VIETNAM GENERAL CONFEDERATION OF LABOR
TON DUC THANG UNIVERSITY
INFORMATION TECHNOLOGY FACULTY

MACHINE LEARNING

Midterm Report

RECOMMENDATION SYSTEMS

Instructor: **Prof. Lê Anh Cường**

Student 1: **Đỗ Phạm Quang Hưng - 520K0127**

Student 2: **Lê Phước Thịnh - 520K0343**

HO CHI MINH CITY, 2023

Table of Contents

I. Introduction

Có lẽ ai cũng đã gặp những hiện tượng này nhiều lần:

- Youtube tự động chuyển các clip liên quan đến clip bạn đang xem. Youtube cũng tự gợi ý những clip mà có thể bạn sẽ thích.
- Khi bạn mua một món hàng trên Amazon, hệ thống sẽ tự động gợi ý "Frequently bought together", hoặc nó biết bạn có thể thích món hàng nào dựa trên lịch sử mua hàng của bạn.
- Facebook hiển thị quảng cáo những sản phẩm có liên quan đến từ khoá bạn vừa tìm kiếm.
- Facebook gợi ý kết bạn.
- Netflix tự động gợi ý phim cho người dùng.

Và rất nhiều ví dụ khác mà hệ thống có khả năng tự động gợi ý cho người dùng những sản phẩm họ có thể thích. Bằng cách quảng cáo hướng đúng đối tượng như thế này, hiệu quả của việc marketing cũng sẽ tăng lên. Những thuật toán đằng sau những ứng dụng này là những thuật toán Machine Learning có tên gọi chung là Recommender Systems hoặc Recommendation Systems, tức Hệ thống gợi ý.

Recommendation Systems là một mảng khá rộng của Machine Learning và có tuổi đời ít hơn so với Classification vì internet mới chỉ thực sự bùng nổ khoảng 10-15 năm đổ lại đây. Có hai thực thể chính trong Recommendation Systems là users và items. Users là người dùng. Items là sản phẩm, ví dụ như các bộ phim, bài hát, cuốn sách, clip, hoặc cũng có thể là các users khác trong bài toán gợi ý kết bạn. Mục đích chính của các Recommender Systems là dự đoán mức độ quan tâm của một user tới một item nào đó, qua đó có chiến lược recommend phù hợp.

1.1. Vai trò của Recommendation System?

Recommendation hiểu một cách đơn giản là khuyến nghị một sản phẩm đến đúng người cần mua. Giả sử công ty của bạn hoạt động ở Việt Nam. Điều đó không có nghĩa rằng 95 triệu người Việt sẽ mua hàng của công ty bạn mà chỉ một phần trong số đó có nhu cầu. Trong số các khách hàng có nhu cầu thì không phải ai cũng sẽ mua tất cả các sản phẩm của công ty mà họ có khi chỉ mua một vài sản phẩm mà họ quan tâm.

Một công ty nếu muốn tối đa hóa lợi nhuận thì điều quan trọng nhất họ phải hiểu được khách hàng họ cần gì? Recommendation là một thuật toán kì diệu có thể giúp bạn thực hiện điều đó.

Hãy tưởng tượng tình hình kinh doanh sẽ ra sao nếu không có thuật toán này? Một loạt các hệ quả mà ta có thể hình dung ra:

- Công ty không thể tìm được đúng khách hàng tiềm năng khi người có nhu cầu đối với một sản phẩm lại không được chào bán. Người không có nhu cầu lại bị tiếp cận mời chào. Điều này gây lãng phí thời gian và dẫn tới mất thiện cảm của khách hàng về dịch vụ của công ty.
- Hiệu quả marketing gần như là không đáng kể nếu không tìm đúng tập khách hàng. Chi phí quảng cáo, chi phí cho nhân viên sale tăng lên nhưng doanh thu vẫn thế. Theo một nghiên cứu, một số doanh nghiệp sẵn sàng bỏ ra từ 30-40% lợi nhuận cho việc marketing. Đây là một chi phí không hề nhỏ nhưng để tồn tại họ không thể ngừng đốt tiền. Cuối cùng người hưởng lợi nhiều nhất lại là google, facebook.

Chính vì thế ngày nay thuật toán recommendation được phát triển và ứng dụng rộng rãi trong nhiều doanh nghiệp thuộc đa dạng các lĩnh vực khác nhau như thương mại điện tử, tài chính, ngân hàng, kinh doanh, bán lẻ, phim ảnh,….

1.2. Các phương pháp recommendation

Đối với những bạn đã quen thuộc với recommendation thì các phương pháp tôi sắp giới thiệu sẽ không còn mới lạ, đối với những bạn mới tiếp xúc đây có lẽ là những thông tin hữu ích. Lịch sử ra đời sau internet, recommendation chỉ mới xuất hiện và phát triển cách đây khoảng 10 năm. Mặc dù có tuổi đời còn khá trẻ nhưng recommendation là một lĩnh vực phát triển rất nhanh, có số lượng bài báo khoa học lớn và thu hút được nhiều nhà nghiên cứu. Theo trường phái machine learning cổ điển, recommendation sẽ bao gồm 2 nhánh chính:

- **Content-based**: Đưa ra các khuyến nghị mua bán cho người dùng dựa trên nội dung liên quan đến sản phẩm. Chẳng hạn một bài hát với các đặc điểm như: người biểu diễn - Xuân Mai, năm phát hành - 2002, thể loại nhạc thiếu nhi sẽ phù hợp với các bé học mẫu giáo. Một sản phẩm có đặc điểm: là xì gà, thương hiệu - Habanos, quốc gia sản xuất - Cuba sẽ phù hợp với những người giới tính nam, có thu nhập cao và sành hút thuốc lá.

