TRƯỜNG ĐẠI HỌC MỎ ĐỊA CHẤT
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN HỆ THỐNG THÔNG TIN VÀ TRI THỨC

BÁO CÁO SEMINAR
TÌM HIỂU VỀ HỆ THỐNG KHUYẾN NGHỊ

Người trình bày: ThS. Bùi Thị Vân Anh

Hà Nội 11/2023

1. MỞ ĐẦU

Hệ thống khuyến nghị hay còn gọi là hệ thống gợi ý (Recommender Systems – RS) được ứng dụng rất thành công trong dự đoán sở thích/thói quen của người dùng dựa vào sở thích/thói quen của họ trong quá khứ. RS đang được ứng dụng trong rất nhiều lĩnh vực khác nhau như:

- ✓ thương mại điện tử (hỗ trợ bán hàng trực tuyến),
- ✓ giải trí (gợi ý phim ảnh, bài hát,..),
- ✓ giáo dục đào tạo (gợi ý nguồn tài nguyên học tập, nghiên cứu,..).

Ví dụ:

- ✓ 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.

RS mở ra nhiều tiềm năng trong nghiên cứu cũng như trong xây dựng các hệ thống thực tế, đặc biệt là các hệ hỗ trợ người dùng ra quyết định.

2. CÁC KHÁI NIỆM CHÍNH

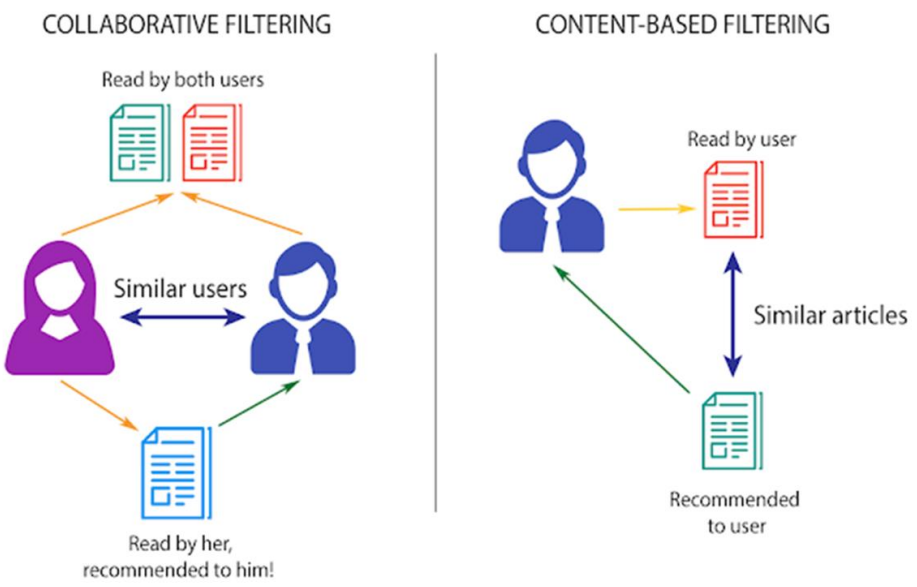
2.1. Giới thiệu

Trong RS, thông thường người ta quan tâm đến ba thông tin chính là

- ✓ **người dùng** (user)
- ✓ **mục tin** (items): item có thể là sản phẩm, bộ phim, bài hát, bài báo,.. tùy hệ thống)
- ✓ **phản hồi** (feedback) của người dùng trên mục tin đó (thường là các **xếp hạng**/đánh giá – rating biểu diễn mức độ thích/quan tâm của họ).

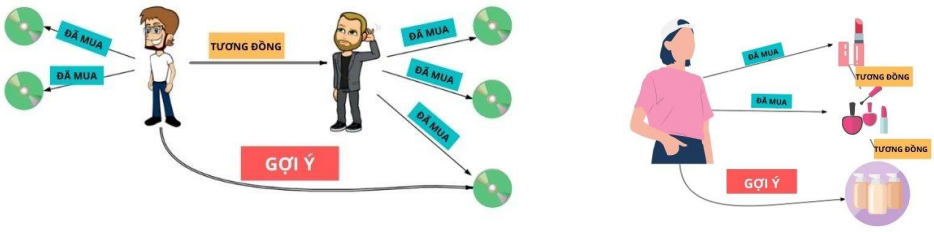
Các Recommendation Systems thường được chia thành hai nhóm lớn:

- ✓ *Content-based systems:* Phương pháp gợi ý dựa theo nội dung
- ✓ *Collaborative filtering:* lọc cộng tác
- ✓ *Hybrid Recommenders*



- *Content-based systems:* Phương pháp gợi ý dựa theo nội dung

- *Collaborative filtering:* lọc cộng tác (Dựa trên sự tương đồng). Có hai loại phương pháp tiếp cận dựa trên sự tương đồng là
  - ✓ Người dùng - Người dùng tương tự (User-User)
  - ✓ Mặt hàng - Mặt hàng tương tự (Item - Item)



2.2. Utility matrix

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*.

