TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN MÔN XỬ LÝ DỮ LIỆU LỚN**

**RECOMMENDER SYSTEMS WITH THE MOVIES DATASET**

*Người hướng dẫn*: **THẦY LÊ ANH CƯỜNG**

*Người thực hiện*: **THÂN TRỌNG HUỲNH NHÂN – 51800590**

**HUỲNH TẤN LỢI – 51800574**

**VÕ TẤN LỰC - 51800900**

Lớp **: 18050302**

Khoá  **: 22**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN MÔN XỬ LÝ DỮ LIỆU LỚN**

**RECOMMENDER SYSTEMS WITH THE MOVIES DATASET**

Người hướng dẫn: **THẦY LÊ ANH CƯỜNG**

Người thực hiện: **THÂN TRỌNG HUỲNH NHÂN**

**HUỲNH TẤN LỢI**

**VÕ TẤN LỰC**

Lớp **: 18050302**

Khoá  **: 22**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020**

LỜI CẢM ƠN

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

Nhóm em xin gửi lời cảm ơn chân thành đến thầy Lê Anh Cường, giảng viên Bộ môn Xử lý dữ liệu lớn trường ĐH Tôn Đức Thắng, người đã tận tình hướng dẫn, chỉ bảo nhóm em trong suốt quá trình làm tiểu luận cũng như trong việc giảng dạy.

Nhóm em cũng xin chân thành cảm ơn các thầy cô giáo trong trường ĐH Tôn Đức Thắng nói chung, các thầy cô trong khoa CNTT nói riêng đã dạy dỗ cho em kiến thức về các môn đại cương cũng như các môn chuyên ngành, giúp chúng em có được cơ sở lý thuyết vững vàng và tạo điều kiện giúp đỡ nhóm em trong suốt quá trình học tập.

Cuối cùng, nhóm em xin chân thành cảm ơn gia đình và bạn bè, đã luôn tạo điều kiện, quan tâm, giúp đỡ, động viên em trong suốt quá trình học tập và hoàn thành bài đồ án này.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của thầy Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong tiểu luận còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 25 tháng 04 năm 2021*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Thân Trọng Huỳnh Nhân*

*Huỳnh Tấn Lợi*

*Võ Tấn Lực*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

* 1. Lý do chọn đề tài

Trong thị trường kinh doanh, việc nắm bắt được thị hiếu, nhu cầu, xu hướng,… thông qua sở thích, cá tính, thói quen, sự quan tâm của khách hàng là một trong những sức mạnh to lớn của các nhà kinh doanh, đầu tư thường chú trọng để có thể sống sót hoặc thậm chí là dành lợi thế, chiến thắng trong cuộc đua thị trường đầy tính cạnh tranh và chông gai. Và với ngữ cảnh công nghệ, các ngành dịch vụ (bao gồm các lĩnh vực giải trí, ăn uống, … nhất là trong lĩnh vực phim ảnh) đang phát triển một cách ồ ạt như hiện nay cũng kéo theo việc cạnh tranh vốn đã vô cùng căng thẳng nay lại càng căng thẳng hơn.

Hệ thống kiến nghị (Recommender Systems) xuất hiện với vai trò là một cánh tay phải đắc lực cho những nhà kinh doanh trong việc thu thập các dữ liệu của khách hàng và từ những dữ liệu đó mà nhà kinh doanh có thể đưa ra những sản phẩm của mình phù hợp với khách hàng nhất. Chính nhờ có áp dụng Recommender Systems mà các nhà kinh doanh có thể giành cho mình nhiều lợi thế hơn trong thị trường gay gắt như hiện nay.

Chúng em quyết định chọn đề tài “Hệ thống khuyến nghị với movies dataset - Recommender Systems with Movies Dataset” để vừa có thể nghiên cứu, tìm hiểu, học hỏi và trau dồi khả năng chuyên môn, cũng nhữ hiểu biết thêm về quá trình phân tích và xử lý dữ liệu thu thập được của các rạp phim, các kênh chiếu phim online để có thể phân tích xem khách hàng có thể sẽ thích những bộ phim nào từ đó kiến nghị các bộ phim cho khách hàng. Ở đề tài này chúng em sẽ lựa chọn hình thức Content Based Filtering (CB) của Recommender Systems, Consine Similarity và TF-IDF Vectorizer (Term Frequency – Inverse Document Frequency). Từ đó củng cố thêm khả năng phân tích, xử lý dữ liệu và ra quyết định của bản thân phục vụ cho cuộc sống và công việc sau này.

