.4	1.02	50.6	0.8059
300	0.9063	0.1739	0.0788	288.99	1.02	49.23	0.8128
400	0.9066	0.1744	0.0792	348.79	1.09	48.38	0.7908
700	0.9083	0.172	0.078	367.83	1.0	50.01	0.7734
1000	0.908	0.1702	0.0774	398.14	0.98	46.85	0.7769

Theo như bảng 3.5 có thể thấy RMSE, hit rate và ARHR không bị ảnh hưởng quá nhiều bởi thay đổi của số lượng cụm. Khi số lượng cụm được chia tăng lên, thời gian huấn luyện mô hình tăng lên do phải thực hiện tính toán khoảng cách từ người dùng tới tâm cụm nhiều hơn và phải tính lại tâm cụm nhiều hơn; đồng thời việc xây dựng danh sách gợi ý tốn ít thời gian hơn do danh sách gợi ý của mỗi cụm ngắn hơn. Ngoài ra, ILS giảm tức mức độ đa dạng có xu hướng tăng lên tuy nhiên không đáng kể.

3.4 So sánh kết quả thực nghiệm các phương pháp

Bảng 3.6 giúp so sánh kết quả thực nghiệm của các phương pháp. Các kết quả được đưa vào bảng có được từ các tham số như sau. Phương pháp user-based dựa trên đánh giá 30 người dùng tương tự để đưa ra dự đoán, phương pháp item-based dựa trên kết quả đánh giá của 20 quyển sách tương tự, phương pháp MFCF chọn 8 tính chất ẩn và learning rate bằng 10, phương pháp clustering chọn số cụm bằng 80.

Trong các phương pháp đã cho, phương pháp item-based là phương pháp cho kết quả RMSE tốt nhất, thời gian xây dựng mô hình ngắn nhất vì số lượng quyển sách không quá nhiều. Phương pháp user-based có hit rate và ARHR cao nhất, tuy vậy phương pháp này lại có thời gian dự đoán kết quả đánh giá và thời gian xây dựng danh sách gợi ý lớn nhất. Phương pháp MFCF có RMSE cao nhất, hit rate và ARHR thấp nhất trong các phương pháp, có thời gian xây dựng mô hình lâu nhất, tuy nhiên phương pháp này này có thời gian dự đoán đánh giá ngắn. Cuối cùng, phương pháp clustering có RMSE thấp hơn, hit rate tương đương với phương pháp MFCF, tuy vậy thời gian xây dựng mô hình của phương pháp ngắn hơn rất nhiều,

hơn nữa thời gian dự đoán đánh giá và thời gian xây dựng danh sách gợi ý ngắn nhất trong các phương pháp. Xét đến mức độ đa dạng của danh sách gợi ý ở mỗi phương pháp, có thể thấy phương pháp clustering cho kết quả *ILS* thấp nhất tương đương với có danh sách gợi ý đa dạng nhất. Phương pháp đưa ra danh sách gợi ý ít đa dạng nhất là user-based.

Bảng 3.6: So sánh kết quả thực nghiệm của các phương pháp

Hướng tiếp cận	*RMSE*	*HR*	*ARHR*	t_1	t_2	t_3	*ILS*
User-based	0.8504	0.2333	0.1247	162.16	1449.08	1156.49	3.5745
Item-based	0.8178	0.2211	0.0979	9.36	407.01	623.8	3.448
MFCF	0.9178	0.1749	0.036	2976.32	1.47	912.09	2.0227
Clustering	0.9074	0.175	0.0793	263.52	1.02	57.39	0.8533