- **Collaborative filtering**: Hay còn gọi là lọc tương tác, sử dụng sự tương tác qua lại trong hành vi mua sắm giữa các khách hàng để tìm ra sở thích của một khách hàng đối với một sản phẩm. Hầu hết các hành vi hoặc sở thích của mọi người đều có những đặc điểm chung và có thể nhóm lại thành các nhóm tương đồng. Một phụ nữ A nếu đến siêu thị mua dầu ăn thường mua thêm nước tương và nước mắm. Hành vi này lặp lại đối với 100 lượt mua sắm là 90 lần thì khả năng cao một phụ nữ B nếu mua dầu ăn cũng sẽ mua thêm nước tương và nước mắm. Từ đó sẽ khuyến nghị sản phẩm cho khách hàng dựa trên hành vi của các khách hàng khác liên quan nhất.
- **Kết hợp cả 2 phương pháp**: Ngoài ra chúng ta cũng có thể sử dụng kết hợp cả 2 phương pháp trên để tạo thành một thuật toán kết hợp. Ưu điểm của phương pháp này đó là vừa tận dụng được các thông tin từ phía sản phẩm và các thông tin về hành vi mua sắm của người dùng.

II. Utility matrix (Ma trận tiện ích)

2.1. Ví dụ về Utility matrix

Như đã đề cập, có hai thực thể chính trong các Recommendation Systems là *users* và *items*. Mỗi user sẽ có *mức độ quan tâm* (*degree of preference*) tới từng item khác nhau. Mức độ quan tâm này, nếu đã biết trước, được gán cho một giá trị ứng với mỗi cặp *user-item*. Giả sử rằng mức độ quan tâm được đo bằng giá trị user rate cho item, ta tạm gọi giá trị này là *rating*. Tập hợp tất cả các *ratings*, bao gồm cả những giá trị chưa biết cần được dự đoán, tạo nên một ma trận gọi là ***utility matrix***. Xét ví dụ sau:

	A	B	C	D	E	F
Mưa nửa đêm	5	5	0	0	1	?
Cỏ 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	?	?	?

Hình 1: Ví dụ về ***utility matrix*** với hệ thống Gợi ý bài hát. Các bài hát được người dùng đánh giá theo mức độ từ 0 đến 5 sao. Các dấu '?' nền màu xám ứng với việc dữ liệu chưa tồn tại trong cơ sở dữ liệu. Recommendation Systems cần phải tự điền các giá trị này.

Trong ví dụ này, có 6 *users A, B, C, D, E, F* và 5 bài hát. Các ô màu xanh thể hiện việc một *user* đã đánh giá một bài hát với *ratings* từ 0 (không thích) đến 5 (rất thích). Các ô có dấu '?' màu xám tương ứng với các ô chưa có dữ liệu. Công việc của một Recommendation Systems là dự đoán giá trị tại các ô màu xám này, từ đó đưa ra gợi ý cho người dùng. Recommendation Systems, vì vậy, đôi khi cũng được coi là bài toán *Matrix Completion* (*Hoàn thiện ma trận*).

Trong ví dụ đơn giản này, dễ thấy có 2 thể loại nhạc khác nhau: 3 bài đầu là nhạc *Bolero* và 2 bài sau là nhạc *Thiếu nhi*. Từ dữ liệu này, ta cũng có thể đoán được rằng *A, B* thích thể loại *Bolero*; *C, D*, E, F thích thể loại *Thiếu nhi*. Từ đó, một hệ thống tốt nên gợi ý *Cỏ úa* cho *B*; *Vùng lá me bay* cho *A,*; *Em yêu trường em* cho *D, E, F*. Giả sử chỉ có hai thể loại nhạc này, khi có một bài hát mới, ta chỉ cần phân lớp nó vào thể loại nào, từ đó đưa ra gợi ý với từng người dùng.

Thông thường, có rất nhiều *users* và *items* trong hệ thống, và mỗi *user* thường chỉ *rate* một số lượng rất nhỏ các *item*, thậm chí có những *user* không rate *item* nào (với những *users* này thì cách tốt nhất là gợi ý các *items* phổ biến nhất). Vì vậy, lượng ô màu xám của utility matrix trong các bài toán đó thường là rất lớn, và lượng các ô đã được điền là một số rất nhỏ.

Rõ ràng rằng càng nhiều ô được điền thì độ chính xác của hệ thống sẽ càng được cải thiện. Vì vậy, các hệ thống luôn 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 các sản phẩm, vì thế, 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 đó có chính sách quảng cáo hợp lý.