* 1. Mục đích của tài liệu

Biết được cách thức phân tích, xử lý dữ liệu từ tập dữ liệu (dataset) cho trước từ đó áp dụng hệ thống khuyến nghị (ở đây là CB), kết hợp cùng TF-IDF Vectorizer và Consine Similarity để có thể biết được các bộ phim có khả năng được khách hàng xem nhất . Đồng thời, biết được quá trình xử lý dữ liệu của hệ thống khuyến nghị nói chung và hình thức CB của hệ thống khuyến nghị nói riêng cũng như của cả TF-IDF Vectorizer và Consine Similarity.

* 1. Các vấn đề nghiên cứu

Để tiếp cận cũng như thực hiện đề tài này, chúng em đã nghiên cứu và tìm hiểu ctheo nhiều khía cạnh của vấn đề, cụ thể như sau:

*Chương 1 – Giới thiệu:* Nhằm trang bị kiến thức cho người đọc một cách tổng quan nhất về hệ thống khuyến nghị (Recommender Systems), các hình thức, phương pháp của hệ thống khuyến nghị, …

*Chương 2 – Movies Meta Dataset* và các phương pháp: Nội dung chủ yếu ở chương này là giới thiệu về tập dữ liệu movies meta dataset, trực quan dữ liệu trong dataset bằng đồ thị, đánh giá khái quát về dữ liệu, cũng như giới thiệu về các thuật toán, phương pháp, mô hình sẽ sử dụng trong demo của đồ án.

*Chương 3 – Chương trình demo:* Chương này sẽ tập trung áp dụng các kiến thức, dữ liệu phân tích mà có được từ hai chương trước để xử lý tập tin có sẵn Movies meta dataset.

* 1. Quá trình thực hiện và kết quả nghiên cứu

Để giải quyết các vấn đề đặt ra trong đề tài này, nhóm đã đề ra lịch trình và từng bước thực hiện như sau:

* Lên kế hoạch họp vào mỗi thứ 3, 5, 7 hàng tuần
* Phân công tra cứu, thu nhập thông tin từ các nguồn trên internet
* Tiến hành tổng hợp các thông tin đã thu nhập được
* Tiến hành xác định nền tảng triển khai chương trình
* Hoàn thành chương trình
* Hoàn thành và chỉnh sửa để phù hợp với đề tài
* Kết quả của nhóm đã đạt được thồn qua đề tài:

CÁC THÀNH VIÊN ĐÃ HOÀN THÀNH TỐT NHIỆM VỤ ĐƯỢC GIAO

MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc70282770)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN 3](#_Toc70282771)

[TÓM TẮT 4](#_Toc70282772)

[1.1 Lý do chọn đề tài 4](#_Toc70282773)

[1.2 Mục đích của tài liệu 5](#_Toc70282774)

[1.3 Các vấn đề nghiên cứu 5](#_Toc70282775)

[1.4 Quá trình thực hiện và kết quả nghiên cứu 5](#_Toc70282776)

[MỤC LỤC 7](#_Toc70282777)

[DANH MỤC CHỮ VIẾT TẮT 9](#_Toc70282778)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 10](#_Toc70282779)

[BẢNG PHÂN CÔNG NHIỆM VỤ 11](#_Toc70282780)

[CHƯƠNG 1 – GIỚI THIỆU 13](#_Toc70282781)

[1.1 Đôi nét về Hệ thống khuyến nghị (Recommender Systems) 13](#_Toc70282782)

[1.1.1 Hiện tượng Long Tail trong thương mại điện tử 13](#_Toc70282783)

[1.1.2 Hệ thống khuyến nghị - Recommender Systerm 13](#_Toc70282784)

[1.2 Thành phần của một Recommender System và các phương pháp/mô hình Recommender System 14](#_Toc70282785)

[CHƯƠNG 2 – MOVIES META DATASET VÀ CÁC PHƯƠNG PHÁP 19](#_Toc70282786)

[2.1 Đôi nét về tập dữ liệu Movies Meta 19](#_Toc70282787)