Ví dụ về utility matrix với hệ thống Gợi ý bài hát.

	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	?	?	?

- ✓ 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 trên, 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*.

	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	?	?	?

Xây dựng Utility Matrix

- ✓ Nhờ người dùng rate sản phẩm
- ✓ Dựa trên hành vi của users.

3. CONTENT-BASED RECOMMENDATION

Trong các hệ thống content-based, tức dựa trên *nội dung* của mỗi *item*, chúng ta cần xây dựng một bộ hồ 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:

- ✓ *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.
- ✓ *Nhạc sĩ sáng tác*. Cùng là nhạc trẻ nhưng có người thích Phan Mạnh Quỳnh, người khác lại thích MTP.
- ✓ *Năm sáng tác*. Một số người thích nhạc xưa cũ hơn nhạc hiện đại.
- ✓ *Thể loại*. Điều này thì chắc rồi, Quan họ và Bolero sẽ có thể thu hút những nhóm người khác nhau.

Ví dụ 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 $\mathbf{x_1}$,$\mathbf{x_2}$,$\mathbf{x_3}$,$\mathbf{x_4}$,$\mathbf{x_5}$

	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	← **need to optimize**

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*

Bài toán đi tìm mô hình θ_i cho mỗi *user* có thể được coi là một bài toán Regression trong trường hợp *ratings* là một dải giá trị, hoặc bài toán *Classification* trong trường hợp *ratings* là một vài trường hợp cụ thể, như *like/dislike* chẳng hạn.

Dữ liệu training để xây dựng mỗi mô hình θ_i là các cặp (*item profile, ratings*) tương ứng với các *items* mà *user* đó đã *rated*. Việc *điền* các giá trị còn thiếu trong ma trận Utility

chính là việc dự đoán đầu ra cho các *unrated items* khi áp dụng mô hình θ_i lên chúng.

Việc lựa chọn mô hình Regression/Classification nào tuỳ thuộc vào ứng dụng.

Giả sử rằng số *users* là N, số *items* là M, *utility maxtrix* 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. Gọi R là ma trận *rated or not* thể hiện việc một *user* đã *rated* một *item* hay chưa. Cụ thể, r_{ij} bằng 1 nếu *item* thứ i đã được *rated* bởi *user* thứ j, bằng 0 trong trường hợp ngược lại. Giả sử rằng ta có thể tìm được một mô hình có thể tính được mức độ quan tâm của mỗi user với mỗi item bằng một hàm tuyến tính:

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

Trong đó, $\mathbf{x_m}$ là vector đặc trưng của item m. Mục tiêu của chúng ta sẽ là học ra mô hình của user, tức là tìm ra w_n và b_n.

Xét một user thứ n bất kỳ, nếu ta coi training set là tập hợp các thành phần đã được điền của y_n, ta có thể xây dựng hàm mất mát tương tự 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 $\mathbf{b_n}$. Trong thực hành, trung bình cộng của lỗi thường được dùng, và mất mát nên L_n được viết lại thành:

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

Trong đó sn là số lượng các items mà user thứ n đã rated. Nói cách khác, sn là tổng các phần tử trên cột thứ n của ma trận rated or not R:

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

Vì biểu thức loss function chỉ phụ thuộc vào các items đã được rated, ta có thể rút gọn nó bằng cách đặt là sub vector của y được xây dựng bằng cách trích các thành phần khác dấu? ở cột thứ n, tức đã được rated bởi user thứ n trong Utility Matrix Y. Đặt $\widehat{X_n}$ là sub matrix của ma trận feature X, được tạo bằng cách trích các hàng tương ứng với các items đã được rated bởi user thứ n. Khi đó, biểu thức hàm mất mát của mô hình cho user thứ n được viết gọn thành:

$$\mathcal{L}_n = \frac{1}{2s_n}||\hat{\mathbf{X}}_n\mathbf{w}_n + b_n\mathbf{e}_n - \hat{\mathbf{y}}_n||_2^2 + \frac{\lambda}{2s_n}||\mathbf{w}_n||_2^2$$

