**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**


# BÁO CÁO ĐỒ ÁN MÔN HỌC

## Nhập môn Học máy và Khai phá dữ liệu

## Đề tài: Movie Recommedation System

Giảng viên hướng dẫn: *PGS. Thân Quang Khoát*

Mã lớp : *131404*

Nhóm sinh viên thực hiện*:* *Cao Như Đạt* – *20194013*

*Nguyễn Huy Hoàng* – *20194058*

*Lê Huy Hoàng* – *20190053*

# MỤC LỤC

# Lời nói đầu

Hiện nay, hầu khắp trên các ứng dụng hay phương tiện truyền thông chúng ta đều nhận thấy sự xuất hiện của Recommendation System (Hệ thống gợi ý). Khi ta xem một video, YouTube tự động gợi ý các video có nội dung liên quan hoặc có thể tự chuyển tiếp khi ta xem xong. Tương tự với Neflix hay Spotify, những bộ phim hay bản nhạc mà họ gợi ý có ý nghĩa rất lớn trong sự phát triển mạnh mẽ của họ trong thời gian gần đây.

Nhận thấy đây là một đề tài thú vị và có nhiều ý nghĩa trong thực tiễn, nhóm chúng em quyết định lựa chọn đề tài Movie Recommendation System (Hệ thống gợi ý phim) làm đề tài báo cáo của môn học Nhập môn Học máy và Khai phá dữ liệu. Quá trình thực hiện đề tài không thể tránh khỏi thiếu sót, chúng em rất mong nhận được những sự góp ý của thầy!

# 1. Giới thiệu bài toán và các khái niệm cơ bản

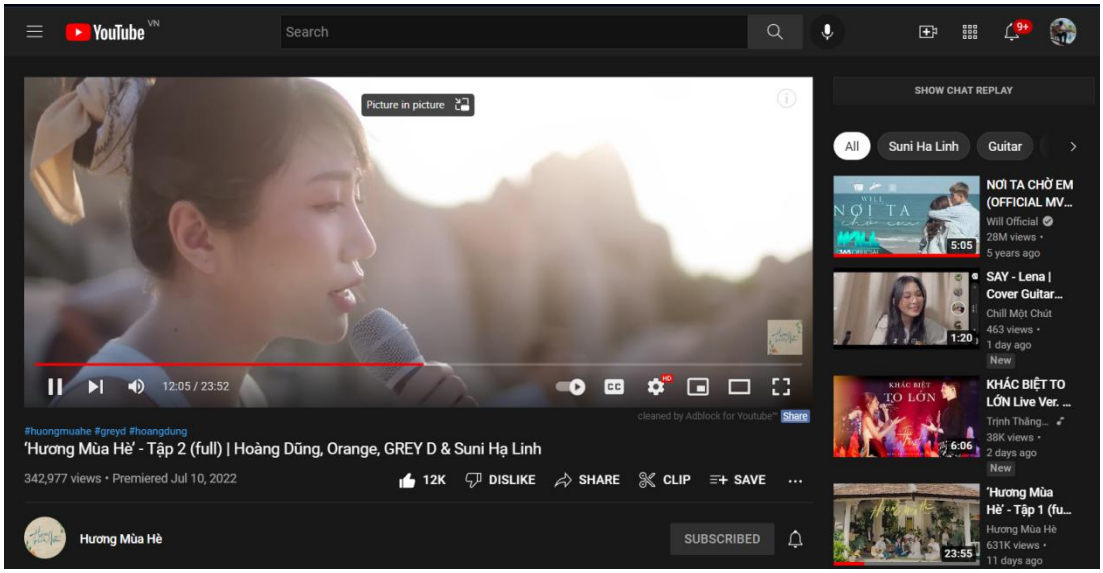
## 1.1. Bài toán Movie Recommender System

### a. Mô tả chung

Một *hệ thống gợi ý* (hay cũng có thể gọi là *hệ thống khuyên dùng*, tiếng anh: *recommendation system*) là một lớp con của *hệ thống lọc thông tin*, tìm kiếm dự đoán "đánh giá" hoặc "ưa thích" của người dùng với một sản phẩm hoặc đối tượng nào đó.

Các hệ thống gợi ý được tận dụng trong nhiều lĩnh vực, chẳng hạn như trình tạo danh sách phát cho video và âm nhạc của một số ứng dụng như Netflix, YouTube và Spotify. Sau khi người xem một vài video hoặc nghe một số bài hát, hệ thống gợi ý sẽ nhận diện sở thích người dùng và mức độ tương tác với video/bài hát để tạo ra một danh sách gợi ý theo chủ đề và mức độ liên quan mà người dùng có thể ưa thích. Hệ thống gợi ý còn áp dụng cho các dịch vụ gợi ý sản phẩm ở Amazon, các gợi ý nội dung cho các nền tảng xã hội như Facebook hay Twitter.

Youtube tự động đề xuất các video khác liên quan khi người dùng sử dụng:




Hệ thống gợi ý (Recommendation system) dần trở thành một thành phần không thể thiếu của các sản phẩm điện tử có nhiều người dùng. Các sản phẩm cá nhân hóa điện tử ngày càng phổ biến với mục đích mang sản phẩm phù hợp tới người dùng hoặc giúp người dùng có các trải nghiệm tốt hơn. Nếu quảng cáo sản phẩm tới đúng người dùng, khả năng các món hàng được mua nhiều hơn. Nếu gợi ý một video mà người dùng nhiều khả năng thích hoặc gợi ý kết bạn đến đúng đối tượng, họ sẽ ở lại trên nền tảng của bạn lâu hơn. Khi họ ở trên nền tảng của bạn lâu hơn, họ sẽ nhìn thấy nhiều quảng cáo hơn và lợi nhuận từ quảng cáo sẽ lại càng nhiều hơn.

Quảng cáo điện tử ngoài việc giúp các doanh nghiệp bán được nhiều hàng còn giúp họ tiết kiệm được chi phí kho bãi. Họ sẽ không cần các cửa hàng ở vị trí thuận lợi để thu hút khách hàng hay phải trưng ra mọi mặt hàng ở vị trí đắc địa nhất trong cửa hàng. Mọi thứ có thể được cá nhân hóa sao cho mỗi người dùng nhìn thấy những sản phẩm khác nhau phù hợp với nhu cầu và sở thích của họ.

Quảng cáo trên Internet đã chiếm thị phần ngày càng cao so với quảng cáo truyền hình nhờ sự đa dạng và cá nhân hóa một cách tối đa. Một người dùng 20-30 tuổi thường xuyên nghe nhạc rap ít có khả năng thích nhạc Bolero. Một người dùng tìm kiếm các thông tin về xe hơi nhiều khả năng sắp mua xe và quan tâm tới những dịch vụ sửa và rửa xe. Một người dùng thường xuyên xe các video về làm vườn nhiều khả năng sẽ quan tâm tới việc mua bán hạt giống. Từ những thông tin thu thập được từ hành vi người dùng, hệ thống có thể gợi ý ra những lựa chọn phù hợp để đạt được hiệu quả cao nhất.

## b. Phân loại

Các hệ thống gợi ý thường được chia thành 4 nhóm chính:

***- Lọc Nhân khẩu học (Demographic Filtering):*** Đưa ra các đề xuất tổng quát cho mọi người dùng, dựa trên mức độ phổ biến và / hoặc thể loại phim. Hệ thống đề xuất những bộ phim tương tự cho những người dùng có các đặc điểm nhân khẩu học tương tự. Vì mỗi người dùng khác nhau nên cách làm này được coi là quá đơn giản. Ý tưởng cơ bản đằng sau hệ thống này là những bộ phim nổi tiếng hơn và được giới phê bình đánh giá cao hơn sẽ có xác suất được khán giả bình thường thích cao hơn.

**- Lọc dựa trên nội dung (Content-based Filtering):** nhóm thuật toán này gợi ý cho người dùng những sản phẩm tương tự như những sản phẩm mà người dùng đã có phản hồi tích cực. Hệ thống này cần xây dựng đặc trưng cho các sản phẩm sao cho những sản phẩm tương tự nhau có khoảng cách tới nhau nhỏ. Việc này khá tương tự như việc xây dựng các embedding cho các sản phẩm. Việc dự đoán cho mỗi người dùng hoàn toàn chỉ dựa trên lịch sử thông tin của người dùng đó.