2.2. Xây dựng Utility Matrix

Không có Utility matrix, gần như không thể gợi ý được sản phẩm tới người dùng, ngoài cách luôn luôn gợi ý các sản phẩm phổ biến nhất. Vì vậy, trong các Recommender Systems, việc xây dựng Utility Matrix là tối quan trọng. Tuy nhiên, việc xây dựng ma trận này thường có gặp nhiều khó khăn. Có hai hướng tiếp cận phổ biến để xác định giá trị *rating* cho mỗi cặp *user-item* trong Utility Matrix:

1. *Nhờ* người dùng *rate* sản phẩm. Amazon luôn *nhờ* người dùng *rate* các sản phẩm của họ bằng cách gửi các email nhắc nhở nhiều lần. Rất nhiều hệ thống khác cũng làm việc tương tự. Tuy nhiên, cách tiếp cận này có một vài hạn chế, vì thường thì người dùng ít khi *rate* sản phẩm. Và nếu có, đó có thể là những đánh giá thiên lệch bởi những người sẵn sàng *rate*.

2. Hướng tiếp cận thứ hai là dựa trên hành vi của *users*. Rõ ràng, nếu một người dùng mua một sản phẩm trên Amazon, xem một clip trên Youtube (có thể là nhiều lần), hay đọc một bài báo, thì có thể khẳng định rằng người dùng đó *thích* sản phẩm đó. Facebook cũng dựa trên việc bạn *like* những nội dung nào để hiển thị *newsfeed* của bạn những nội dung liên quan. Bạn càng đam mê Facebook, Facebook càng được hưởng lợi, thế nên nó luôn mang tới bạn những thông tin mà khả năng cao là bạn *muốn* đọc. (*Đừng đánh giá xã hội qua facebook*). Thường thì với cách này, ta chỉ xây dựng được một ma trận với các thành phần là 1 và 0, với 1 thể hiện người dùng thích sản phẩm, 0 thể hiện chưa có thông tin. Trong trường hợp này, 0 không có nghĩa là thấp hơn 1, nó chỉ có nghĩa là người dùng chưa cung cấp thông tin. Chúng ta cũng có thể xây dựng ma trận với các giá trị cao hơn 1 thông qua thời gian hoặc số lượt mà người dùng xem một sản phẩm nào đó. Đôi khi, nút *dislike* cũng mang lại những lợi ích nhất định cho hệ thống, lúc này có thể gán giá trị tương ứng bằng -1 chẳng hạn.

III. Content-Based Recommendations

3.1. Item profiles

Trong các hệ thống content-based, tức dựa trên *nội dung* của mỗi *item*, chúng ta cần xây dựng một bộ hộ sơ (profile) cho mỗi item. *Profile* này được biểu diễn dưới dạng toán học là một feature vector. Trong những trường hợp đơn giản, *feature vector* được trực tiếp trích xuất từ *item*. Ví dụ, xem xét các *features* của một bài hát mà có thể được sử dụng trong các Recommendation Systems:

1. *Ca sĩ*. Cùng là bài *Thành phố buồn* nhưng có người thích bản của Đan Nguyên, có người lại thích bản của Đàm Vĩnh Hưng.
2. *Nhạc sĩ sáng tác*. Cùng là nhạc trẻ nhưng có người thích Phan Mạnh Quỳnh, người khác lại thích MTP.
3. *Năm sáng tác*. Một số người thích nhạc xưa cũ hơn nhạc hiện đại.
4. *Thể loại*. Điều này thì chắc rồi, Quan họ và Bolero sẽ có thể thu hút những nhóm người khác nhau.

Có rất nhiều yếu tố khác của một bài hát có thể được sử dụng. Ngoại trừ *Thể loại* khó định nghĩa, các yếu tố khác đều được xác định rõ ràng.

Trong ví dụ ở Hình 1 phía trên, chúng ta đơn giản hoá bài toán bằng việc xây dựng một feature vector hai chiều cho mỗi bài hát: chiều thứ nhất là mức độ *Bolero*, chiều thứ hai là mức độ *Thiếu nhi* của bài đó. Đặt các feature vector cho mỗi bài hát là **x1, x2, x3, x4, x5**. Giả sử các feature vector (ở dạng hàng) cho mỗi bài hát được cho trong Hình 2 dưới đây:

	A	B	C	D	E	F	item's feature vectors
Mưa nửa đêm	5	5	0	0	1	?	$\mathbf{x}_1 = [0.99, 0.02]$
Cỏ úa	5	?	?	0	?	?	$\mathbf{x}_2 = [0.91, 0.11]$
Vùng lá me bay	?	4	1	?	?	1	$\mathbf{x}_3 = [0.95, 0.05]$
Con cò bé bé	1	1	4	4	4	?	$\mathbf{x}_4 = [0.01, 0.99]$
Em yêu trường em	1	0	5	?	?	?	$\mathbf{x}_5 = [0.03, 0.98]$
User's models	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	$\leftarrow$ **need to optimize**

Hình 2: Giả sử feature vector cho mỗi item được cho trong cột cuối cùng. Với mỗi user, chúng ta cần tìm một mô hình θi tương ứng sao cho mô hình thu được là tốt nhất.

3.2. Xây dựng hàm mất mát

Giả sử rằng số users là N, số items là M, utility matrix được mô tả bởi ma trận Y. Thành phần ở hàng thứ m, cột thứ n của Y là mức độ quan tâm (ở đây là số sao đã rate) của user thứ n lên sản phẩm thứ m mà hệ thống đã thu thập được. Ma trận Y bị khuyết rất nhiều thành phần tương ứng với các giá trị mà hệ thống cần dự đoán. Thêm nữa, gọi R là ma trận rated or not thể hiện việc một user đã rated một item hay chưa. Cụ thể, rij bằng 1 nếu item thứ i đã được rated bởi user thứ j, bằng 0 trong trường hợp ngược lại.