Đây chính xác là hàm mất mát của Ridge Regression. Cặp nghiệm wn, bn có thể được tìm qua Stochastic Gradient Descent (SGD), hoặc Mini-batch GD

Ví dụ về hàm mất mát cho user E

	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	← **need to optimize**

feature matrix cho các items (mỗi hàng tương ứng với một item) là:

X=[0.99 0.02 0.91 0.11 0.95 0.05 0.01 0.99 0.03 0.98]

Xét trường hợp của user E với n=5, y5=[1, ? , ?, 4, ?]T ⇒ r5=[1, 0, 0, 1, 0]T. Vì E mới chỉ rated cho items thứ nhất và thứ tư nên s5=2. Hơn nữa:

$$\hat{\mathbf{X}}_5 = [0.99 \quad 0.02\,0.01 \quad 0.99], \hat{\mathbf{y}}_5 = [1\,4], \mathbf{e}_5 = [1\,1]$$

Khi đó, hàm mất mát cho hệ số tương ứng với user E là:

$$\mathcal{L}_5 = \frac{1}{4}||[0.99 \quad 0.02\,0.01 \quad 0.99]\,\mathbf{w}_5 + b_5[1\,1] - [1\,4]||_2^2 + \frac{\lambda}{4}||\mathbf{w}_5||_2^2$$

4. COLLABORATIVE FILTERING

Đặc điểm của Content-based Recommendation Systems là việc xây dựng mô hình cho mỗi user không phụ thuộc vào các users khác mà phụ thuộc vào profile của mỗi items. Việc làm này có lợi thế là tiết kiệm bộ nhớ và thời gian tính toán. Hệ thống có khả năng tận dụng các thông tin đặc trưng của mỗi item như được mô tả trong bản mô tả (description) của mỗi item.

Cách làm trên có hai nhược điểm cơ bản.

- ✓ Thứ nhất, khi xây dựng mô hình cho một user, các hệ thống Content-based không tận dụng được thông tin từ các users khác.
- ✓ Thứ hai, không phải lúc nào chúng ta cũng có bản mô tả cho mỗi item.

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

Có một số phương pháp CF có tên là:

- ✓ Neighborhood-based Collaborative Filtering (NBCF)
- ✓ Matrix Factorization Collaborative Filtering.

4.1. Neighborhood-based Collaborative Filtering

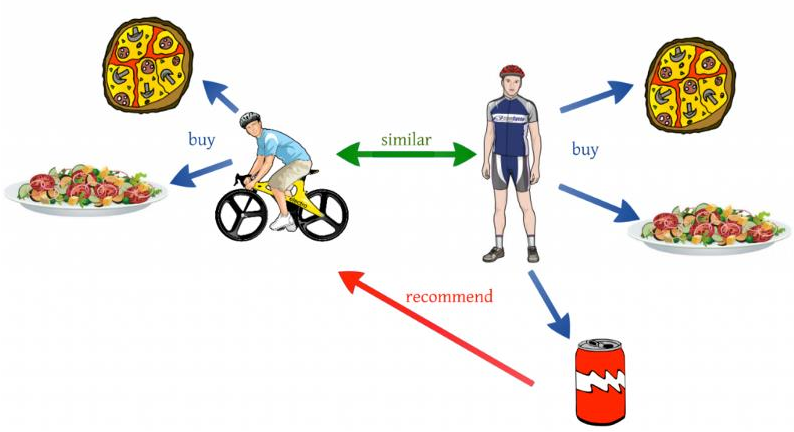
Ý tưởng cơ bản của NBCF là xác định mức độ quan tâm của một user tới một item dựa trên các users khác gần giống với user này. Việc gần giống nhau giữa các users có thể được xác định thông qua mức độ quan tâm của các users này tới các items khác mà hệ thống đã biết. Ví dụ, A, B đều thích phim Cảnh sát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

Hai câu hỏi quan trọng nhất trong một hệ thống Neighborhood-based Collaborative Filtering là:

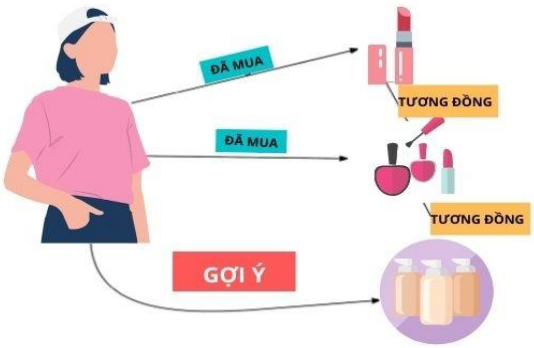
- ✓ Làm thế nào xác định được sự giống nhau giữa hai users?
- ✓ Khi đã xác định được các users gần giống nhau (similar users) rồi, làm thế nào dự đoán được mức độ quan tâm của một user lên một item?