**- Lọc cộng tác (Collaborative Filtering)**: nhóm thuật toán này không chỉ dựa trên thông tin về sản phẩm tương tự mà còn dựa trên hành vi của những người dùng tương tự. Ví dụ: người dùng A, B, C đều thích các bài hát của Noo Phước Thịnh. Ngoài ra, hệ thống biết rằng B, C cũng thích các bài hát của Bích Phương nhưng chưa có thông tin về việc liệu user A có thích Bích Phương hay không. Dựa trên thông tin của những người dùng tương tự là B và C, hệ thống có thể dự đoán rằng A cũng thích Bích Phương và gợi ý các bài hát của ca sĩ này tới A.

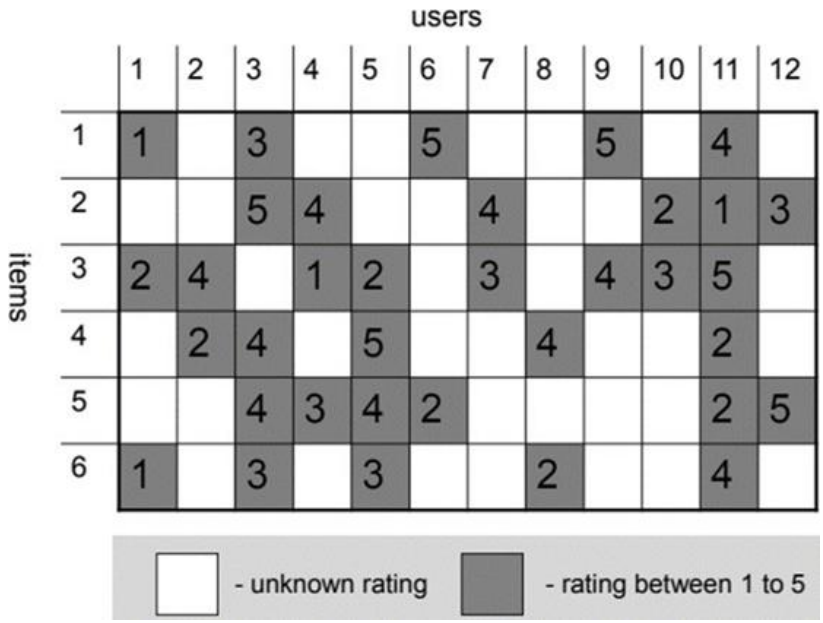
**- Lọc cộng tác kết hợp (Hybrid Collaborative Filtering):** Hệ thống hybrid CF kết hợp CF với các kỹ thuật đề xuất khác (thường là với các hệ thống dựa trên nội dung (content-based system) để đưa ra các dự đoán hoặc khuyến nghị **.**

*Trong phạm vi nội dung của bài báo cáo, nhóm chúng em chỉ tập trung vào hệ thống gợi ý sử dụng phương pháp* ***Lọc dựa trên nội dung (Content-based Filtering)*** *và* ***Lọc cộng tác (Collaborative Filtering)****.*

## 1.2. Các khái niệm cơ bản

### a. Rating matrix

Trong hệ thống gợi ý, có hai thực thể chính là users (những người dùng) và items (những sản phẩm). Mỗi user sẽ có mức độ quan tâm (degree of preference) tời 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 đánh giá 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 ratings matrix.




Thông thường, có rất nhiều users và items trong hệ thống, và mỗi user thường chỉ đánh giá một số lượng rất nhỏ các item, thậm chí có những user không đánh giá item nào. Vì vậy, lượng ô trống của ratings 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 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.

## b. Bias (độ lệch hay thiên vị)

Trong thực tế, các ratings đều có những thiên lệch tùy theo cá nhân users hoặc items. Có user dễ và khó tính, cũng có những item được rated cao hơn những items khác chỉ vì user thấy các users khác đã đánh giá item đó cao rồi.

Vấn đề thiên lệch có thể được giải quyết bằng các biến gọi là biases, phụ thuộc vào mỗi user và item. Biases đối với user là giá trị chênh lệch giữa đánh giá trên các sản phẩm khác của người dùng với trung bình kết quả đánh giá của những người dùng khác. Đối với item là giá trị chênh lệch giữa đánh giá của sản phẩm này với trung bình kết quả đánh giá các sản phẩm khác. Đồng thời giá trị biases có thể phụ thuộc vào giá trị trung bình của toàn bộ ratings.

## c, Đánh giá hệ gợi ý

- Các tiêu chí định lượng:

i. Đánh giá độ chính xác của hàm dự đoán xếp hạng (rating prediction):

- Sai số bình phương trung bình (Mean square error – MSE)
- Căn của sai số bình phương trung bình (Root mean square error – RMSE)
- Sai số trung bình tuyệt đối (Mean absolute error – MAE)

ii. Đánh giá độ chính xác của hàm gợi ý (item recommendation):

Gợi ý được xem là phù hợp khi người dùng chọn sản phẩm từ danh sách các sản phẩm đã được hệ thống gợi ý cho người dùng. Sự phù hợp này có thể được đánh giá qua các độ đo như:

- Precision
- Recall
- F1-Score

*Ở trong phạm vi project, chúng em áp dụng **RMSE** để đánh giá độ chính xác của hàm dự đoán.*

- Các tiêu chí định tính:

iii. Tính mới của các gợi ý
iv. Tính đa dạng (Diversity) của các gợi ý
v. Độ bao phủ của các gợi ý
vi. Sự hài lòng của người dùng

## d, Lựa chọn tham số cho mô hình

Để lựa chọn tham số cho mô hình ta cần xây dựng tập valid tách ra từ tập train để đánh giá tham số hiện tại.

Trong project, do số lượng đánh giá của từng người dùng không giống nhau nên chúng em sử dụng stratified sampling để lựa chọn tham số bằng cách sau: ứng với mỗi người dùng đánh giá, ta lấy ra 20% đánh giá để làm tập test và 80% còn lại để huấn luyện mô hình. Trong tập huấn luyện ta tách ra tiếp 20% để làm tập đánh giá

# 2. Bộ dữ liệu

## 2.1 Tổng quan về bộ dữ liệu TMDB

Bộ dữ liệu TMDB được tạo từ The Movie Database API và được công khai trên Kagle: TMDB 5000 Movie Dataset | Kaggle. Bộ dữ liệu mang các thông tin chi tiết của gần 5000 bộ phim trong hai file csv chứa các trường thông tin sau:

**tmdb_5000_movies.csv:**

- *budget:* ngân sách của phim
- *genres:* thể loại phim
- *homepage:* trang chủ của phim
- *id:* mã phim
- *keywords:* từ khóa
- *original_language:* ngôn ngữ ban đầu sử dụng trong phim
- *original_title:* tiêu đề phim ban đầu
- *overview:* tổng quan về phim
- *popularity:* độ phổ biến của phim
- *production_companies:* hãng phim
- *production_countries:* quốc gia
- *release_date:* ngày phát hành
- *revenue:* thu nhập
- *runtime:* thười lượng phim
- *spoken_languages:* ngôn ngữ nói trong phim
- *status:* trạng thái hiện tại của phim
- *tagline:* khẩu hiệu cho phim
- *title:* tiêu đề
- *vote_average:* điểm bình chọn trung bình
- *vote_count:* số lượng người bình chọn

**tmdb_5000_credits.csv:**

- *movie_id:* mã phim (tương ứng với id của file trên)

- *title:* tiêu đề phim

- *cast:* dàn diễn viên (bao gồm id, tên trong phim, tên thật)

- *crew:* đoàn làm phim (bao gồm id, tên, bộ phận, công việc)

***Nhận xét: Bộ dữ liệu IMDB 5000 đã rất là đầy đủ nhưng các thông tin còn lẫn với nhau chưa rõ ràng. Cần đưa dữ liệu về dạng dễ truy xuất hơn***

## 2.2. Tổng quan về bộ dữ liệu Movie Lens 100K

### a, Mô tả chung

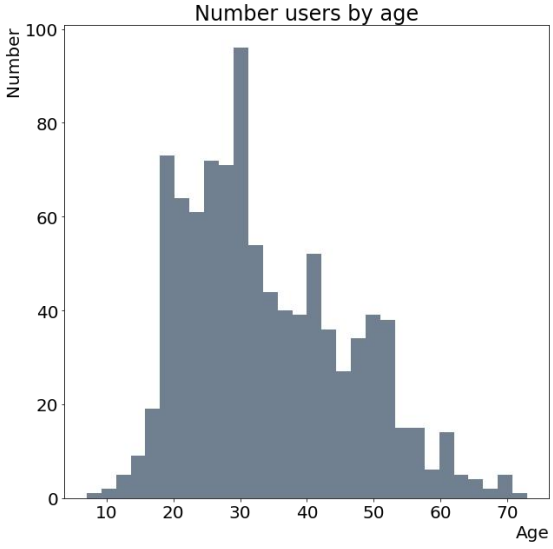
- Bộ dữ liệu MovieLens được thu thập bởi GroupLens Research Project tại Đại học Minnesota, đăng công khai tại GroupLens.

