

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



BÁO CÁO
PROJECT 2

Đề tài:

Recommender Systems (Các Hệ Thống Gợi Ý)

**Hệ Thống Gợi Ý Phim Dựa Trên Lọc Cộng Tác (Collaborative Filtering) Và
Giải Thuật User K-Nearest-Neighbors**

Giảng viên hướng dẫn: **TS. Nguyễn Nhật Quang**

Sinh viên thực hiện: Lê Công Nguyên

MSSV: 20173290 – Lớp: KHMT.03

Hà Nội, 2020

Mục lục


I. Giới thiệu.....	2
1. Collaborative Filtering (CF).....	3
2. Neighborhood-based Collaborative Filtering (NBCF).....	4
3. Bộ dữ liệu MovieLens 100k.....	5
II. Hệ thống gợi ý NBCF.....	6
1. User-based CF.....	6
a. Similar Function.....	6
b. Rating Prediction.....	8
2. Item-based CF.....	8
3. Lập trình trên Python.....	10
a. Class CF.....	11
b. Áp dụng vào ví dụ.....	14
c. Áp dụng trên MovieLens 100k.....	15
III. Thảo luận.....	16
IV. Tài liệu tham khảo.....	17

I. Giới thiệu

Hệ thống gợi ý là một kỹ thuật **lọc thông tin** được dùng để **dự đoán** sở thích của người dùng, giúp người dùng ra quyết định và lựa chọn những mục tin phù hợp (mặt hàng, nhạc, phim, ảnh, tin tức, sách, ...) từ **bộ dữ liệu** có sẵn.


Hiện nay, các hệ thống gợi ý được dùng nhiều trong các lĩnh vực như thương mại điện tử, giải trí, giáo dục, ... Cụ thể, hệ gợi ý đóng một vai trò quan trọng trong các trang web nổi tiếng như Amazon.com, Youtube, Netflix, Yahoo, ...

Một số ví dụ về hệ thống gợi ý:

 INSTAGRAM

37 phút trước


na.lee_ và avo.circle đã theo dõi Lê Phương Thảo trên Instagram. Xem bài viết của họ.


 **VTV Giải trí** đã đăng một video vào danh sách phát Quỳnh Búp Bê.


1 Tháng 5 lúc 20:00 ·


Vì cô biết mình không thể nào thoát khỏi "Thiên Thai"

👉 Xem trọn bộ độc quyền "Quỳnh Búp Bê" trên VTV Giải Trí.


**MICHAEL MYERS: Con Quỷ Vô NHÂN TÍNH** Của HALLOWEEN
Phê Phim ✓
Recommended for you
13:07

**Top Những Đại Dịch Đáng Sợ Nhất Trên Màn Ảnh**
Phê Phim ✓
Recommended for you
HƠN CORONA 10:24


**10 CLOVERFIELD LANE: QUÁI VẬT TỒN TẠI Ở ĐÂU?**
Phê Phim ✓
Recommended for you
GIẢI THÍCH 10 CLOVERFIELD LANE 18:06

**AVENGERS: INFINITY WAR Được Tạo Ra NHƯ THẾ NÀO?**
Phê Phim ✓
Recommended for you
100% KỶ XẢO? 13:14

Sản phẩm liên quan



Apple iPhone 11 Pro, 256GB, Space Gray, Fully Unlocked...
★★★★☆ (17)
28,932,483 đ



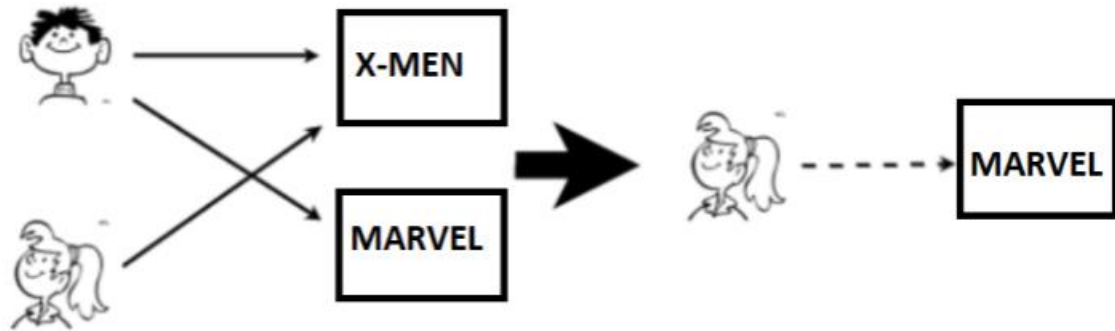
Apple iPhone Xs Max, 512GB, Space Gray - Fully Unlocked...
★★★★★ (1,267)
18,506,834 đ

Trên Netflix, hai phần ba số phim được xem đều được gợi ý.