Mô hình tuyến tính:

Giả sử rằng ta có thể tìm được một mô hình cho mỗi user, minh hoạ bởi vector cột hệ số w_i và bias b_n sao cho mức độ quan tâm của một user tới một item có thể tính được bằng một hàm tuyến tính:

$$y_{mn} = \mathbf{x}_m \mathbf{w}_n + b_n \quad (1)$$

Xét một user thứ n bất kỳ, nếu ta coi training set là tập hợp các thành phần đã được điền của yn, ta có thể xây dựng hàm mất mát tương tự như Ridge Regression như sau:

$$\mathcal{L}_n = \frac{1}{2} \sum_{m\,:\,r_{mn}=1} (\mathbf{x}_m \mathbf{w}_n + b_n - y_{mn})^2 + \frac{\lambda}{2} ||\mathbf{w}_n||_2^2$$

Trong đó, thành phần thứ hai là regularization term và λ là một tham số dương. Chú ý rằng regularization thường không được áp dụng lên bias bn. Trong thực hành, trung bình cộng của lỗi thường được dùng, và mất mát L_n được viết lại thành:

$$\mathcal{L}_n = \frac{1}{2s_n} \sum_{m\,:\,r_{mn}=1} (\mathbf{x}_m \mathbf{w}_n + b_n - y_{mn})^2 + \frac{\lambda}{2s_n} ||\mathbf{w}_n||_2^2$$

Trong đó s_n là số lượng các items mà user thứ n đã rated. Nói cách khác:

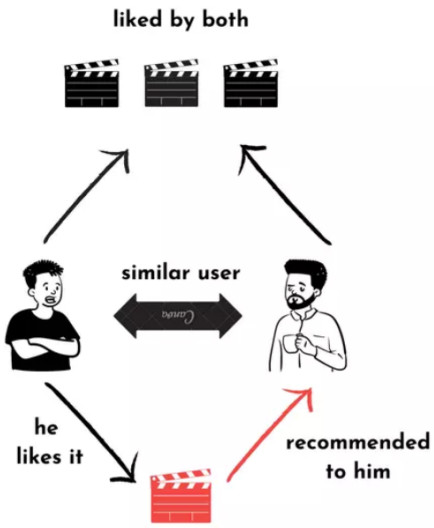
$$s_n = \sum_{m=1}^{M} r_{mn},$$

là tổng các phần tử trên cột thứ **n** của ma trận *rated or not **R***

IV. Collaborative Filtering Recommender (CFR)

Collaborative Filtering Recommender (CFR) là một cách tiếp cận (mô hình) dựa trên dữ liệu lịch sử về hành vi mua sắm của khách hàng với ý tưởng rằng nếu, ví dụ, hai khách hàng cùng mua một cuốn sách (hoặc xem cùng một bộ phim) thì rất có thể họ sẽ tái diễn pattern đó trong tương lai. Hoặc hai khách hàng A và B được cho là có hành vi mua sắm tương tự nhau thì nếu khách hàng A đã xem một bộ phim X nào đó nhưng B thì lại chưa xem bộ phim này thì nếu hệ thống gợi ý rằng nên giới thiệu phim X này cho khách hàng B. Cách tiếp cận này rõ ràng chỉ dựa trên sở thích của khách hàng (user preferences) chứ

không dựa trên những đặc điểm hay nội dung (contents) của sách/phim để đưa ra khuyến nghị cho người dùng/khách hàng tiềm năng.



Có 2 hướng tiếp cận Collaborative Filtering:

- Một là xác định mức độ quan tâm của mỗi user tới một item dựa trên mức độ quan tâm của users gần giống nhau (similar users) tới item đó còn được gọi là User-user collaborative filtering.
- Hai là thay vì xác định user similarities, 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.

Thuật toán sẽ không cần sử dụng thông tin sản phẩm là đầu vào cho dự báo rating. Đầu vào của thuật toán là một ma trận tiện ích (utility matrix) chứa giá trị rating của các cặp (user, item). Mỗi cột là các rating mà một user đã rate và mỗi dòng là các rating của một item được rate. Có 2 phương pháp chính được sử dụng trong collaborative filtering bao gồm: Neighborhood-based Collaborative Filtering và Matrix Factorization.

4.1. Neighborhood-based Collaborative Filtering (NBCF)

Ý tưởng cơ bản của NBCF là xác định mức độ quan tâm của một user tới một item dựa trên các users khác gần giống với user này. Việc gần giống nhau giữa các users có thể được xác định thông qua mức độ quan tâm của các users này tới các items khác mà hệ thống đã biết. Ví dụ, A, B đều thích phim Cảnh sát hình sự, tức đều rate bộ phim này 5

sao. Ta đã biết A cũng thích Người phán xử, vậy nhiều khả năng B cũng thích bộ phim này.

Các bạn có thể đã hình dung ra, hai câu hỏi quan trọng nhất trong một hệ thống Neighborhood-based Collaborative Filtering là:

- Làm thế nào xác định được sự giống nhau giữa hai users?
- Khi đã xác định được các users gần giống nhau (similar users) rồi, làm thế nào dự đoán được mức độ quan tâm của một user lên một item?