- Bộ dữ liệu bao gồm:
  - 100000 đánh giá (1-5) từ 943 người dùng trên 1682 bộ phim
  - Mỗi người dùng đánh giá ít nhất 20 bộ phim
  - Dữ liệu thu nhập trong thời gian 7 tháng từ 19/9/1997 - 22/4/1998
  - Thông tin về các bộ phim bao gồm: tên, thể loại, ngày ra mắt, link thông tin
  - Thông tin về người dùng bao gồm: giới tính, tuổi, zipcode, nghề nghiệp

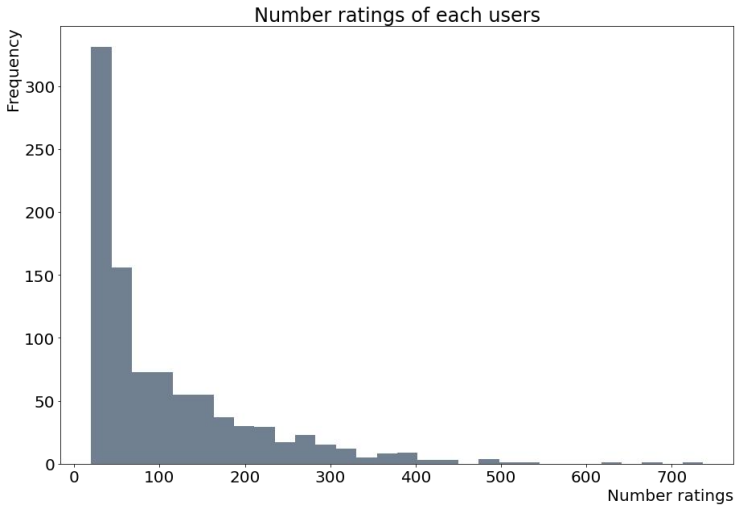
### b, EDA

***- Về người dùng:***

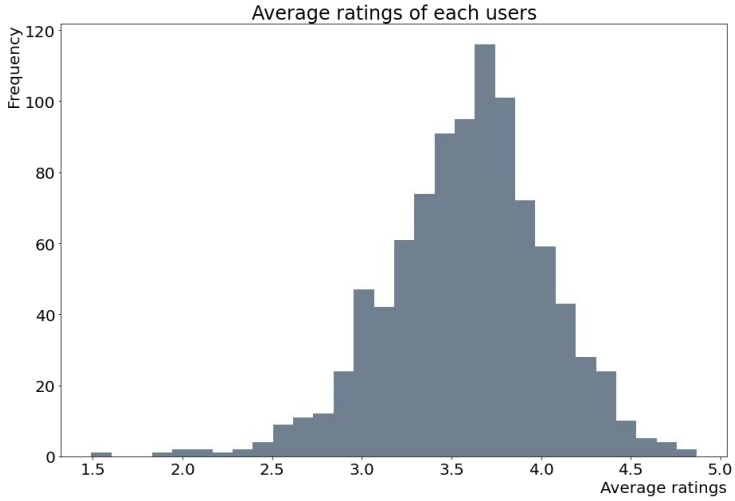
- Người dùng có độ tuổi từ 7 đến 73 tuổi. Số lượng người tập trung chủ yếu ở khoảng 30 tuổi.



- Một người đánh giá từ 20 đến 737 bộ phim. Trong đó trung bình 1 người đánh giá khoảng 106 bộ phim. 75% người dùng đánh giá dưới 148 bộ phim

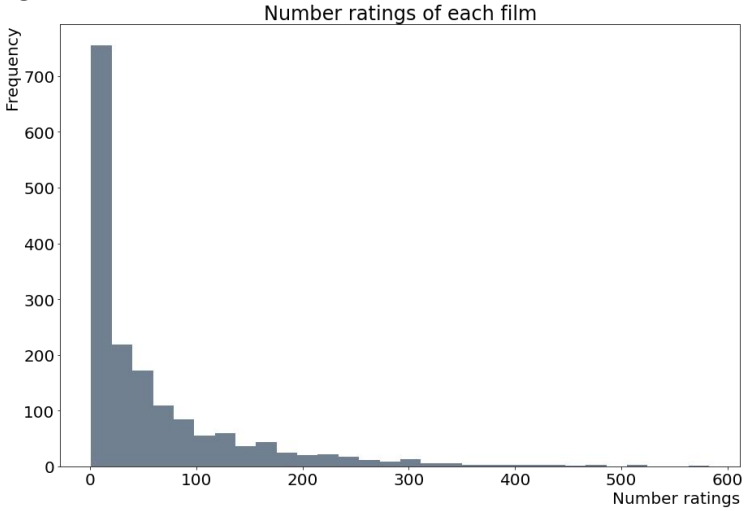



- Trung bình đánh giá của người dùng của 3.58 cho mỗi bộ phim

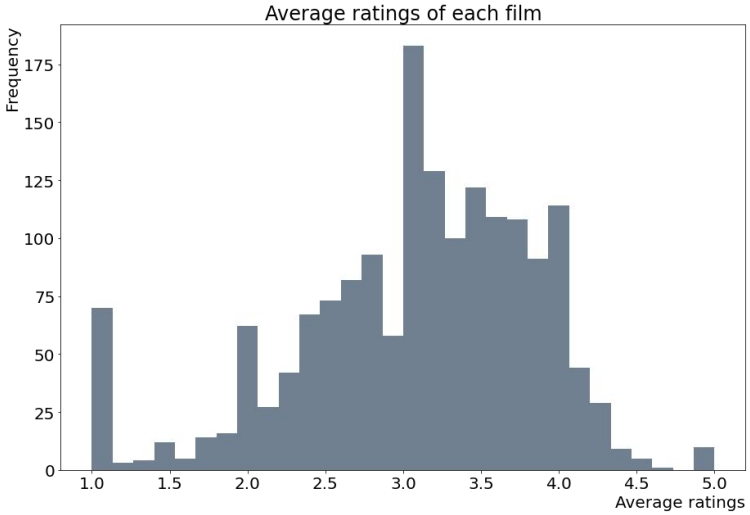



***- Về bộ phim:***

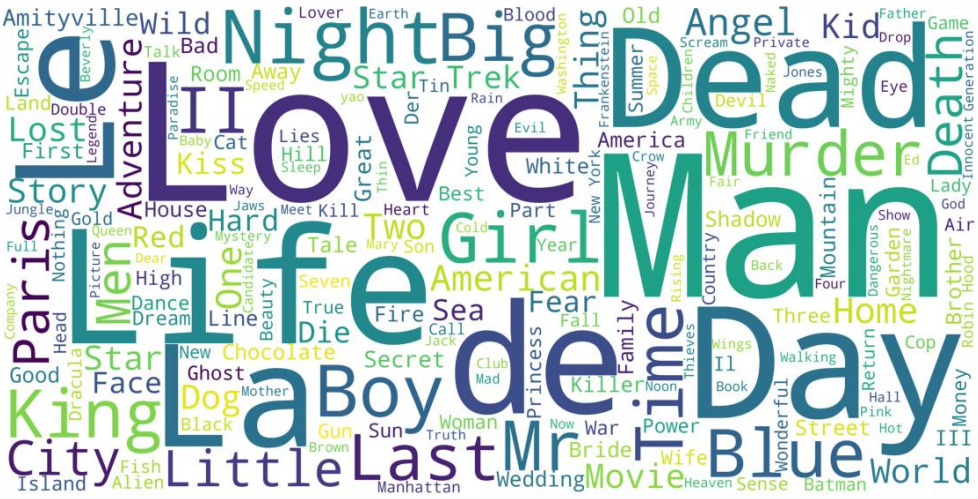
- Trung bình mỗi bộ phim được đánh giá 59 lần. Số đánh giá cho 1 bộ phim dao động từ 1 đến 583.