Đề tài yêu cầu xây dựng một hệ thống gợi ý bộ phim nên xem với người dùng cụ thể một cách hiệu quả dựa trên **lọc cộng tác** với tập dữ liệu **MovieLens**.

1. Collaborative Filtering (CF)

Dưới đây là một mô tả đơn giản về cách hoạt động của hệ thống gợi ý khi được áp dụng vào gợi ý phim:



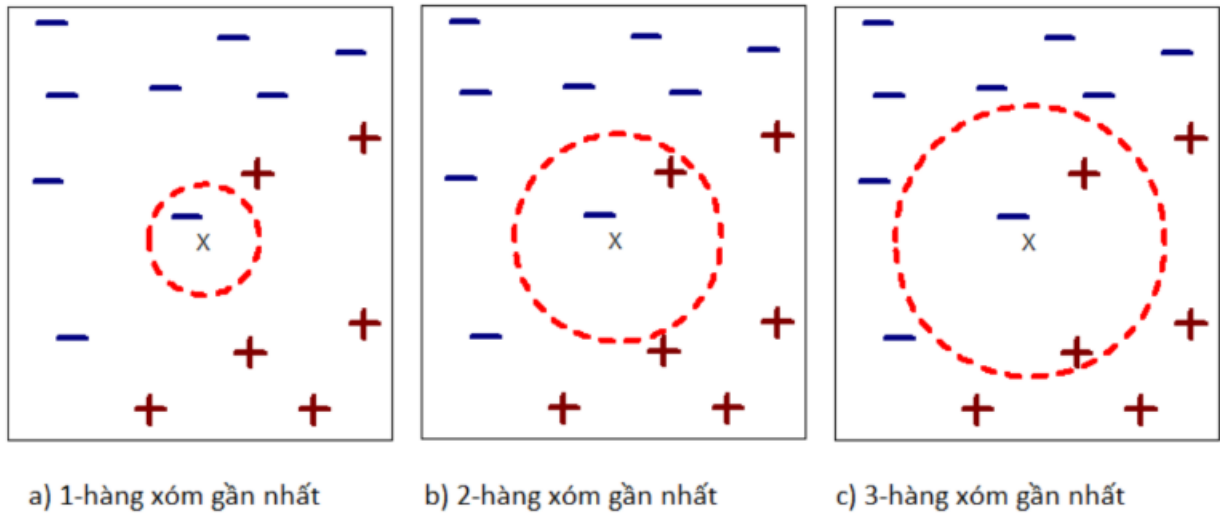
CF là kỹ thuật **dự đoán** đối với nội dung không thể được mô tả dễ dàng và đầy đủ bởi siêu dữ liệu, hoạt động bằng cách xây dựng một cơ sở dữ liệu sở thích về các mục tin theo người dùng. Sau đó kết hợp những người dùng với các sở thích và mối quan tâm của họ bằng cách tính toán các **độ tương tự** giữa những hồ sơ người dùng để tạo các gợi ý.

Có hai loại mô hình CF, đó là **Memory-based** và **Model-based**. Ưu điểm của kỹ thuật dựa trên bộ nhớ là đơn giản thực hiện và các khuyến nghị kết quả thường dễ giải thích. Chúng được chia thành hai:

- **User-based CF**: Các đối tượng được gợi ý là các đối tượng được người dùng yêu thích tương tự như người dùng. Ví dụ: A và B thích phim X-Men, A cũng thích phim Marvel thì hệ thống sẽ gợi ý phim đó cho B vì hệ thống xác định A và B giống nhau về sở thích.
- **Item-based CF**: Hệ thống xác định các mục tương tự dựa trên xếp hạng trước đây của người dùng. Ví dụ: A, B xếp hạng 5 sao cho phim X-Men và Marvel. Khi C xem phim X-Men thì hệ thống sẽ gợi ý phim Marvel cho C vì hệ thống xác định hai phim này giống nhau dựa trên xếp hạng của A và B.