Việc xác định mức độ quan tâm của mỗi user tới một item dựa trên mức độ quan tâm của similar users tới item đó còn được gọi là User-user collaborative filtering. Có một hướng tiếp cận khác được cho là làm việc hiệu quả hơn là Item-item collaborative filtering. Trong hướng tiếp cận này, thay vì xác định user similarities, 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.

4.1.1. User-user Collaborative Filtering

Similarity functions

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. Dữ liệu duy nhất chúng ta có là Utility matrix Y, vậy nên sự giống nhau này phải được xác định dựa trên các cột tương ứng với hai users trong ma trận này.

	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

Ví dụ về utility matrix dựa trên số sao một user rate cho một item. Một cách trực quan, hành vi của **u0** giống với **u1** hơn là **u2, u3, u4, u5, u6**. Từ đó có thể dự đoán rằng u0 sẽ quan tâm tới i2 vì u1 cũng quan tâm tới item này.

Giả sử có các users từ u0 đến u6 và các items từ i0 đến i4 trong đó các số trong mỗi ô vuông thể hiện số sao mà mỗi user đã rated cho item với giá trị cao hơn thể hiện mức độ quan tâm cao hơn. Các dấu hỏi chấm là các giá trị mà hệ thống cần phải đi tìm. Đặt mức độ giống nhau của hai users ui , uj là sim (u_i , u_j).

Quan sát đầu tiên chúng ta có thể nhận thấy là các u0, u1 thích i0 , i1 , i2 và không thích i3 , i4 cho lắm. Điều ngược lại xảy ra ở các users còn lại. Vì vậy, một similarity function tốt cần đảm bảo:

$$\text{sim}(u_0, u_1) > \text{sim}(u_0, u_i),\ \forall i > 1.$$

Từ đó, để xác định mức độ quan tâm của u0 lên i2 , chúng ta nên dựa trên hành vi của u1 lên sản phẩm này. Rất may rằng u1 đã thích i2 nên hệ thống cần recommend i2 cho u0.

Câu hỏi đặt ra là: hàm số *similarity* nào là tốt? Để đo *similarity* giữa hai *users*, **cách thường làm là xây dựng *feature vector* cho mỗi *user* rồi áp dụng một hàm có khả năng đo *similarity* giữa hai *vectors* đó**. Chú ý rằng việc xây dựng *feature vector* này khác với việc xây dựng item profiles như trong Content-based Recommendation Systems. Các vectors này được xây dựng trực tiếp dựa trên Utility matrix chứ không dùng dữ liệu ngoài như item profiles. Với mỗi user, thông tin duy nhất chúng ta biết là các ratings mà *user* đó đã thực hiện, tức cột tương ứng với user đó trong Utility matrix. Tuy nhiên, khó khăn là các cột này thường có rất nhiều *missing 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.

Vậy mỗi dấu '?' nên được thay bởi giá trị nào để hạn chế việc sai lệch quá nhiều? Một lựa chọn bạn có thể nghĩ tới là thay các dấu '?' bằng giá trị '0'. Điều này không thực sự tốt vì giá trị '0' tương ứng với mức độ quan tâm thấp nhất. Một giá trị an toàn hơn là 2.5 vì nó là trung bình cộng của 0, mức thấp nhất, và 5, mức cao nhất. Tuy nhiên, giá trị này có hạn chế đối với những users dễ tính hoặc khó tính. Với các users dễ tính, thích tương ứng với 5 sao, không thích có thể ít sao hơn 1 chút, 3 sao chẳng hạn. Việc chọn giá trị 2.5 sẽ khiến cho các items còn lại là quá negative đối với user đó. Điều ngược lại xảy ra với những user khó tính hơn khi chỉ cho 3 sao cho các items họ thích và ít sao hơn cho những items họ không thích.

Một giá trị khả dĩ hơn cho việc này là trung bình cộng của các ratings mà user tương ứng đã thực hiện. Việc này sẽ tránh được việc users quá khó tính hoặc dễ tính, tức lúc nào cũng có những items mà một user thích hơn so với những items khác.

	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

a) Original utility matrix $\mathbf{Y}$ and mean user ratings.

	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

b) Normalized utility matrix $\bar{\mathbf{Y}}$.

	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

c) User similarity matrix $\mathbf{S}$.

	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

d) $\hat{\mathbf{Y}}$

Predict normalized rating of u_1 on i_1 with $k = 2$

Users who rated i_1 : $\{u_0, u_3, u_5\}$

Corresponding similarities: {0.83, -0.40, -0.23}

$\Rightarrow$ most similar users: $\mathcal{N}(u_1, i_1) = \{u_0, u_5\}$

with **normalized ratings** {**0.75, 0.5**}

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

e) Example

	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

f) Full $\mathbf{Y}$

Ví dụ mô tả User-user Collaborative Filtering. ***a)*** *Utility Matrix ban đầu.* ***b)*** *Utility Matrix đã được chuẩn hoá.* ***c)*** *User similarity matrix.* ***d)*** *Dự đoán các (normalized) ratings còn thiếu.* ***e)*** *Ví dụ về cách dự đoán normalized rating của u1 cho i1 .* ***f)*** *Dự đoán các (denormalized) ratings còn thiếu.*

Cosine Similarity:

Đây là hàm được sử dụng nhiều nhất, và cũng quen thuộc với các bạn nhất. Nếu các bạn không nhớ công thức tính cos của góc giữa hai vector u1 , u2 trong chương trình phổ thông, thì dưới đây là công thức:

$$\text{cosine\_similarity}(\mathbf{u}_1, \mathbf{u}_2) = \cos(\mathbf{u}_1, \mathbf{u}_2) = \frac{\mathbf{u}_1^T \mathbf{u}_2}{||\mathbf{u}_1||_2 . ||\mathbf{u}_2||_2} \quad (1)$$

Trong đó u1,2 là vectors tương ứng với users1, 2 đã được chuẩn hoá như ở trên. Có một tin vui là python có hàm hỗ trợ tính toán hàm số này một cách hiệu quả. Độ similarity của

hai vector là 1 số trong đoạn [-1, 1]. Giá trị bằng 1 thể hiện hai vector hoàn toàn similar 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 cos bằng -1 thể hiện hai vector này hoàn toàn trái ngược nhau. Điều này cũng hợp lý , tức khi hành vi của hai users là hoàn toàn ngược nhau thi similarity giữa hai vector đó là thấp nhất.

4.1.2. Item-item Collaborative Filtering

Một số hạn chế của User-user CF:

- Trên thực tế, số lượng users luôn lớn hơn số lượng items rất nhiều. Kéo theo đó là Similarity matrix là rất lớn với số phần tử phải lưu giữ là hơn 1 nửa của bình phương số lượng users (chú ý rằng ma trận này là đối xứng). Việc này, như đã đề cập ở trên, khiến cho việc lưu trữ ma trận này trong nhiều trường hợp là không khả thi.
- Ma trận Utility Y thường là rất sparse. Với số lượng users rất lớn so với số lượng items, rất nhiều cột của ma trận này sẽ rất sparse, tức chỉ có một vài phần tử khác 0. Lý do là users thường lười rating. Cũng chính vì việc này, một khi user đó thay đổi rating hoặc rate thêm items, trung bình cộng các ratings cũng như vector chuẩn hóa tương ứng với user này thay đổi nhiều. Kéo theo đó, việc tính toán ma trận Similarity, vốn tốn nhiều bộ nhớ và thời gian, cũng cần được thực hiện lại.

Ngược lại, nếu chúng ta tính toán similarity giữa các items rồi recommend những items gần giống với item yêu thích của một user thì sẽ có những lợi ích sau:

- Vì số lượng items thường nhỏ hơn số lượng users, Similarity matrix trong trường hợp này cũng nhỏ hơn nhiều, thuận lợi cho việc lưu trữ và tính toán ở các bước sau.
- Vì số lượng phần tử đã biết trong Utility matrix là như nhau nhưng số hàng (items) ít hơn số cột (users), nên trung bình, mỗi hàng của ma trận này sẽ có nhiều phần tử đã biết hơn số phần tử đã biết trong mỗi cột. Việc này cũng dễ hiểu vì mỗi item có thể được rated bởi nhiều users. Kéo theo đó, giá trị trung bình của mỗi hàng ít bị thay đổi hơn khi có thêm một vài ratings. Như vậy, việc cập nhật ma trận Similarity Matrix có thể được thực hiện ít thường xuyên hơn.

Cách tiếp cận thứ hai này được gọi là Item-item Collaborative Filtering. Hướng tiếp cận này được sử dụng nhiều trong thực tế hơn.

Quy trình dự đoán missing ratings cũng tương tự như trong User-user CF. Hình 3 mô tả quy trình này với ví dụ nêu ở phần trên.

	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

a) Original utility matrix $\mathbf{Y}$ and mean item ratings.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	2.4	2.4	-.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

b) Normalized utility matrix $\bar{\mathbf{Y}}$.

	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

c) Item similarity matrix $\mathbf{S}$.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	2.4	2.4	-.6	-2.6	-1.6	-0.29	-1.52
i_1	2	2.4	-0.6	-2	-1.25	0	-2.25
i_2	2.4	2.25	-0.75	-2.6	-1.20	-0.75	-0.75
i_3	-1.17	-1.17	-0.17	0.83	0.83	0.34	0.83
i_4	-0.75	-2.75	1.25	1.03	1.16	0.65	2.25

d) Normalized utility matrix $\bar{\mathbf{Y}}$.

Ví dụ mô tả Item-Item Collaborative Filtering. a) Utility Matrix ban đầu. b) Utility Matrix đã được chuẩn hoá. c) User similarity matrix. d) Dự đoán các (normalized) ratings còn thiếu.

4.2. Matrix Factorization Collaborative Filtering

Trong **Content-based Recommendation Systems**, mỗi item được mô tả bằng một vector x được gọi là item profile. Trong phương pháp này, ta cần tìm một vector hệ số w tương ứng với mỗi user sao cho rating đã biết mà user đó cho item xấp xỉ với:

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

Với cách làm trên, Utility Matrix Y , giả sử đã được điền hết, sẽ xấp xỉ với:

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

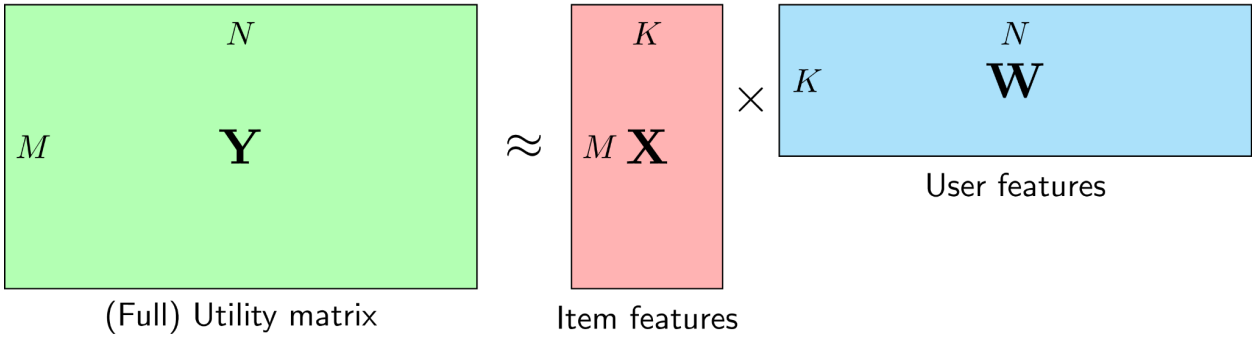
với M , N lần lượt là số items và số users.

Chú ý rằng, 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 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 X và 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.

Thông thường, K được chọn là một 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 . Chính vì vậy, phương pháp này còn được gọi là Low-Rank Matrix Factorization.

$$\mathbf{Y} \approx \hat{\mathbf{Y}} = \mathbf{XW}$$



Có một vài điểm lưu ý ở đây:

- Ý tưởng chính đằng sau Matrix Factorization cho Recommendation Systems là tồn tại các latent features (tính chất ẩn) 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 x w 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.
- Vậy tại sao Matrix Factorization lại được xếp vào Collaborative Filtering? Câu trả lời đến từ việc đi tối ưu hàm mất mát mà chúng ta sẽ thảo luận ở Mục 2. Về cơ bản, để tìm nghiệm của bài toán tối ưu, ta phải lần lượt đi tìm X và W khi thành phần còn lại được cố định. Như vậy, mỗi hàng của X sẽ phụ thuộc vào toàn bộ các cột của W . Ngược lại, mỗi cột của W lại phục thuộc vào toàn bộ các hàng của X . Như vậy, có những mối quan hệ ràng buộc chằng chịt giữa các thành phần của hai ma trận trên. Tức chúng ta cần sử dụng thông tin của tất cả để suy ra tất cả. Vậy nên phương pháp này cũng được xếp vào Collaborative Filtering.
- 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.
- Thêm nữa, việc lưu trữ hai ma trận $\boldsymbol{X}$ và $\boldsymbol{W}$ yêu cầu lượng bộ nhớ nhỏ khi so với việc lưu toàn bộ Similarity matrix trong Neighborhood-based Collaborative

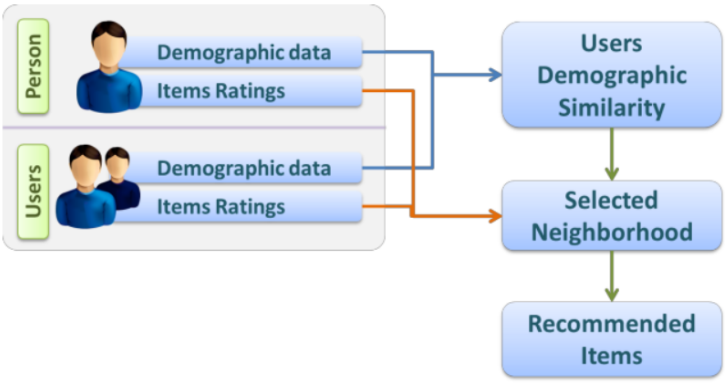
Filtering. Cụ thể, ta cần bộ nhớ để chứa K(M + N) phần tử thay vì lưu M^2 hoặc N^2 của Similarity matrix.

4.3. Hạn chế của phương pháp collaborative filtering

- Thường phải lưu một ma trận hệ số tương quan với kích thước rất lớn. Việc này dẫn tới tốn tài nguyên lưu trữ và thời gian tính toán.
- Ở trên ta đã lựa chọn việc chuẩn hóa theo chiều user. Ngoài ra, ta cũng có thể lựa chọn chuẩn hóa theo chiều item mà không làm thay đổi phương pháp bằng cách chuyển vị ma trận tiện ích Y. Việc lựa chọn chuẩn hóa theo chiều nào sẽ căn cứ trên kích thước theo chiều nào là lớn hơn. Thông thường số lượng user sẽ nhiều hơn item. Khi đó chuẩn hóa theo item sẽ có lợi thế hơn bởi: Kích thước ma trận hệ số tương quan giữa các user là nhỏ hơn nên tốn ít tài nguyên. Thêm nữa khi một user rating một item mới thì giá trị thay đổi về trung bình trên mỗi cột item là nhỏ hơn so với trường hợp chuẩn hóa theo user. Điều này dẫn tới ma trận hệ số tương quan ít thay đổi hơn và tần suất cập nhật cũng ít hơn.