[2.2 Trực quan hóa các trường dữ liệu trong movies\_metadata.csv 20](#_Toc70282788)

[2.3 Các phương pháp, thuật toán, mô hình được sử dụng 21](#_Toc70282789)

[2.3.1 Phương pháp xây dựng dựa theo nội dung (Content Based Filtering) 21](#_Toc70282790)

[2.3.2 Phương pháp TF-IDF vectorizer 23](#_Toc70282791)

[2.3.3 Consine similarity 25](#_Toc70282792)

[2.3.4 Demographic filtering 25](#_Toc70282793)

[CHƯƠNG 3 – CHƯƠNG TRÌNH DEMO 26](#_Toc70282794)

[3.1 Chương trình demo không áp dụng PySpark 26](#_Toc70282795)

[3.1.1 Quá trình chuẩn bị 26](#_Toc70282796)

[3.1.2 Quá trình Demographic Filtering và chọn ra weight rank 29](#_Toc70282797)

[3.1.3 TF-IDF Vectorizer 33](#_Toc70282798)

[3.1.4 Consine Similarity 34](#_Toc70282799)

[3.2 Chương trình demo có áp dụng PySpark 36](#_Toc70282800)

[3.2.1 Quá trình chuẩn bị 36](#_Toc70282801)

[3.2.2 TF-IDF Vectorizer 38](#_Toc70282802)

[3.2.3 Consine Similarity 39](#_Toc70282803)

[TÀI LIỆU THAM KHẢO 42](#_Toc70282804)

DANH MỤC CHỮ VIẾT TẮT

**CÁC CHỮ VIẾT TẮT**

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 1 – Utility matrix 15](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282805)

[Hình 2 – Các loại mô hình của hệ thống Recommender System 16](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282806)

[Hình 3 - Content Based Filtering 16](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282807)

[Hình 4 - Collaborative Filtering 17](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282808)

[Hình 5 – Hybrid Recommendations 18](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282809)

[Hình 6 – Movies meta dataset được tổng hợp ở trang kaggle.com 19](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282810)

[Hình 7 – Dữ liệu trong file movies\_metadata.csv 20](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282811)

[Hình 8 – Đồ thị biểu hiện tỷ trọng cũng như số lượng khách hàng xem phim ở độ tuổi trưởng thành 20](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282812)

[Hình 9 – Đồ thị thể hiện số số lượng vote trung bình của trường dữ liệu vote\_average 20](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282813)

[Hình 10 – Đồ thị thể hiện dữ liệu của trường dữ liệu release\_date của movies\_metadata.csv 21](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282814)

[Hình 11 – công thức xác định TF 24](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282815)

[Hình 12 – Công thức xác định IDF 24](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282816)

[Hình 13 – Công thức tính Consine Similarity 25](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282817)

[Hình 14 – Demographic Filtering 25](file:///C:\Users\MSI%20GV72\OneDrive\Máy%20tính\51800590_51800574_51800900_RecomendationSystem.docx#_Toc70282818)

**DANH MỤC BẢNG**

[Bảng 1 – Bảng phân công công việc 11](#_Toc70281176)

BẢNG PHÂN CÔNG NHIỆM VỤ

|  |  |  |
| --- | --- | --- |
| **Tên thành viên** | **Nhiệm vụ** | **Mức độ hoàn thành** |
| Thân Trọng Huỳnh Nhân | - Phụ trách mục 1.1 “Đôi nét về Hệ thống khuyến nghị (Recommender Systems)” | Hoàn thành tốt |
| - Phụ trách mục 1.2 “Thành phần của một Recommender System và các phương pháp / mô hình Recommender System” | Hoàn thành tốt |
| - Tìm hiểu tài liệu và phụ trách mục 2.3.3 “Consine similarity” | Hoàn thành |
| - Tìm kiếm tài liệu và phụ trách mục 2.3.1 “Phương pháp xây dựng dựa theo nội dung (Content Based Filtering)” | Hoàn thành |
| - Tổng hợp và hoàn tất báo cáo | Hoàn thành tốt |
| Huỳnh Tấn Lợi | - Tìm kiếm dataset | Hoàn thành |
| - Tìm kiếm tài liệu và phụ trách mục 2.3.1 “Phương pháp xây dựng dựa theo nội dung (Content Based Filtering)” | Hoàn thành tốt |
| - Tìm kiếm tài liệu và phụ trách mục 2.3.4 “Demographic filtering” | Hoàn thành tốt |
| - Phụ trách mục 1.2 “Thành phần của một Recommender System và các phương pháp / mô hình Recommender System” | Hoàn thành |
| Võ Tấn Lực | - Phụ trách mục 1.2 “Thành phần của một Recommender System và các phương pháp / mô hình Recommender System” | Hoàn thành tốt |
| - Tìm kiếm tài liệu và phụ trách mục 2.3.1 “Phương pháp xây dựng dựa theo nội dung (Content Based Filtering)” | Hoàn thành |
| - Tìm kiếm tài liệu và phụ trách mục 2.3.2 “Phương pháp TF-IDF vectorizer” | Hoàn thành |
| - Tìm kiếm tài liệu và phụ trách mục 2.3.4 “Demographic filtering” | Hoàn thành tốt |
| Các thành viên trong nhóm tham gia code | | Hoàn thành |