2. Neighborhood-based Collaborative Filtering (NBCF)

Một đối tượng mới được quan sát sẽ được xếp cùng nhóm (lớp) với các đối tượng đã quan sát nếu ‘khoảng cách’ giữa chúng là nhỏ nhất có thể. Từ ‘khoảng cách’ được hiểu theo nghĩa rộng là độ đo sự giống nhau của đối tượng mới quan sát với các thành viên của nhóm.



Hình trên minh họa tiền đề của mô hình k – hàng xóm gần nhất với dữ liệu biểu diễn đối tượng mới quan sát được đánh dấu x, còn các đối tượng đã biết được chia thành hai nhóm (đánh dấu +, -).

Ý tưởng cơ bản của NBCF là xác định mức độ quan tâm của một user tới một item dựa trên các users khác tương tự với user này. Việc tương tự giữa các user có thể xác định thông qua mức độ quan tâm của các users này tới các items khác mà hệ thống đã biết.

Việc xác định mức độ quan tâm của mỗi user tới một item có hai hướng tiếp cận là dựa trên mức độ quan tâm của similar users tới item đó (User-based CF) và dựa trên similar items (Item-based CF), gợi ý các items tương tự với những items mà user có mức độ quan tâm cao.

3. Bộ dữ liệu MovieLens 100k

Bộ cơ sở dữ liệu MovieLens 100k được công bố năm 1998 bởi **GroupLens**, bao gồm 100,000 (100k) ratings từ 943 users cho 1682 movies.

Trong bộ cơ sở dữ liệu này gồm rất nhiều tập tin nhỏ, một trong số các tập tin này gồm:

- **u.data**: Chứa toàn bộ các ratings của 943 users cho 1682 movies. Mỗi user rate ít nhất 20 movies.
- **u.genre**: Chứa tên của 19 thể loại phim.
- **u.user**: Chứa thông tin về users: id, tuổi, giới tính, nghề nghiệp, zipcode. Đề tài sẽ chỉ sử dụng thông tin về id để xác định các user khác nhau.

```
4 # Reading user file
5 u_cols = ['user_id', 'age', 'occupation', 'zip_code']
6 users = pd.read_csv('ml-100k/u.user', sep='|', names=u_cols, encoding='latin-1')
7 n_users = users.shape[0]
8 print('Number of users: ',n_users)
9 print(users.head())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2: Python D

```
Number of users: 943
  user_id age occupation zip_code
1      24  M  technician  85711
2      53  F    other    94043
3      23  M    writer    32067
4      24  M  technician  43537
5      33  F    other    15213
```

- **u.item**: Chứa thông tin về mỗi bộ phim: id, tên phim, ngày phát hành, link trên IMDB và các số nhị phân 0, 1 để chỉ ra bộ phim thuộc các thể loại trong 19 thể loại đã cho trong **u.genre**.

```
20 # Reading items file:
21 i_cols = ['movie id', 'movie title', 'release date', 'video release date', 'IMDb URL', 'unknown', 'Action', 'Adventure',
22          'Animation', 'Children's', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror', 'Musical',
23          'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western']
24 items = pd.read_csv('ml-100k/u.item', sep='|', names=i_cols, encoding='latin-1')
25 n_items = items.shape[0]
26 print('Number of items:', n_items)
27 print(items.head())
28
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2: Python Debug Consc

```
Number of items: 1682
  movie id  movie title release date video release date ... Sci-Fi Thriller War Western
0         1  Toy Story (1995)  01-Jan-1995             NaN ...      0      0      0      0
1         2  GoldenEye (1995)  01-Jan-1995             NaN ...      0      1      0      0
2         3   Four Rooms (1995)  01-Jan-1995             NaN ...      0      1      0      0
3         4  Get Shorty (1995)  01-Jan-1995             NaN ...      0      0      0      0
4         5   Copycat (1995)  01-Jan-1995             NaN ...      0      1      0      0
```

[5 rows x 24 columns]

II. Hệ thống gợi ý NBCF

1. User-based CF

a. Similar Function

Công việc quan trọng nhất trước tiên là phải xác định được similarity giữa hai users.

Dữ liệu chúng ta có là **Utility matrix Y**, vậy nên similarity được xác định dựa trên các cột tương ứng với hai users trong ma trận.