V. Demographic-based Recommender

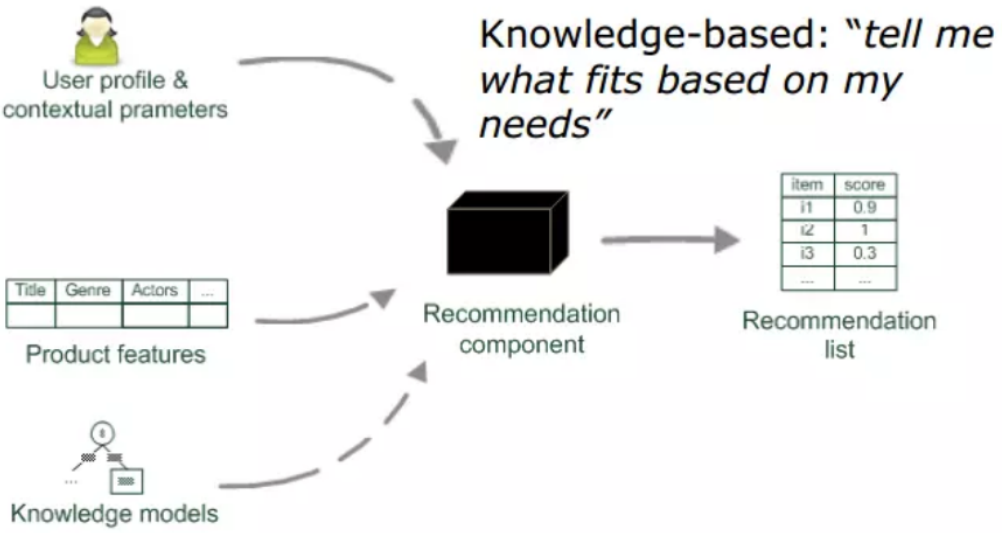
Đúng như tên gọi, kiểu hệ thống này sẽ gợi ý item dựa nhân khẩu học của user. Giả thuyết cho rằng những thị trường khác nhau nên được gợi ý những item khác nhau. Chẳng hạn, user sẽ được điều hướng tới những website khác nhau dựa trên ngôn ngữ và địa lý. Hoặc là việc gợi ý có thể thay đổi dựa trên tuổi của user. Mặc dù cách tiếp cận này khá phổ biến trong mảng tiếp thị truyền thống, nhưng *có rất ít nghiên cứu khoa học về kỹ thuật này.*



Nhiều ngành đã áp dụng cách tiếp cận này vì nó không quá phức tạp và dễ thực hiện. Trong hệ thống đề xuất dựa trên nhân khẩu học, trước tiên các thuật toán cần nghiên cứu thị trường phù hợp trong khu vực được chỉ định kèm theo một cuộc khảo sát ngắn để thu thập dữ liệu để phân loại. Các kỹ thuật nhân khẩu học hình thành mối tương quan "người với người" giống như Collaborative Filtering Recommender, nhưng sử dụng dữ liệu khác. Lợi ích của cách tiếp cận Demographic-based là nó không yêu cầu lịch sử ratings của người dùng như trong các hệ thống đề xuất Collaborative và Content-based.

VI. Knowledge-based Recommender

Khác với Collaborative Recommender, Knowledge-based Recommender không yêu cầu thông tin về ratings các sản phẩm của khách hàng. Hệ thống Knowledge-based sẽ gợi ý item dựa trên miền kiến thức cụ thể rằng những đặc điểm nào của item đáp ứng được nhu cầu và thị hiếu của user, cuối cùng là item đó có hữu ích cho user hay không. Hệ thống này có khuynh hướng hoạt động tốt hơn các hệ thống khác khi bắt đầu triển khai, nhưng nếu không được trang bị các thành phần hỗ trợ học thì nó có thể bị vượt qua bởi các phương pháp khác mà có thể khai thác lịch sử tương tác giữa user và item (chẳng hạn như CFR).



Hệ thống có thể thông báo người dùng đưa ra một loạt quy tắc hoặc nguyên tắc về kết quả sẽ như thế nào hoặc ví dụ về một mục. Sau đó, hệ thống sẽ tìm kiếm thông qua cơ sở dữ liệu các mục của nó và trả về kết quả tương tự.
Một ví dụ điển hình là các trang web tìm nhà, xe hơi,... . Hệ thống sẽ yêu cầu người dùng input các giá trị như giá nhà, vị trí địa lý, tiện ích xung quanh… để tìm trong cơ sở dữ liệu về các item phù hợp và đưa ra kết quả truy xuất được.

Như vậy, vì sao phải sử dụng Knowledge-based Recommender thay vì chỉ sử dụng tìm kiếm (searching) thông thường?
Câu trả lời là Đúng, có thể chỉ sử dụng tìm kiếm thông thường, nhưng cách tiếp cận đó sẽ thiếu nhiều chức năng hữu ích khác của Knowledge-based Recommender. Các item cần phải tìm kiếm có thể sẽ rất phức tạp, một vài tổ hợp tìm kiếm sẽ không cho ra bất kỳ sản phẩm trùng khớp nào cả. Knowledge-based Recommender còn cho bạn cá nhân hóa (personalize) kết quả cho từng khách hàng.

References

1. Bài 23: Content-based Recommendation Systems | Machine Learning cơ bản (machinelearningcoban.com)
2. Bài 24: Neighborhood-Based Collaborative Filtering | Machine Learning cơ bản (machinelearningcoban.com)
3. Bài 25: Matrix Factorization Collaborative Filtering | Machine Learning cơ bản (machinelearningcoban.com)
4. Bài 15 - collaborative và content-based filtering | Khoa học dữ liệu (phamdinhkhanh.github.io)s