Bảng 1 – Bảng phân công công việc

CHƯƠNG 1 – GIỚI THIỆU

* 1. Đôi nét về Hệ thống khuyến nghị (Recommender Systems)
     1. Hiện tượng Long Tail trong thương mại điện tử

Nguyên tắc dễ thấy để đạt doanh thu cao là trưng ra các sản phẩm phổ biến nhất ở những nơi dễ nhìn thấy và những sản phẩm ít phổ biến hơn được cất trong kho. Cách làm này có một hạn chế rõ rệt: những sản phẩm được trưng ra mang tính phổ biến chứ chưa chắc đã phù hợp với một khách hàng cụ thể. Một cửa hàng có thể có món hàng một khách hàng tìm kiếm nhưng có thể không bán được vì khách hàng không nhìn thấy sản phẩm đó trên giá ⇨ việc này dẫn đến việc khách hàng không tiếp cận được sản phẩm ngay cả khi chúng đã được trưng ra.

Ngoài ra, vì không gian có hạn, cửa hàng không thể trưng ra tất cả các sản phẩm mà mỗi loại chỉ đưa ra một số lượng nhỏ. Ở đây, phần lớn doanh thu (80%) đến từ phần nhỏ số sản phẩm phổ biến nhất (20%). Nếu sắp xếp các sản phẩm của cửa hàng theo doanh số từ cao đến thấp, ta sẽ nhận thấy có thể phần nhỏ các sản phẩm tạo ra phần lớn doanh số; và một danh sách dài phía sau chỉ tạo ra một lượng nhỏ đóng góp. Hiện tượng này còn được gọi là long tail phenomenon, tức phần đuôi dài của những sản phẩm ít phổ biến.

* + 1. Hệ thống khuyến nghị - Recommender Systerm

Hệ thống gợi ý (Recommender systems hoặc Recommendation systems, Recommender platform, Recommender engine) là một dạng của hệ hỗ trợ ra quyết định, cung cấp giải pháp mang tính cá nhân hóa mà không phải trải qua quá trình tìm kiếm phức tạp. Hệ thống gợi ý học từ người dùng và gợi ý các sản phẩm tốt nhất trong số các sản phẩm phù hợp. Recommender System (RS) là một lớp con của filtering system với ý tưởng là dự đoán “xếp hạng” hay “sở thích” của người dùng.

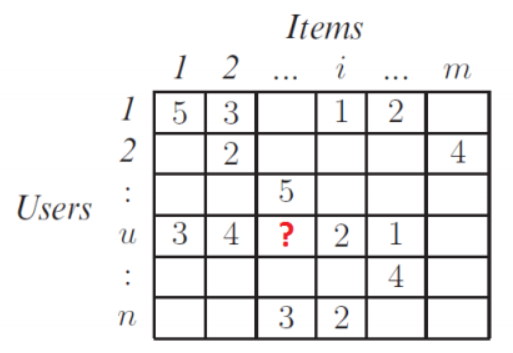
***Ví dụ*:**

* 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 gợi ý kết bạn
* Facebook hiển thị quảng cáo những sản phẩm có liên quan đến từ khoá bạn vừa tìm kiếm.