	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: users từ u_0 đến u_6 và items từ i_0 đến i_4 trong đó các số trong mỗi ô vuông thể hiện số sao mà user đã rate 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 tìm. Đặt mức độ giống nhau của hai users u_i và u_j là **sim(u_i, u_j)**

$$sim(u_i, u_j) = \frac{\sum_{p \in P} (r_{u_i, p} - \bar{r}_{u_i}) (r_{u_j, p} - \bar{r}_{u_j})}{\sqrt{\sum_{p \in P} (r_{u_i, p} - \bar{r}_{u_i})^2} \sqrt{\sum_{p \in P} (r_{u_j, p} - \bar{r}_{u_j})^2}}$$

u_i, u_j : users

$r_{u, p}$: rating of user for item p

P : set of items, rated both by u_i and u_j

$$-1 \leq sim(u_i, u_j) \leq 1$$

Ví dụ mô tả User-based CF:

	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

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
\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** $\{0.75, 0.5\}$

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

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 $\hat{\mathbf{Y}}$

Chuẩn hóa dữ liệu: Hàng cuối cùng ở hình a là giá trị trung bình ratings của mỗi user. Giá trị cao tương ứng với các user dễ tính và ngược lại. Nếu tiếp tục trừ từ mỗi rating đi giá trị này và thay các giá trị chưa biết bằng 0, ta sẽ được normalized utility matrix như hình b. Bước chuẩn hóa này là quan trọng vì giúp tối ưu bộ nhớ và dễ dàng tính toán. Sau khi đã chuẩn hóa dữ liệu, ta có:

$$\text{sim}(u_i, u_j) = \text{cosine_similarity}(u_i, u_j) = \cos(u_i, u_j) = \frac{u_i^T \cdot u_j}{\|u_i\| \cdot \|u_j\|}$$

u_{ij} là vectors tương ứng với users i, j đã được chuẩn hóa.

b. Rating Prediction

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

$\mathcal{N}(u, i)$ là tập hợp k users trong neighborhood (tức có similarity cao nhất) của u mà đã rate cho i.

Việc tính normalized rating của u_1 cho i_1 được cho trong hình e phần a, hình d thể hiện việc điền các giá trị còn thiếu trong normalized utility matrix. Các ô nền đỏ thể hiện các giá trị dương, tức các items mà có thể users đó quan tâm.

Quy đổi các giá trị ratings đã chuẩn hóa về thang 5 bằng cách cộng các cột của ma trận \hat{Y} với giá trị rating trung bình của mỗi user trong hình a thu được hình f.

Việc hệ thống quyết định recommend items nào cho mỗi user có thể được xác định bằng nhiều cách. Có thể xếp theo thứ tự từ lớn đến bé của các predicted ratings, hoặc chỉ chọn các items có normalized predicted ratings dương.

2. Item-based CF

Một số nhược điểm của User-based CF:

- Trên thực tế, số lượng users luôn lớn hơn số lượng items rất nhiều, kéo theo là Similarity matrix là rất lớn.
- Ma trận Utility Y thường rất sparse (thưa thớt, rải rác). Với số lượng users rất lớn so với số lượng items, rất nhiều cột của ma trận này sẽ rất sparse, tức là chỉ có một vài phần tử khác 0. Hơn nữa, khi user 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 là việc tính toán tốn nhiều bộ nhớ và thời gian.

Ngược lại, việc tính toán similarity giữa các items rồi recommend những items gần giống với item yêu thích của một user có những ưu điểm sau:

- Similarity matrix nhỏ hơn nhiều, thuận lợi cho việc tính toán và lưu trữ
- Số hàng (items) ít hơn số cột (users) trong Utility matrix nên trung bình mỗi hàng của ma trận sẽ có nhiều phần tử đã biết hơn số phần tử đã biết trong cột. Từ đó, giá trị trung bình của mỗi hàng ít bị thay đổi hơn và việc cập nhật Similarity matrix ít thường xuyên hơn.

Hướng tiếp cận này được sử dụng nhiều hơn trong thực tế.