- Đánh giá trung bình cho 1 bộ phim là 3.07




- Tiêu đề của các bộ phim:




***Nhận xét: Bộ dữ liệu Movie Lens 100K đã khá tường minh, rõ ràng, dễ dàng truy xuất nhưng lại khá thiếu thông tin về các bộ phim so với IMDB 5000. Cần thu thập thêm một số dữ liệu khác để hoàn thiện hơn.***

# 3. Content-based Filtering

## 3.1. Áp dụng trên bộ dữ liệu IMDB 5000

### 3.1.1. Thư viện cần sử dụng

Một số thư viện được sử dụng như: *numpy, pandas, difflib, sklearn*

### 3.1.2. Tiền xử lí dữ liệu

Có thể thấy bộ dữ liệu trên bị phân tách thành 2 file riêng biệt, mỗi file có quá nhiều trường dữ liệu, hơn nữa dữ liệu có trong mỗi trường chưa thể sử dụng ngay được do chúng tồn tại ở nhiều dạng khác nhau. Vì vậy, trước khi áp dụng các thuật toán, ta phải tiền xử lý dữ liệu:

- Ghép 2 file csv: cả hai file dữ liệu mang thông tin của các bộ phim đều có chung trường "id", vì thế ta sẽ tiến hành ghép hai file này lại dựa trên trường dữ liệu chung này.

Sau đó, ta thu được một data frame với kích thước 4803x23 với 4803 là số bộ phim và 23 trường dữ liệu.

- Do các ô dữ liệu chỉ đang ở dạng string nên ta cần đưa chúng về dạng object tương ứng trong python ( ví dụ xâu "{"id":1}" sẽ được đưa về dạng dictionary với key = id và value = 1.

```
# Parse the stringified features into their corresponding python objects
# And selecting features
from ast import literal_eval

features = ['cast', 'crew', 'keywords', 'genres']
for feature in features:
    movie[feature] = movie[feature].apply(literal_eval)
```

Tiếp theo, từ trường *crew*, thực hiện trích xuất để lấy ra tên giám đốc sản xuất (Director) vì đây là công việc quan trọng nhất trong đoàn làm phim. Đối với trường *cast* ta trích xuất để lấy ra chỉ tên diễn viên (vì trường này còn chứa các thông tin khác như id, tên nhân vật trong phim).

- Tuy nhiên ta cần  thêm một bước nữa là chuyển các từ về dạng lowercase và bỏ dấu trắng đối với tên riêng của các diễn viên.

- Bước cuối cùng, gộp tất cả những thông tin của 4 trường *cast, director, keywords, genres* vào thành một xâu duy nhất (trường *metadata*). Điều này sẽ có ích cho việc ta sử dụng các thuật toán ở sau.

```
def merge_string(x):
    return ' '.join(x['keywords']) + ' '
         + ' '.join(x['cast']) + ' '
         + x['director'] + ' '
         + ' '.join(x['genres'])

movie['metadata'] = movie.apply(merge_string, axis=1)
s = movie.apply(merge_string, axis=1)

movie.head(2)
```

| tittle | cast | crew | director | metadata |
|---|---|---|---|---|
| Avatar | [samworthington, zoesaldana, sigourneyweaver, ... | [{'credit_id': '52fe48009251416c750aca23', 'de... | jamescameron | cultureclash future spacewar spacecolony socie... |
| Pirates of the Caribbean: At World's End | [johnnydepp, orlandobloom, keiraknightley, ste... | [{'credit_id': '52fe4232c3a36847f800b579', 'de... | goreverbinski | ocean drugabuse exoticisland eastindiatradingc... |

### 3.1.3 Thuật toán

Ta thực hiện mã hóa các xâu thu được ở bước tiền xử lý dữ liệu thành các vector. Hai thuật toán được sử dụng là CountVectorizer và TfidfVectorizer.

**a, CountVectorizer**

CountVectorizer là một công cụ  được cung cấp bởi thư viện scikit-learn bằng Python. Nó được sử dụng để chuyển một văn bản nhất định thành một vector trên cơ sở tần suất (số lượng) của mỗi từ xuất hiện trong toàn bộ văn bản. Điều này rất hữu ích khi có nhiều văn bản như vậy và muốn chuyển đổi từng từ trong mỗi văn bản thành vector để sử dụng trong phân tích văn bản sâu hơn.

Ví dụ ta có một mảng A gồm 2 xâu dữ liệu “this is the first document” và “this document is the second document”. Sau khi sử dụng CountVectorizer ta thu được hai vector tần suất như sau:

| | this | is | the | first | document | second |
|---|---|---|---|---|---|---|
| A[0] | 1 | 1 | 1 | 1 | 1 | 0 |
| A[1] | 1 | 1 | 1 | 0 | 2 | 1 |

**b, TfidfVectorizer**

TF-IDF (term frequency-inverse document frequency) là một phương pháp thống kê để tính toán sự quan trọng của các từ trong một văn bản. TF-IDF đặc

biệt hữu dụng trong các thuật toán Machine-learning sử dụng trong Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP).

- TF (term frequency): số lần một cụm từ xuất hiện trong một văn bản.

- IDF (inverse document frequency): độ đo mức độ phổ biến của một từ trên toàn bộ các văn bản. Lưu ý rằng nó là chung cho tất cả các văn bản, nên nếu một từ xuất hiện ở hầu hết các văn bản thì độ đo này sẽ tiệm cận về 1 (theo công thức dưới).

$$\text{idf}(t) = \log_e [\,(1+n) / (\,1 + \text{df}(t)\,)\,] + 1$$

Trong đó, t : từ đang xét

n : số lượng văn bản

df(t) : số lượng văn bản chứa từ t.

Khi đó ta tính được các vector ứng với từng văn bản bao gồm giá trị tại từ t là tf(t) * idf(t). Kết quả thu được cuối cùng là các vector trực chuẩn.

**c, Cosine similarity**

Sau khi mã hóa các xâu thành các vector theo hai cách trên, ta cần sử dụng một độ đo để tính toán sự liên quan giữa các vector với nhau. Ví dụ, hai vector có độ đo lớn thể hiện chúng có liên quan đến nhau nhiều hơn, và đây có thể là sự lựa chọn cho việc gợi ý phim.

Độ đo được sử dụng là Cosine similarity, được định nghĩa bằng cosine của góc tạo bởi 2 vector, do đó nó bằng tích vô hướng của 2 vector chia cho tích độ dài 2 vector.

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

Do các vector được mã hóa khi sử dụng CountVectorizer hay TfidfVectorizer đều có các thành phần lớn hơn 0 nên cosine similarity cho kết quả chạy từ 0 đến 1. Kết quả càng gần 1 càng cho thấy 2 vector gần nhau, và 2 phim tương ứng với 2 vector này sẽ có ích cho việc gợi ý.

***Cuối cùng để gợi ý cho một phim X, ta chỉ cần chọn các bộ phim có giá trị cosine similarity với phim X lớn nhất!***

## 3.2. Áp dụng trên bộ dữ liệu Movie Lens 100K

### 3.2.1. Thư viện sử dụng

Một số thư viện được sử dụng như: *numpy, pandas, requests, BeautifulSoup, sklearn, nltk*

### 3.2.2 Thu thập và tiền xử lí dữ liệu

Như đã nhận xét ở trên, bộ dữ liệu Movie lens 100K chứa khá ít thông tin về các bộ phim nên ta cần thu thập thêm

***a, Thu thập dữ liệu***

- Sử dụng 2 thư viện có sẵn của python là requests để lấy thông tin trang web và BeautifulSoup để lấy thông tin cần thiết.

- Do link thông tin của từng phim có sẵn trong dữ liệu đã không còn tồn tại nên chúng em đã thu thập dữ liệu trên trang https://www.imdb.com bằng 2 bước sau:
  - Chuyển từ tiêu đề sang query truy vấn phim bằng tên của trang web
  - Lấy phim từ trang truy vấn của trang web
  - Từ link phim lấy dữ liệu của phim bao gồm các trường: Tóm tắt bộ phim, Đạo diễn, Biên kịch, Diễn viên
  - Dữ liệu bị sai thông tin chiếm khoảng 15% tổng dữ liệu. Chúng em xử lí hoàn toàn bằng cách tự truy vấn và tìm kiếm thông tin và điền lại vào file link phim.