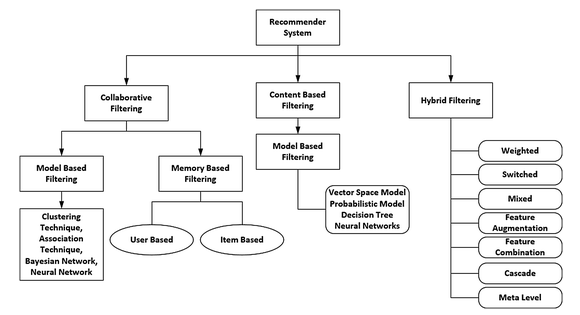
Vì tính hữu ích và được mệnh danh là cánh tay phải đắc lực trong vấn đề kinh doanh (đặc biệt là thương mại online ngày càng phát triển hiện nay) nên các nhà thương mại điện tử lớn như Amazon hoặc Netflix, Facebook. Cụ thể, đối với Amazon:

* Dựa vào dữ liệu quá khứ của khách hàng như dữ liệu về điểm mà họ đã đánh giá trên từng sản phẩm, thời gian duyệt trên từng sản phẩm, số lần click vào sản phẩm,…để có thể nắm bắt được việc những mặt hàng nào mà khách hàng của họ yêu thích.
* Từ đó, Amazon có thể dự đoán được khách hàng có thể sẽ thích những sản phẩm nào khác và đưa ra gợi ý thích hợp.
  1. Thành phần của một Recommender System và các phương pháp/mô hình Recommender System

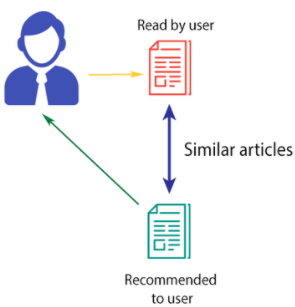
Hệ thống gợi ý (Recommender System) được tiếp cận theo phương pháp Machine Learning thường có ba thành phần sau:

* *Dữ liệu (thường là các dữ liệu về user, items, feedback)*: mà hệ thống quan tâm để lấy các dữ liệu xác định sở thích của đối tượng phục vụ cho việc dự đoán và đưa ra gợi ý. Trong đó:
  + users là danh sách người dùng
  + items là danh sách sản phẩm, đối tượng của hệ thống. Ví dụ như các bài viết trên trang viblo, các video trên youtube,… Và mỗi item có thể kèm theo thông tin mô tả.
  + feedback là lịch sử tương tác của user với mỗi item, có thể là đánh giá của mỗi user với một item, số ratings, hoặc comment, việc user click, view hoặc mua sản phẩm,…
  + *Ma trận user-item (Utility matrix)*: là ma trận biểu diễn mức độ quan tâm (rating) của user với mỗi item. Ma trận này được xây dựng từ dữ liệu. Những ma trận này có rất nhiều các giá trị miss. Nhiệm vụ của Hệ gợi ý chính là dựa vào các ô đã có giá trị trong ma trận trên (dữ liệu thu được từ trong quá khứ), thông qua mô hình đã được xây dựng, dự đoán các ô còn trống (của user hiện hành), sau đó sắp xếp kết quả dự đoán (ví dụ, từ cao xuống thấp) và chọn ra Top-N items theo thứ tự rating giảm dần, từ đó gợi ý chúng cho người dùng.

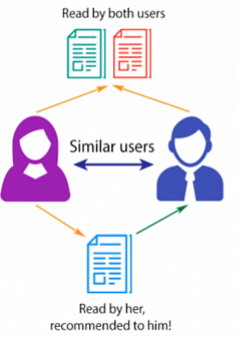
Hình – Utility matrix

* *Phương pháp / mô hình:* một hệ thống khuyên nghị thường sẽ được xây dựng dụa trên một trong ba phương pháp / mô hình sau:

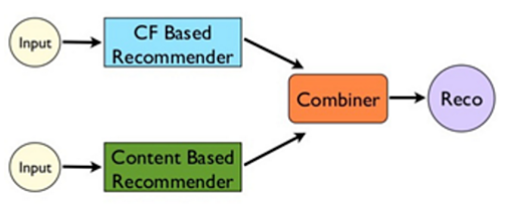
Hình – Các loại mô hình của hệ thống Recommender System

* + Content Based Filtering (CB): là phương pháp phổ biến 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.

Hình - Content Based Filtering

* + Collaborative Filtering (CF): 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 đổi 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.