Có hai hướng tiếp cận:

***User-user collaborative filtering*:** Xác định mức độ quan tâm của mỗi user tới một item dựa trên mức độ quan tâm của similar users tới item đó



Item-item collaborative filtering: 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.



4.2. Similarity functions

Xác định được sự giống nhau (similarity) giữa hai users. Sự giống nhau được xác định dựa trên các cột tương ứng với hai users trong ma trận Utility matrix

	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 u_0 giống với u_1 hơn là u_2, u_3, u_4, u_5, u_6. Từ đó có thể dự đoán rằng u_0 sẽ quan tâm tới i_2 vì u_1 cũng quan tâm tới item này. Đặt mức độ giống nhau của hai users u_i, u_j là sim(u_i, u_j). Quan sát đầu tiên chúng ta có thể nhận thấy là các u_0,u_1 thích i_0, i_1, i_2 và không thích i_3, i_4 cho lắm. Điều ngược lại xảy ra ở các users còn lại. Vì vậy, một similiarity function tốt cần đảm bảo:

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

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. Các feature vector được xây dựng trực tiếp dựa trên Utility matrix chứ không dùng dữ liệu ngoài như item profiles. Các cột trong Utility matrix thường có rất nhiều mising ratings vì mỗi user thường chỉ rated một số lượng rất nhỏ các items.

Cách khắc phục là bằng cách nào đó, ta giúp hệ thống điền các giá trị này sao cho việc điền không làm ảnh hưởng nhiều tới sự giống nhau giữa hai vector. Vậy mỗi dấu '?' nên được thay bởi giá trị nào để hạn chế việc sai lệch quá nhiều?

- ✓ thay các dấu '?' bằng giá trị 0. Không thực sự tốt vì giá trị '0' tương ứng với mức độ quan tâm thấp nhất.
- ✓ thay các dấu '?' bằng giá trị 2.5. Một giá trị an toàn vì nó là trung bình cộng của mức thấp nhất, và mức cao nhất. giá trị này có hạn chế đối với những users dễ tính hoặc khó tính
- ✓ Thay bằng trung bình cộng của các ratings mà user tương ứng đã thực hiện

	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
	↓	↓	↓	↓	↓	↓	↓
	3.25	2.75	2.5	1.33	2.5	1.5	3.33

Hàng cuối cùng trong là giá trị trung bình của ratings cho mỗi user.

	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

Utility Matrix đã được chuẩn hoá

Trừ từ mỗi rating đi giá trị trung bình này và thay các giá trị chưa biết bằng 0, ta sẽ được normalized utility matrix

Tại sao phải chuẩn hoá: Việc trừ đi trung bình cộng của mỗi cột khiến trong trong mỗi cột có những giá trị dương và âm.

- ✓ giá trị dương tương ứng với việc user thích item,
- ✓ giá trị âm tương ứng với việc user không thích item.
- ✓ giá trị 0 chưa xác định được liệu user có thích item hay không.

Về mặt kỹ thuật, số chiều của utility matrix là rất lớn với hàng triệu users và items, => chúng ta lưu ma trận này dưới dạng sparse matrix, tức chỉ lưu các giá trị khác không và vị trí của chúng. Việc này không những tối ưu bộ nhớ mà việc tính toán similarity matrix sau này cũng hiệu quả hơn.

Sau khi đã chuẩn hoá dữ liệu như trên, một vài similiraty function thường được sử dụng là:

Euclidean Distance

$$d(x,y) = \sqrt{\sum_{i}^{n}(x_i - y_i)^2}$$

Cosine Similarity

$$sim(a,b) = cos\theta = \frac{\vec{a}.\vec{b}}{\| \vec{a} \| \| \vec{b} \|}$$

Person corelation

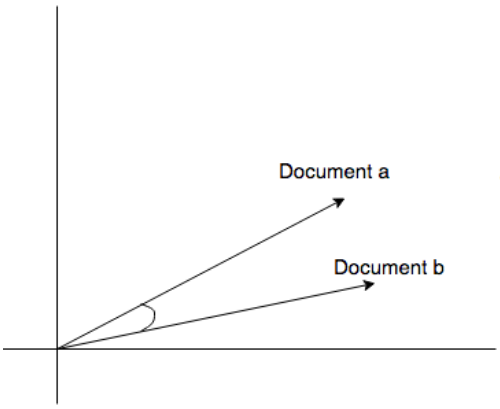
$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}}$$