- Kết quả:
  - File links.csv là file bao gồm movie_id và link thu thập bước đầu chưa được điền lại
  - File full_links.csv là file bao gồm movie_id và link hoàn chỉnh đã điền điền lại bằng tay những giá trị sai.
  - File full_details.csv là file bao gồm movie_id, movie_brief, director, writers, casts đầy đủ của từng bộ phim. Có 1 vài bộ phim do không còn dữ liệu thì sẽ để là Unknown.

```
In [73]: get_full_details()
```

Out[73]:

| | movie_id | movie_brief | director | writers | casts |
|---|---|---|---|---|---|
| **0** | 1 | A cowboy doll is profoundly threatened and jea... | John Lasseter | [John Lasseter, Pete Docter, Andrew Stanton] | [Tom Hanks, Tim Allen, Don Rickles] |
| **1** | 2 | Years after a friend and fellow 00 agent is ki... | Martin Campbell | [Ian Fleming, Michael France, Jeffrey Caine] | [Pierce Brosnan, Sean Bean, Izabella Scorupco] |
| **2** | 3 | Four interlocking tales that take place in a f... | Directors | [Allison Anders, Alexandre Rockwell, Robert Ro... | [Tim Roth, Antonio Banderas, Sammi Davis] |
| **3** | 4 | A mobster travels to Hollywood to collect a de... | Barry Sonnenfeld | [Elmore Leonard, Scott Frank] | [Gene Hackman, Rene Russo, Danny DeVito] |
| **4** | 5 | An agoraphobic psychologist and a female detec... | Jon Amiel | [Ann Biderman, David Madsen] | [Sigourney Weaver, Holly Hunter, Dermot Mulroney] |
| **...** | ... | ... | ... | ... | ... |
| **1677** | 1678 | A man goes for a walk through the countryside ... | Aleksandr Sokurov | [Yuriy Arabov] | [Aleksei Ananishnov, Gudrun Geyer] |
| **1678** | 1679 | A young woman attempts to end her criminal car... | Michael Radford | [Andrew Davies, Chloe King, Michael Radford] | [Asia Argento, Jared Harris, Rupert Everett] |
| **1679** | 1680 | A London woman's love life and career both hin... | Peter Howitt | [Peter Howitt] | [Gwyneth Paltrow, John Hannah, John Lynch] |
| **1680** | 1681 | Stand up comedy by Martin Lawrence, filmed in ... | Thomas Schlamme | [Martin Lawrence] | [Martin Lawrence] |

- Hai trường writers và casts đã được chuyển thành list các string

### *b, Tiền xử lí dữ liệu*

- Đưa dữ liệu 2 trường casts và writers trong full_details về list các string. Gộp dữ liệu về bộ phim mà bộ dữ liệu đã có với full_details đã thu thập được theo trường movie_id

- Chuyển thể loại của phim thành 1 trường genres duy nhất có giá trị là list các genres

- Áp dụng tiền xử lí lên các trường có giá trị là list hoặc string (trừ trường movie_brief):
  - Nếu là string: Xóa khoảng trắng và đưa về dạng không viết hoa
  - Nếu là list: Áp dụng tiền xử lí với mỗi string trong list

- Áp dụng tiền xử lí lên trường movie_brief
  - Tokenization: Tách chuỗi đã cho thành từng thành phần riêng lẻ như các từ, các dấu câu.
  - Remove Stop word: Từ dừng là những từ xuất hiện nhiều trong văn bản mà không thể hiện nhiều ý nghĩa chung của văn bản. Nếu để lại thì có thể gây khó khăn trong việc xác định 2 văn bản tương tự nhau do từ dừng xuất hiện trùng lặp giữa 2 văn bản xuất hiện rất nhiều.

- Remove Stemming: Stemming là hiện tượng 1 từ nhưng có nhiều cách viết khác nhau, ví dụ như ở dạng số nhiều và số ít, dạng danh từ với động từ, hiện tại với quá khứ, …. Việc loại bỏ hiện tượng Stemming giúp ta giảm được tổng số lượng các từ trên toàn bộ các văn bản.

- Gộp tất cả thành phần trong các trường movie_brief, genres, director, writers, casts thành 1 trường metadata ngăn cách nhau bởi dấu " " giúp cho chúng ta tiện xử lí.

```
In [7]: movies_details
```

Out[7]:

| | movie_id | title | genres | movie_brief | director | writers | casts | metadata |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Toy Story (1995) | [animation, children, comedy] | [cowboy, doll, profoundli, threaten, jealou, n... | johnlasseter | [johnlasseter, petedocter, andrewstanton] | [tomhanks, timallen, donrickles] | animation children comedy cowboy doll profound... |
| **1** | 2 | GoldenEye (1995) | [action, adventure, thriller] | [year, friend, fellow, 00, agent, kill, joint,... | martincampbell | [ianfleming, michaelfrance, jeffreycaine] | [piercebrosnan, seanbean, izabellascorupco] | action adventure thriller year friend fellow 0... |
| **2** | 3 | Four Rooms (1995) | [thriller] | [four, interlock, tale, take, place, fade, hot... | directors | [allisonanders, alexandrerockwell, robertrodri... | [timroth, antoniobanderas, sammidavis] | thriller four interlock tale take place fade h... |
| **3** | 4 | Get Shorty (1995) | [action, comedy, drama] | [mobster, travel, hollywood, collect, debt, di... | barrysonnenfeld | [elmoreleonard, scottfrank] | [genehackman, renerusso, dannydevito] | action comedy drama mobster travel hollywood c... |
| **4** | 5 | Copycat (1995) | [crime, drama, thriller] | [agoraphob, psychologist, femal, detect, must,... | jonamiel | [annbiderman, davidmadsen] | [sigourneyweaver, hollyhunter, dermotmulroney] | crime drama thriller agoraphob psychologist fe... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **1677** | 1678 | Mat' i syn (1997) | [drama] | [man, goe, walk, countrysid, die, mother] | aleksandrsokurov | [yuriyarabov] | [alekseiananishnov, gudrungeyer] | drama man goe walk countrysid die mother aleks... |
| **1678** | 1679 | B. Monkey (1998) | [romance, thriller] | [young, woman, attempt, end, crimin, career, n... | michaelradford | [andrewdavies, chloeking, michaelradford] | [asiaargento, jaredharris, ruperteverett] | romance thriller young woman attempt end crimi... |
| **1679** | 1680 | Sliding Doors (1998) | [drama, romance] | [london, woman, 's, love, life, career, hing, ... | peterhowitt | [peterhowitt] | [gwynethpaltrow, johnhannah, johnlynch] | drama romance london woman 's love life career... |
| **1680** | 1681 | You So Crazy (1994) | [comedy] | [stand, comedi, martin, lawrenc, film, majest,... | thomasschlamme | [martinlawrence] | [martinlawrence] | comedy stand comedi martin lawrenc film majest... |
| **1681** | 1682 | Scream of Stone (Schrei aus Stein) | [drama] | [two, famou, competit, climber, | wernerherzog | [hans-ulrichklenner, waltersaxer, | [vittoriomezzogiorno, stefanglowacz, | drama two famou competit climber |

## 3.2.3. Thuật toán

### a. Sử dụng Vectorizer

Sau khi thu thập và tiền xử lí dữ liệu ta áp dụng kĩ thuật giống như với IMDB 5000 để tìm ra *bộ phim có giá trị cosine similarity với phim X lớn nhất!*

### b. Sử dụng Hồi qui Ridge

Dựa vào việc chúng ta có dữ liệu về các bộ phim và đánh giá của 1 người về các một số bộ phim đó, ta hoàn thành có thể đưa về bài toán Hồi quy dự đoán đánh giá của người đó trên những bộ phim mới dựa trên những đánh giá đã có.

*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:

$$y_{mn} = \mathbf{x}_m \mathbf{w}_n + b_n$$

(Chú ý rằng x_m là một vector hàng, w_n là một vector cột.)

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 Ridge Regression như sau:

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

Trong đó, thành phần thứ hai là regularization term và $\lambda$ là một tham số dương. Chú ý rằng regularization thường không được áp dụng lên bias b_n. 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 rate. 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.

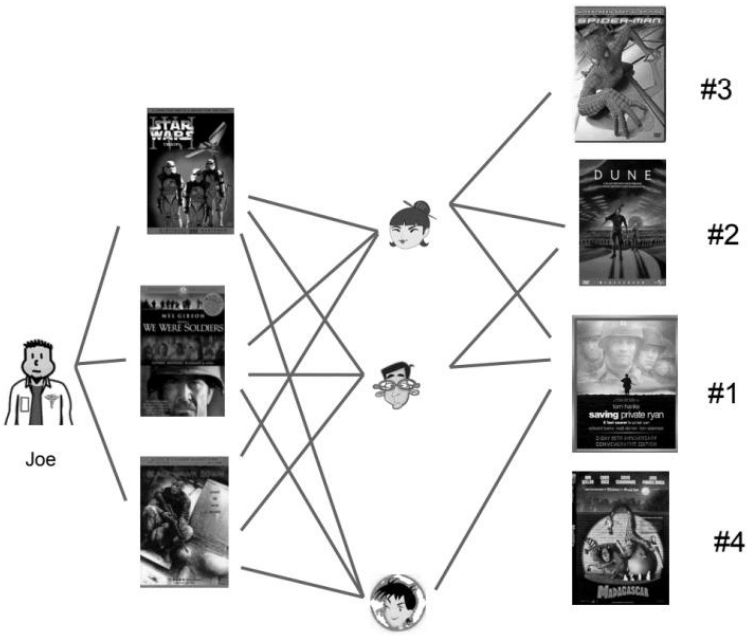
Do đa số trung bình 1 người chỉ đánh giá khoảng 100 bộ phim nên feature_vector không nên có độ dài quá lớn.