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

Ví dụ mô tả Item-based CF:

	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 \bar{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	-0.6	-2.6	-1.6	0	0
i_1	2	0	0	-2	0	0	0
i_2	0	2.25	-0.75	0	0	-0.75	-0.75
i_3	-1.17	-1.17	-0.17	0.83	0.83	0	0.83
i_4	-0.75	-2.75	1.25	0	0	0	2.25

b) Normalized utility matrix \bar{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 S .

	u_0	u_1	u_2	u_3	u_4	u_5	u_6
i_0	2.4	2.4	-0.6	-2.6	-1.6	-0.29	-1.52
i_1	2	2.4	-0.6	-2	-1.25	0	-2.25
i_2	2.4	2.25	-0.75	-2.6	-1.20	-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{Y} .

Có một điểm thú vị trong hình c là có các phần tử trong hai hình vuông xanh và đỏ đều là các số không âm, các phần tử bên ngoài là các số âm. Một cách vô tình, các items đã được **clustering** (phân cụm).

Việc hệ thống quyết định recommend items nào cho mỗi user có thể xác định bởi các ô nền đỏ trong hình d. Kết quả này có khác một chút so với kết quả tìm được bởi User-based CF.

Về mặt tính toán, Item-based CF có thể nhận được từ User-based CF bằng cách **transpose** (chuyển vị) Utility matrix. Sau khi tính ra kết quả cuối cùng, ta transpose một lần nữa để thu được kết quả.

3. Lập trình trên Python

File **ex.dat** mô tả dữ liệu đã biết cho ví dụ. Thứ tự ba cột là user_id, item_id, rating. Ví dụ, hàng đầu tiên nghĩa là u_0 rate i_0 số sao là 5.

```
1  0 0 5.
2  0 1 4.
3  0 3 2.
4  0 4 2.
5  1 0 5.
6  1 2 4.
7  1 3 2.
8  1 4 0.
9  2 0 2.
10 2 2 1.
11 2 3 3.
12 2 4 4.
13 3 0 0.
14 3 1 0.
15 3 3 4.
16 4 0 1.
17 4 3 4.
18 5 1 2.
19 5 2 1.
20 6 2 1.
21 6 3 4.
22 6 4 5.
```

Khi làm việc với Item-based CF, chúng ta chỉ cần đổi vị trí của hai cột đầu tiên.

a. Class CF

Class CF được sử dụng chung cho cả User-based CF và Item-based CF.

Dữ liệu đầu vào là ma trận Utility Y_data được lưu dưới dạng một ma trận với 3 cột, k là số lượng các điểm lân cận được sử dụng để dự đoán kết quả. Hàm **dist_func** tính similarity giữa hai vectors, mặc định là **cosine_similarity** được lấy từ **sklearn.metrics.pairwise**, python hỗ trợ tính toán hàm số này một cách hiệu quả. Biến **uuCF** thể hiện việc đang sử dụng User-based CF (1) hay Item-based CF (0).

```
33 # Class CF
34 class CF(object):
35
36     def __init__(self, Y_data, k, dist_func = cosine_similarity, uuCF = 1):
37         self.uuCF = uuCF #user-based (1) or item-based (0)
38         self.Y_data = Y_data if uuCF else Y_data[:, [1, 0, 2]] # convert user_id and item_id
39         self.k = k # number of neighbor points
40         self.dist_func = dist_func
41         self.Ybar_data = None
42         self.n_users = int(np.max(self.Y_data[:, 0])) + 1 # id starts from 0
43         self.n_items = int(np.max(self.Y_data[:, 1])) + 1
```

Khi có dữ liệu mới, cập nhật Utility matrix bằng cách thêm các hàng này vào cuối ma trận.

```
45     def add(self, new_data):
46         self.Y_data = np.concatenate((self.Y_data, new_data), axis = 0)
```

Tính toán Normalized utility matrix và Similarity matrix.

```
48     def normalize_Y(self):
49         users = self.Y_data[:, 0] # all users - first col of the Y_data
50         self.Ybar_data = self.Y_data.copy()
51         self.mu = np.zeros((self.n_users,))
52         for n in range(self.n_users):
53             # row indices of rating done by user n
54             ids = np.where(users == n)[0].astype(np.int32)
55             # indices of all ratings associated with user n
56             item_ids = self.Y_data[ids, 1]
57             # and the corresponding ratings
58             ratings = self.Y_data[ids, 2]
59             m = np.mean(ratings)
60             if np.isnan(m):
61                 m = 0 # to avoid empty array and nan value
62             self.mu[n] = m
63             # normalize
64             self.Ybar_data[ids, 2] = ratings - self.mu[n]
65         self.Ybar = sparse.coo_matrix((self.Ybar_data[:, 2], (self.Ybar_data[:, 1],
66             self.Ybar_data[:, 0])), (self.n_items, self.n_users))
67         self.Ybar = self.Ybar.tocsr()
68
69     def similarity(self):
70         self.S = self.dist_func(self.Ybar.T, self.Ybar.T)
```