Cosine Similarity

Công thức:

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

Trong đó u1, u2 là vectors tương ứng với users 1, 2 đã được chuẩn hoá. 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. Khi hành vi của hai users là hoàn toàn ngược nhau thi similarity giữa hai vector đó là thấp nhất.



$$sim(a,b) = cos\theta = \frac{\vec{a}.\vec{b}}{\| \vec{a} \| \| \vec{b} \|}$$

Similarity matrix là một ma trận đối xứng vì cos là một hàm chẵn, và nếu user A giống user B thì điều ngược lại cũng đúng. Các ô màu xanh trên đường chéo đều bằng 1 vì đó là cos của góc giữa 1 vector và chính nó, tức cos(0)=1. Khi tính toán ở các bước sau, chúng ta không cần quan tâm tới các giá trị 1 này.

	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

User similarity matrix

Rating prediction

Tương tự như KNN, trong Collaborative Filtering, missing rating cũng được xác định dựa trên thông tin về k neighbor users (chỉ quan tâm tới các users đã rated item đang xét). Predicted rating thường được xác định là trung bình có trọng số của các ratings đã chuẩn hoá. Công thức phổ biến được sử dụng để dự đoán rating của uu cho i là:

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

trong đó N(u,i) là tập hợp k users trong neighborhood (tức có similarity cao nhất) của u mà đã rated i.

Ví dụ tính normalized rating của u1 cho i1 được cho trong với số nearest neighbors là k=2. Các bước thực hiện là:

- ✓ Xác định các users đã rated i_1, đó là u_0, u_3, u_5.
- ✓ Xác định similarities của u_1 với các users này ta nhận được 0.83,−0.40,−0.23. Hai (k=2) giá trị lớn nhất là 0.83 và −0.23 tương ứng với u_0 và u_5.

Xác định các normalized ratings của u0, u5 cho i1, ta thu được hai giá trị lần lượt là 0.75 và 0.5.

Dự đoán kết quả:

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

Việc quy đổi các giá trị ratings đã chuẩn hoá về thang 5 có thể được thực hiện bằng cách cộng các cột của ma trận Dự đoán các (normalized) ratings còn thiếu với giá trị rating

trung bình của mỗi user

	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

	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

Việc hệ thống quyết định recommend items nào cho mỗi user có thể được xác định bằng nhiều cách khác nhau:

- ✓ Có thể sắp xếp unrated items theo thứ tự tự lớn đến bé của các predicted ratings,
- ✓ hoặc chỉ chọn các items có normalized predicted ratings dương - tương ứng với việc user này có nhiều khả năng thích hơn.

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 => Similarity matrix là rất lớn => việc lưu trữ ma trận này trong nhiều trường hợp là không khả thi.
- ✓ Ma trận Utility thường là rất sparse. Với số lượng users rất lớn so với số lượng items, rất nhiều cột của ma trận này sparse, tức chỉ có một vài phần tử khác 0 => tính toán ma trận Similarity, vốn tốn nhiều bộ nhớ và thời gian

Việc tính toán similarity giữa các items rồi recommend những items gần giống với item yêu thích của một user 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.

- ✓ Trong Utility matrix 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. => giá trị trung bình của mỗi hàng ít bị thay đổi hơn khi có thêm một vài ratings => việc cập nhật ma trận Similarity Matrix có thể được thực hiện ít thường xuyên hơn.

Quy trình dự đoán missing ratings cũng tương tự như trong User-user CF.

Về mặt tính toán, Item-item CF có thể nhận được từ User-user CF bằng cách chuyển vị (transpose) ma trận utility, và coi như items đang rate users. Sau khi tính ra kết quả cuối cùng, ta lại chuyển vị một lần nữa để thu được kết quả.

	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 $\bar{\mathbf{Y}}$.

5. KẾT LUẬN

Bài viết này đã giới thiệu một cái nhìn tổng quát về hệ thống gợi. Tác giả đã tóm lược các vấn đề cơ bản trong hệ thống gợi . (RS) cũng như các nhóm kỹ thuật phổ biến hiện nay trong RS, từ đó đi sâu vào trình bày chi tiết một vài kỹ thuật cho kết quả dự đoán tin cậy nhất hiện nay

TÀI LIỆU THAM KHẢO

[1]. Website
https://machinelearningcoban.com/2017/05/17/contentbasedrecommendersys/
[2]. Francesco Ricci · Lior Rokach · Bracha Shapira · Paul B. Kantor, "Recommender Systems Handbook", Springer Science+Business Media, LLC 2011