Nhận thấy thực tế, mọi người xem phim thường quyết định xem theo thể loại phim nên ta áp dụng TF-IDF trên trường genres để tạo ra features vector có độ dài 21 vừa phải.

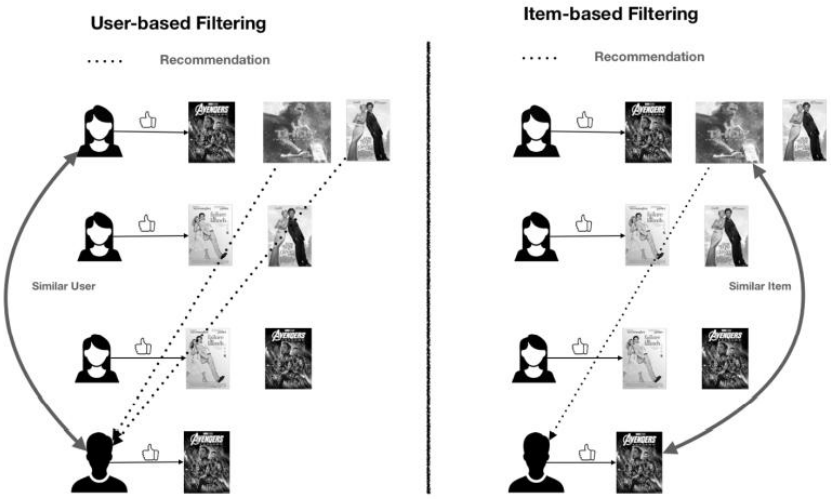
# 4. Neighborhood-Based Collaborative Filtering

## 4.1 Mô tả chung

Thuật toán Lọc cộng tác hàng xóm (the neighborhood-based CF algorithm) là một thuật toán dựa trên bộ nhớ phổ biến. Bằng cách xác định hàng xóm của người dùng (những người dùng đã đánh giá sản phẩm tương tự) và bỏ qua những người dùng khác, thuật toán sẽ đưa ra dự đoán về đánh giá của người dùng đối với sản phẩm. Ý tưởng cơ bản của NBCF là xác định mức độ quan tâm của một user u tới một item i dựa trên các users khác gần giống với user u hoặc cá items khác gần giống item i.




Thuật toán Lọc cộng tác hàng xóm (the neighborhood-based CF algorithm) là một thuật toán dựa trên bộ nhớ phổ biến. Bằng cách xác định hàng xóm của người dùng (những người dùng đã đánh giá sản phẩm tương tự) và bỏ qua những người dùng khác, thuật toán sẽ đưa ra dự đoán về đánh giá của người dùng đối với sản phẩm. Ý tưởng cơ bản của NBCF là xác định mức độ quan tâm của một user u tới một item i dựa trên các users khác gần giống với user u hoặc cá items khác gần giống item I




## 4.2. Chuẩn hóa dữ liệu

- Ứng với mỗi người dùng:

- Ta tính trung bình đánh giá của người đó trên những phim đã được người đó đánh giá
- Với mỗi đánh giá ta trừ đi 1 lượng bằng trung bình đánh giá phía trên
- Với những phim chưa đánh giá ta điền giá trị bằng 0

- Lợi ích:

- 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 *user thích item*, những giá trị âm tương ứng với việc *user không thích item*. Những giá trị bằng 0 tương ứng với việc *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*, nếu lưu toàn bộ các giá trị này trong một ma trận thì khả năng cao là sẽ không đủ bộ nhớ. Quan sát thấy rằng vì số lượng *ratings* biết trước thường là một số rất nhỏ so với kích thước của utility matrix, sẽ tốt hơn nếu chúng ta lưu ma trận này dưới dạng *sparse matrix*, tức chỉ lưu các giá trị khác không và vị trí của chúng.

## 4.3 Thuật toán

Kỹ thuật KNN dựa trên người dùng (user_KNN) xác định độ tương tự giữa hai người dùng thông qua việc so sánh các đánh giá của họ trên cùng sản phẩm, sau đó dự đoán xếp hạng trên sản phẩm i bởi người dùng u, thông qua các xếp

hạng của những người dùng tương tự với người dùng u. Độ tương tự giữa người dùng u và người dùng u' có thể được tính theo cosine similarity:

$$sim_{cosine}(u, u') = \frac{\sum_{i \in I_{uu'}} r_{ui} \cdot r_{u'i}}{\sqrt{\sum_{i \in I_{uu'}} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_{uu'}} r_{u'i}^2}}$$

Trong đó:

- $r_{ui}$ và $r_{u'i}$ là đánh giá của người dùng u và u' trên item i tương ứng
- $I_{uu'}$ là tập các item được đánh giá bởi cả người dùng u và người dùng u'

Tương tự, kỹ thuật KNN dựa trên sản phẩm (Item_KNN) cũng xác định độ tương tự dựa trên các item bằng cosine similatiry như sau:

$$sim_{cosine}(i, i') = \frac{\sum_{u \in I_{ii'}} r_{ui} \cdot r_{ui'}}{\sqrt{\sum_{u \in I_{ii'}} r_{ui}^2} \cdot \sqrt{\sum_{u \in I_{ii'}} r_{ui'}^2}}$$

Trong đó:

- $r_{ui}$ và $r_{ui'}$ là đánh giá của người dùng u trên item i và item i' tương ứng
- $I_{ii'}$ là tập các user đã đánh giá cả sản phẩm i và sản phẩm i'

Sau khi tính toán độ tương tự giữa các người dùng hay giữa các sản phẩm, đánh giá của người dùng u trên sản phẩm i được dự đoán theo các công thức như bên dưới:

Với phương pháp user_KNN, xếp hạng (đánh giá) của người dùng u trên sản phẩm i được dự đoán qua công thức:

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{u' \in K_u} \text{sim}(u, u') \cdot (r_{u'i} - \bar{r}_{u'})}{\sum_{u' \in K_u} |\text{sim}(u, u')|}$$

Trong đó:

- $\bar{r}_u$ là giá trị đánh giá trung bình trên tất cả các item của người dùng u
- $\bar{r}_{u'}$ là giá trị đánh giá trung bình trên tất cả các item của người dùng u'
- Sim(u,u') là độ tương tự giữa người dùng u và u' được xác định bằng phương pháp Cosine như đã trình bày
- $K_u$ là số người dùng có độ lân cận gần người dùng u (k láng giềng của u)

Với phương pháp item_KNN, xếp hạng (đánh giá) của người dùng u trên sản phẩm i được dự đoán qua công thức:

$$\hat{r}_{ui} = \bar{r}_i + \frac{\sum_{i' \in K_i} \text{sim}(i, i') \cdot (r_{ui'} - \bar{r}_{i'})}{\sum_{i' \in K_i} |\text{sim}(i, i')|}$$

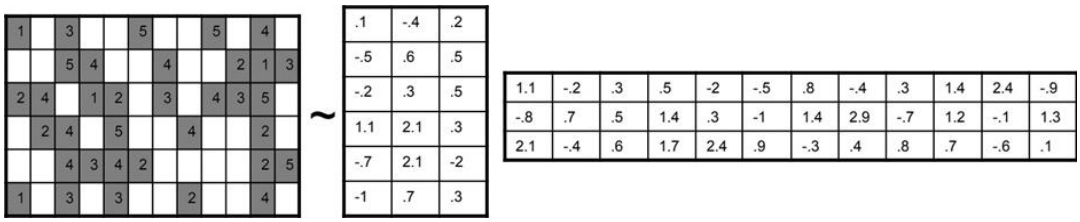
Trong đó:

- $\overline{r}_i$ là giá trị đánh giá trung bình của tất cả các người dùng trên item i
- $\overline{r}_{i'}$ là giá trị đánh giá trung bình của tất cả các người dùng trên item i'
- Sim(i,i') là độ tương tự giữa sản phẩm i và i' được xác định bằng phương pháp cosine như đã trình bày
- $K_i$ là số sản phẩm có độ lân cận gần sản phẩm i (k láng giềng của i)

# 5. Matrix Factorization Collaborative Filtering

## 5.1. Mô tả chung

Các thuật toán dựa trên Matrix Factorization (hoặc Matrix Decomposition), tức Phân tích ma trận thành nhân tử đã được chứng minh là có hiệu quả để giải quyết các thách thức về khả năng mở rộng và độ thưa thớt của dữ liệu (Nội dung 4). CF dựa trên Matrix Factorization hướng đến hai mục tiêu. Mục tiêu đầu tiên là giảm kích thước của ma trận xếp hạng. Mục tiêu thứ hai là đánh giá dựa trên các yếu tố tiềm ẩn của ma trận xếp hạng.