Kết luận

Trong đồ án này, bài toán xây dựng hệ thống gợi ý sách được tiếp cận bằng 3 phương pháp là NBCF, MFCF và clustering, trong đó phương pháp NBCF được chia thành hai phương pháp nhỏ hơn là user-based và item-based. Theo như kết quả thực nghiệm, hướng tiếp cận user-based đang là phương pháp cho kết quả hit rate và ARHR tốt nhất, tuy nhiên phương pháp này lại tốn rất nhiều thời gian để dự đoán đánh giá và xây dựng danh sách gợi ý. Ngoài ra phương pháp này còn có nhược điểm rất lớn là khi số lượng người dùng quá lớn, việc xây dựng ma trận similarity sẽ tốn rất nhiều bộ nhớ. Để giúp tránh được vấn đề này, có thể áp dụng phương pháp item-based. Trong thực tế, số lượng sản phẩm hay cụ thể là số lượng sách của một hệ thống gợi ý thường nhỏ hơn rất nhiều so với số lượng người dùng, do đó việc xây dựng ma trận similarity sẽ tốn ít bộ nhớ hơn và có thể xây dựng mô hình nhanh hơn. Có thể thấy trong kết quả thực nghiệm, phương pháp item-based cho kết quả RMSE tốt nhất, hit rate và ARHR chỉ kết hơn so với phương pháp user-based một chút, trong khi đó thì gian xây dựng model, dự đoán đánh giá và xây dựng danh sách gợi ý thì nhỏ hơn phương pháp user-based rất nhiều. Tuy vậy, cả hai phương pháp trên đều là phương pháp memory-based, phụ thuộc vào bộ nhớ rất nhiều.

Hai phương pháp MFCF và clustering CF đều là phương pháp model-based, hai phương pháp này đều phải thực hiện quá trình huấn luyện mô hình, giúp tiết kiệm bộ nhớ hơn so với phương pháp memory-based. Do đó, cả hai phương pháp đều có thời gian huấn luyện cao hơn hẳn phương pháp NBCF, có thời gian dự đoán đánh giá thấp hơn nhiều. Phương pháp clustering có RMSE tốt hơn một chút, hit rate tương đương với phương pháp MFCF tuy nhiên có thời gian huấn luyện và thời gian xây dựng danh sách gợi ý thấp hơn hẳn. Hướng tiếp cận model-based có lợi thế là khi số lượng người dùng và sách rất lớn vẫn có thể lưu trữ để thực hiện gợi ý mà không tốn quá nhiều bộ nhớ và không mất nhiều thời gian để đưa ra dự đoán. Chẳng hạn như phương pháp MFCF thì chỉ cần lưu trữ hai ma trận $X \in R^{M \times K}$ và $W \in R^{K \times N}$, K thông thường được chọn là số nhỏ hơn M, N rất nhiều. Để đưa ra dự đoán chỉ cần lấy tích vô hướng của hai vector $x^T w$. Phương pháp clustering chỉ cần thực hiện lưu trữ mỗi người dùng thuộc cụm nào và tâm của mỗi cụm. Tuy nhiên hai phương pháp này gặp phải vấn đề là khi có người dùng mới, sách mới, đánh giá

mới sẽ cần phải thực hiện lại việc huấn luyện, điều này có thể dẫn đến tốn thời gian.

Trong thực tế, việc có thể đưa ra được danh sách gợi ý những quyển sách đúng như sở thích người dùng vẫn là chưa đủ, cần có thể đưa ra danh sách bao gồm những quyển sách thuộc nhiều thể loại khác nhau, giúp người dùng có thể tiếp cận được với những thể loại mới. Do vậy mức độ đa dạng của hệ gợi ý cũng rất quan trọng. Trong phần thực nghiệm, mức độ đa dạng này được đánh giá qua chỉ số ILS, chỉ số này càng thấp cho thấy mức độ đa dạng càng cao. Qua kết quả thực nghiệm có thể thấy phương pháp clustering là phương pháp cho kết quả có mức độ đa dạng cao nhất, giúp người dùng tiếp cận tới nhiều thể loại sách. Nhưng phương pháp còn lại có mức độ đa dạng thấp hơn, thấp nhất là phương pháp user-based với chỉ số ILS cao nhất.

Tài liệu tham khảo

[1] Daniel Billsus and Michael Pazzani. Ahybrid user model for news story classification. 12 2001.

[2] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12, 11 2002.

[3] Minh-Phung Do, Dung Nguyen, and Academic Network of Loc Nguyen. Model-based approach for collaborative filtering. 08 2010.

[4] Jeremy Jordan. Setting the learning rate of your neural network.

[5] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Recommendation Systems*, page 292–324. Cambridge University Press, 2 edition, 2014.

[6] Rejoiner. The amazon recommendations secret to selling more online.

[7] Vũ Hữu Tiệp. *Machine Learning cơ bản*. 2018.

[8] Cai-Nicolas Ziegler, Cai-Nicolas, Sean McNee, Sean M, Konstan, Joseph A, Lausen, and Georg. Improving recommendation lists through topic diversification. 01 2005.