Thực hiện lại hai hàm phía trên khi có thêm dữ liệu.

```
72     def refresh(self):
73         self.normalize_Y()
74         self.similarity()
75
76     def fit(self):
77         self.refresh()
```

Dự đoán kết quả.

```
79     def __pred(self, u, i, normalized = 1):
80         # step 1: find all users who rated i
81         ids = np.where(self.Y_data[:, 1] == i)[0].astype(np.int32)
82         # step 2:
83         usersRated_i = (self.Y_data[ids, 0]).astype(np.int32)
84         # step 3: find similarity the current user and others who rated i
85         sim = self.S[u, usersRated_i]
86         # step 4: find the k most similarity users
87         a = np.argsort(sim)[- self.k:]
88         # and the corresponding similarity levels
89         nearest_s = sim[a]
90         r = self.Ybar[i, usersRated_i[a]]
91         if normalized:
92             return (r*nearest_s)[0]/(np.abs(nearest_s).sum() + 1e-8) # to avoid dividing by 0
93         return (r*nearest_s)[0]/(np.abs(nearest_s).sum() + 1e-8) + self.mu[u]
94
95     def pred(self, u, i, normalized = 1):
96         if self.uuCF: return self.__pred(u, i, normalized)
97         return self.__pred(i, u, normalized)
```

Tìm tất cả các items nên được gợi ý cho user u trong trường hợp User-based (uuCF = 1) và ngược lại dựa trên **self.pred(u,i) > 0**.

```
99     def recommend(self, u, normalized = 1):
100         ids = np.where(self.Y_data[:, 0] == u)[0]
101         itemsRated_by_u = self.Y_data[ids, 1].tolist()
102         recommended_items = []
103         for i in range(self.n_items):
104             if i not in itemsRated_by_u:
105                 rating = self.__pred(u, i)
106                 if rating > 0:
107                     recommended_items.append(i)
108         return recommended_items
```

In ra kết quả.

```
110     def print_recommendation(self):
111         print('Recommendation: ')
112         for u in range(self.n_users):
113             recommended_items = self.recommend(u)
114             if self.uuCF:
115                 print('    Recommend items(s): ', recommended_items, ' to user', u)
116             else:
117                 print('    Recommend item ', u, ' to user(s): ', recommended_items)
118
```

b. Áp dụng vào ví dụ

```
119 # Data file Example
120 r_cols = ['user_id', 'item_id', 'rating']
121 ratings = pd.read_csv('ex.dat', sep = ' ', names = r_cols, encoding = 'latin-1')
122 Y_data = ratings.values
```

- Với User-based CF:

```
124 # User-based CF
125 rs = CF(Y_data, k = 2, uuCF = 1)
126 rs.fit()
127 rs.print_recommendation()
```

Ta có kết quả:

```
Recommendation:
Recommend items(s): [2] to user 0
Recommend items(s): [1] to user 1
Recommend items(s): [] to user 2
Recommend items(s): [4] to user 3
Recommend items(s): [4] to user 4
Recommend items(s): [0, 3, 4] to user 5
Recommend items(s): [1] to user 6
```

- Với Item-based CF:

```
129 # Item-based CF
130 rs = CF(Y_data, k = 2, uuCF = 0)
131 rs.fit()
132 rs.print_recommendation()
```

Ta có kết quả:

```
Recommendation:
Recommend item 0 to user(s): []
Recommend item 1 to user(s): [1]
Recommend item 2 to user(s): [0]
Recommend item 3 to user(s): [5]
Recommend item 4 to user(s): [3, 4, 5]
```

c. Áp dụng trên MovieLens 100k

Trước tiên, cần load dữ liệu:

```
136 # Load data MovieLens 100k
137 r_cols = ['user_id', 'movie_id', 'rating', 'unix_timestamp']
138
139 ratings_base = pd.read_csv('ml-100k/ub.base', sep = '\t', names = r_cols, encoding = 'latin-1')
140 ratings_test = pd.read_csv('ml-100k/ub.test', sep = '\t', names = r_cols, encoding = 'latin-1')
141
142 rate_train = ratings_base.values
143 rate_test = ratings_test.values
144
145 rate_train[:, :2] -= 1
146 rate_test[:, :2] -= 1
```

Chia dữ liệu thành tập huấn luyện (ub.base) và tập kiểm tra (ub.test).