Bài toán đặt ra là cố gắng xấp xỉ ma trận đánh giá (ratings matrix) bằng tích của hai ma trận và trong đó P, Q lần lượt là ma trận thông tin sản phẩm (item profiles), mỗi hàng tương ứng với một item và ma trận thông tin người dùng (user profiles), mỗi cột tương ứng với một user; m, n lần lượt là số items và users, k là số thuộc tính, thường được chọn nhỏ hơn rất nhiều so với m, n. Khi đó, cả hai ma trận P, Q đều có rank không quá k.

## 5.2 Thuật toán

Ở đây, ta sẽ sử dụng phương pháp Stochastic gradient descent (SGD).

Như đã phân tích ở trên, phương pháp ứng dụng Matrix factorization trong hệ gợi ý là phương pháp tìm 2 ma trận đặc trưng $P \in R^{kxm}$ và $Q \in R^{kxn}$ sao cho $r_{\text{ui}} \approx p_u^T q_i$ với $r_{\text{ui}}$ là đánh giá của người dùng u với sản phẩm i, k là số đặc

trưng tiềm ẩn, m và n là số lượng users và items trong hệ thống tương ứng, $p_u \in R^k$ và $q_i \in R^k$ tương ứng là cột thứ u của P và cột thứ i của Q.

Nhiệm vụ đặt ra là tối ưu giá trị của hàm mất mát RMSE:

$$\min_{P,Q} \sum_{(u,i)\in R} \frac{1}{s}(r_{ui} - p_u^T q_i)^2 + \lambda_p \|P\|^2 + \lambda_Q \|Q\|^2$$

Trong đó:

- ||.|| là chuẩn Eulide
- $(u, i) \in R$ là các chỉ số của đánh giá đã có của người dùng
- $\lambda_p$, $\lambda_Q$ là các tham số để tránh overfitting.

Ý tưởng của SGD là lựa chọn ngẫu nhiên $r_{ui}$ từ ma trận đánh giá (ratings matrix) $R^{m\times n}$ với u, i là các chỉ số. $p_u$ và $q_i$ được tính như sau:

$$p_u \leftarrow p_u + \gamma(e_{ui} q_i - \lambda_P p_u)\ ,$$

$$q_i \leftarrow q_i + \gamma\left(e_{ui} p_u - \lambda_Q q_i\right)$$

Trong đó:

- $e_{ui} = r_{ui} - p_u^T q_i$ là sai số giữa xếp hạng thực tế và xếp hạng dự đoán cho $r_{ui}$
- $\lambda_P, \lambda_Q$ là các hệ số chính quy để tránh overfitting
- $\gamma$ là tốc độ học

Nhằm mục đích cá nhân hóa người dùng, ta có thể cộng thêm các giá trị thiên lệch (biases), đánh giá của người dùng u dành cho sản phẩm i. Khi đó giá trị dự đoán cho đánh giá được tính bằng:

$$r_{ui} \approx \mu + b_u + b_i + p_u^T q_i \quad (1)$$

Trong đó:

- $\mu$ là giá trị trung bình toàn cục, là giá trị xếp hạng trung bình của tất cả người dùng trên tất cả sản phẩm với tập dữ liệu huấn luyện
- $b_u$ là độ lệch người dùng (là giá trị lệch trung bình của các người dùng so với giá trị trung bình toàn cục)
- $b_v$ là độ lệch của sản phẩm (là giá trị lệch trung bình của các sản phẩm so với giá trị trung bình toàn cục)

Bài toán trở thành:

$$\min_{P,Q}\Big(\sum_{(u,v)\in R}\frac{1}{s}(r_{uv}-\mu-b_u-b_i-p_u^T q_i)^2\Big) + \lambda_P\big((\|P\|^2+\|b\_u\|^2)\big)+\lambda_Q\big((\|Q\|^2+\|b\_i\|^2)\big)$$

# 6. Kết quả, nhận xét và hướng phát triển

## 6.1 Kết quả

### 6.1.1. Sử dụng Vectorizer

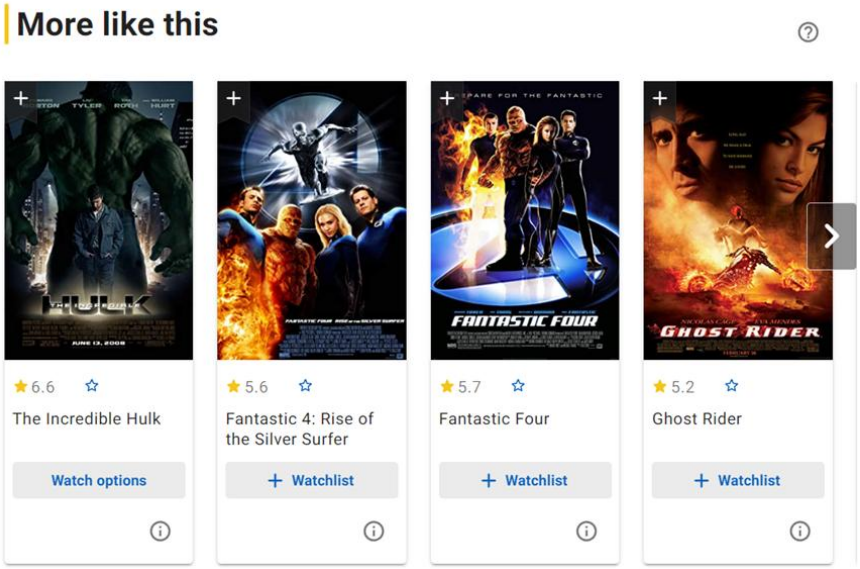
Kết quả áp dụng Content-based Filtering trên bộ dữ liệu TMDB 5000 với phương pháp đánh giá định tính:

Một số gợi ý thu được như sau: (lưu ý *close_match* là tên phim cần gợi ý - Hulk, được lấy từ input của người dùng, *cosine_sim1* và *cosine_sim2* lần lượt ứng với ma trận cosine similarity của CountVectorizer và TfidfVectorizer.

```
get_recommendations(close_match, cosine_sim1)
```

```
4401                        The Helix... Loaded
4638                   Amidst the Devil's Wings
215     Fantastic 4: Rise of the Silver Surfer
4734                                  Echo Dr.
511                                      X-Men
174                       The Incredible Hulk
4399                                  Special
929                                  Outbreak
4708                          Heroes of Dirt
1669                             The Promise
```

Thực nghiệm trên trang web IMDb: Ratings, Reviews, and Where to Watch the Best Movies & TV Shows với bộ phim Hulk, web tự động đề xuất các bộ phim khá giống với việc sử dụng các thuật toán ở trên.

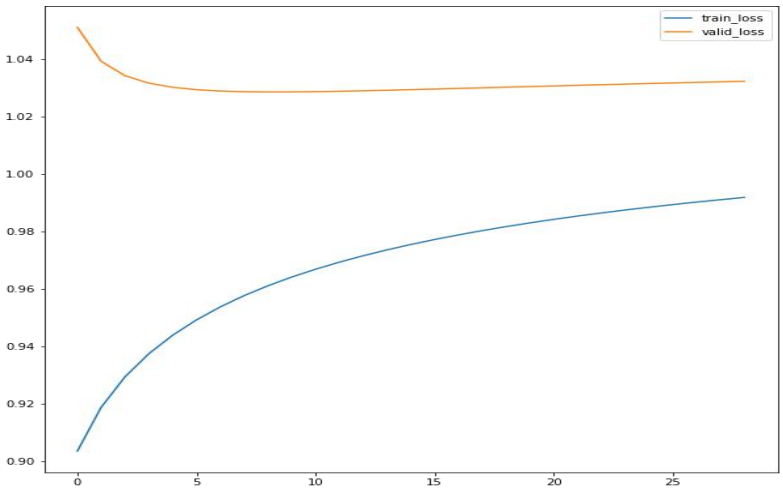



Thực nghiệm đối với một số bộ phim khác cũng cho kết quả các bộ phim được gợi ý trùng khá nhiều với việc sử dụng các thuật toán trên. Tuy nhiên, để đánh giá chính xác độ tốt của thuật toán, ta cần phải có dữ liệu của người dùng về các thao tác người dùng chọn xem các bộ phim được gợi ý. Loại dữ liệu này tương ứng với *TMDB 5000 Movie Dataset* nhóm chúng em vẫn chưa thu thập được, vì vậy chưa thể có đánh giá một cách chính xác nhất cho thuật toán mà chỉ dựa vào sự tương quan với gợi ý của website nói trên.