Đánh giá mô hình bằng kỹ thuật **Root Mean Squared Error (RMSE)**.

- Với **User-based CF**:

```
146 # User-based CF
147 rs = CF(rate_train, k = 30, uuCF = 1)
148 rs.fit()
149 n_tests = rate_test.shape[0]
150 SE = 0 # Squared Error
151 for n in range(n_tests):
152     pred = rs.pred(rate_test[n, 0], rate_test[n, 1], normalized = 0)
153     SE += (pred - rate_test[n, 2])**2
154 RMSE = np.sqrt(SE/n_tests)
155 print('User-based CF, RMSE = ', RMSE)
```

Ta có kết quả:

```
User-based CF, RMSE = 0.9951981100882598
```


- Với **Item-based CF**:

```

157 # Item-based CF
158 rs = CF(rate_train, k = 30, uuCF = 0)
159 rs.fit()
160 n_tests = rate_test.shape[0]
161 SE = 0
162 for n in range(n_tests):
163     pred = rs.pred(rate_test[n, 0], rate_test[n, 1], normalized = 0)
164     SE += (pred - rate_test[n, 2])**2
165 RMSE = np.sqrt(SE/n_tests)
166 print('User-based CF, RMSE = ', RMSE)

```

Ta có kết quả:

```
Item-based CF, RMSE = 0.9867912132705384
```

Item-based CF cho lỗi nhỏ hơn so với User-based CF

III. Thảo luận

- Có thể nói các hệ thống gợi ý áp dụng các giải thuật khai phá dữ liệu, học máy nhằm giúp thu thập thông tin cá nhân trên Internet, đồng thời giúp giảm bớt vấn đề quá tải thông tin với các hệ thống truy xuất thông tin và cho phép người dùng truy cập vào các sản phẩm và dịch vụ trên hệ thống.
- **CF** là một phương pháp gợi ý sản phẩm với ý tưởng chính dựa trên các hành vi của users khác (**collaborative**) cùng trên một item để suy ra mức độ quan tâm (**filtering**) của một user lên item. Việc suy ra này được thực hiện dựa trên **Similarity matrix** đo độ giống nhau giữa các users.
- Để tính được Similarity matrix, trước tiên ta cần **chuẩn hóa dữ liệu**.
- **Similarity function** được dùng là **Cosine similarity**
- **User-based CF** có một vài hạn chế khi lượng users là lớn. Trong các trường hợp đó, **Item-based** thường được sử dụng và cho kết quả tốt hơn.
- **Source code:** <https://github.com/nguyenlecong/project2>

IV. Tài liệu tham khảo

- 1) Chapter 02 – Collaborative recommendation

<http://www.recommenderbook.net/teaching-material/slides>

- 2) Chương 5: Phân loại và dự báo – Nhập môn học máy và khai phá dữ liệu – TS.
Trịnh Anh Phúc

https://drive.google.com/file/d/1Rrn0tChIZTx1ZQIYtkWf6_VVzqxTsDA4/view

- 3) Machine Learning cơ bản – Bài 24: Neighborhood-Based Collaborative Filtering

https://machinelearningcoban.com/2017/05/24/collaborativefiltering/?fbclid=IwAR0g6cHtcaE82nJAOWXEgw9C2-psilrFwq48Lj6qTa6iBVufuZ_nKVaj6s4

- 4) Xây dựng hệ thống gợi ý bằng thuật toán người láng giềng và thử nghiệm trên Movielens dataset

https://www.researchgate.net/publication/322519165_XAY_DUNG_HE_THONG_GOI_Y_BANG_THUAT_TOAN_NGUOI_LANG_GIENG_VA_THU_NGHIEM_TREN_MOVIELENS_DATASET

- 5) Xây dựng một hệ thống gợi ý phim đơn giản với Python

<https://viblo.asia/p/xay-dung-mot-he-thong-goi-y-phim-don-gian-voi-python-eW65Ge1PZDO>