### 6.1.2. Áp dụng các mô hình trên tập dữ liệu MovieLens.

a. *Content-based filtering với Hồi qui Ridge trên bộ dữ liệu Movie Lens 100K*

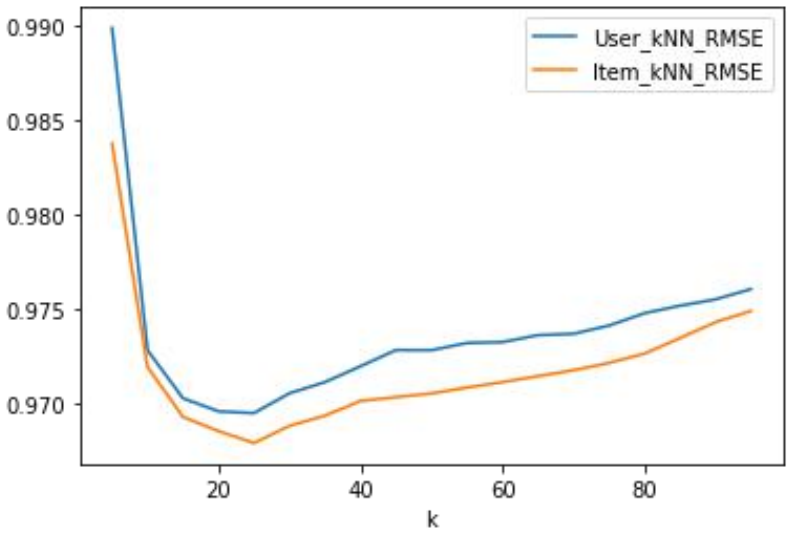
- Model selction:



- Qua model selection ta chọn hệ số lamda của hồi qui Ridge = 8 được kết quả như sau:
  - Train_loss: 0.9623232092593996
  - Test_loss: 1.0202035740552848

*b. Neighborhood-Based Collaborative Filtering với KNN*

- Model selection:




- Qua model selection chọn K = 25 được kết quả như sau:
  - UserKnn với test_loss: 0.9588561341420486
  - Item KNN với test_loss: 0.9539008698118429

*c. Matrix Factorization Collaborative Filtering với SGD*

| K | Alpha | Beta | Iterations | RMSE Train | | RMSE Valid | |
|---|---|---|---|---|---|---|---|
| | | | | Non bias | Bias | Non bias | Bias |
| 10 | 0.001 | 0.005 | 100 | 0.8591 | 0.873 | 0.9616 | 0.9488 |
| 20 | 0.001 | 0.005 | 100 | 0.8603 | 0.8824 | 0.9549 | 0.9438 |
| 50 | 0.001 | 0.005 | 100 | 0.8824 | 0.8989 | 0.9571 | 0.9454 |
| 20 | 0.002 | 0.005 | 100 | 0.6973 | 0.7181 | 0.963 | 0.9519 |
| 20 | 0.005 | 0.005 | 100 | 0.5152 | 0.5073 | 1.0867 | 1.0917 |
| 20 | 0.01 | 0.005 | 100 | 0.4619 | 0.4443 | 1.1765 | 1.1735 |
| 20 | 0.001 | 0.002 | 100 | 0.8569 | 0.8796 | 0.9553 | 0.9441 |
| 20 | 0.001 | 0.01 | 100 | 0.8664 | 0.8866 | 0.9564 | 0.9456 |
| 20 | 0.001 | 0.02 | 100 | 0.8758 | 0.8928 | 0.959 | 0.9447 |
| 20 | 0.001 | 0.005 | 150 | 0.7718 | 0.8027 | 0.9464 | 0.9382 |
| 20 | 0.001 | 0.005 | 200 | 0.6948 | 0.7185 | 0.9627 | 0.9519 |
| | | | | | | | |
| Chọn k = 20, alpha = 0.001, beta = 0.005, iterations = 150, bias = 1<br>Áp dụng trên toàn bộ tập train ta có: | | | | | RMSE_train | 0.7846 | |
| | | | | | RMSE_test | 0.924 | |

Qua model selection ta chọn k = 20, alpha = 0.01, beta = 0.005, iteration = 150, bias = 1 thu được kết quả sau:

- RMSE_train: 0.7846
- RMSE_test: 0.924

## 6.2 Nhận xét

- Ở bước đầu của các hệ thống gợi ý khi chưa có nhiều dữ liệu tương tác của người dùng, content-based filtering là phương pháp được ưu tiên lựa chọn vì tính đơn giản của nó. Chúng ta chỉ cần tìm phim có sự giống nhất với phim mà người dùng đã chọn là được. Thực tế cũng đã chứng minh khi so sánh kết quả chạy và gợi ý trên trang imdb.com.

- Khi hệ thống gợi ý đã thu thập được nhiều hơn thông tin về tương tác của người dùng, lúc đó việc xây dựng các mô hình dựa trên dữ liệu đã cho sẽ đưa ra kết quả gợi ý tốt hơn. Bắt đầu từ việc xây dựng mô hình tuyến tính đơn giản với mỗi người dùng đến sử dụng KNN để tính toán kết quả dựa trên những người dùng có hành vi tương tự và cuối cùng là sử dụng Matrix Factorization Collaborative Filtering với SGD để đánh giá được các yếu tố tiềm ẩn trong ma trận xếp hạng. Sai số lỗi ở trong tập test ứng với từng mô hình theo thứ tự trên cũng giảm dần thể hiện sự ưu việt của các mô hình.
- Mô hình với KNN ưu việt hơn hồi qui Ridge ở việc đã xem xét và tận dụng được lợi thế người dùng thường có xu hướng tương tác giống nhau
- Mô hình MF ưu việc hơn ở việc đánh giá được các yếu tố tiềm ẩn của cả phim và người dùng.

- Dựa vào các thử nghiệm với phương pháp MF ta thấy rằng kết quả với bias thể hiện sự ưu hiện hơn non bias. Điều đó cho thấy rằng các đánh giá thường có xu hướng bị lệch do có sự thiên vị của người dùng.

- Các phương pháp áp dụng trong bài hiện chỉ đang áp dụng với bộ dữ liệu nhỏ. Khi áp dụng với bộ dữ liệu lớn ta cần áp dụng nhiều kĩ thuật khác phù hợp.

## 6.3 Hướng phát triển trong tương lai

- Tìm hiểu về phương pháp Lọc cộng tác kết hợp (Hybrid Collaborative Filtering) để tận dụng được nhiều hơn dữ liệu về các bộ phim.
- Tìm hiểu các phương pháp áp dụng được cho các bộ dữ liệu thưa thớt hoặc bộ dữ liệu lớn.
- Tìm hiểu các phương pháp xử lí ngôn ngữ và hình ảnh để thể hiện dữ liệu về các bộ phim chi tiết hơn. Ví dụ như loại bỏ từ đồng nghĩa, ….

# Lời kết

Bài tập lớn môn học giúp chúng em hiểu được thêm về hệ thống gợi ý - một công nghệ hiện hữu rất nhiều trong đời sống hằng ngày mà ít ai để ý tới. Thông qua việc tìm hiểu và áp dụng, chúng em hiểu hơn được về kiến thức lí thuyết trên lớp và cách áp dụng trên thực tế của nó. Các kĩ thuật trong đời sống hầu như đều xuất phát từ những điều cơ bản nhưng việc biết cách áp dụng nó  phù hợp vào bài toán thực tế mới là điều khó khăn nhất. Ngoài ra chúng em lần đầu tiên tự mình thu thập dữ liệu qua mạng, qua đó nhận thấy mặc dù dữ liệu rất nhiều nhưng để lấy được dữ liệu và dùng dữ liệu đó để dùng được phải qua một thời gian dài.

# Tài liệu tham khảo

TMDB 5000 Movie Dataset | Kaggle

MovieLens 100K Dataset | GroupLens

(483) Project 18. Movie Recommendation System using Machine Learning with Python - YouTube

Getting Started with a Movie Recommendation System | Kaggle

Machine Learning cơ bản (machinelearningcoban.com)

Cosine similarity - Wikipedia

Understanding TF-ID: A Simple Introduction (monkeylearn.com)

How sklearn’s Tfidfvectorizer Calculates tf-idf Values - Analytics Vidhya

Using CountVectorizer to Extracting Features from Text - GeeksforGeeks

Tokenize text using NLTK in python - GeeksforGeeks