Hình - Collaborative Filtering

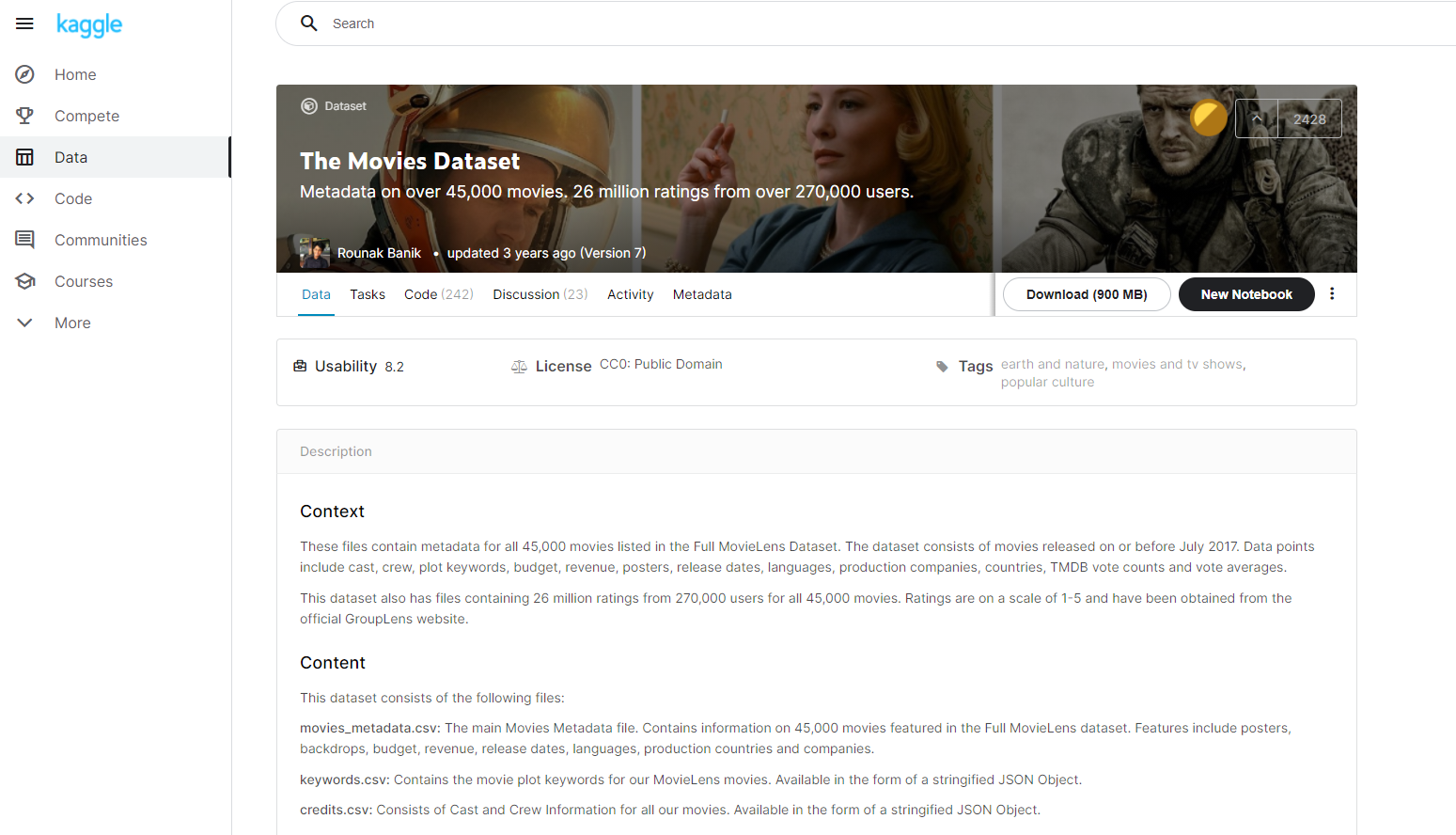
* + Hybrid Methods: Ngoài ra chúng ta cũng có thể sử dụng kết hợp cả 2 phương pháp trên để tạo thành một thuật toán kết hợp. Ưu điểm của phương pháp này đó là vừa tận dụng được các thông tin từ phía sản phẩm và các thông tin về hành vi mua sắm của người dùng.

Hình – Hybrid Recommendations

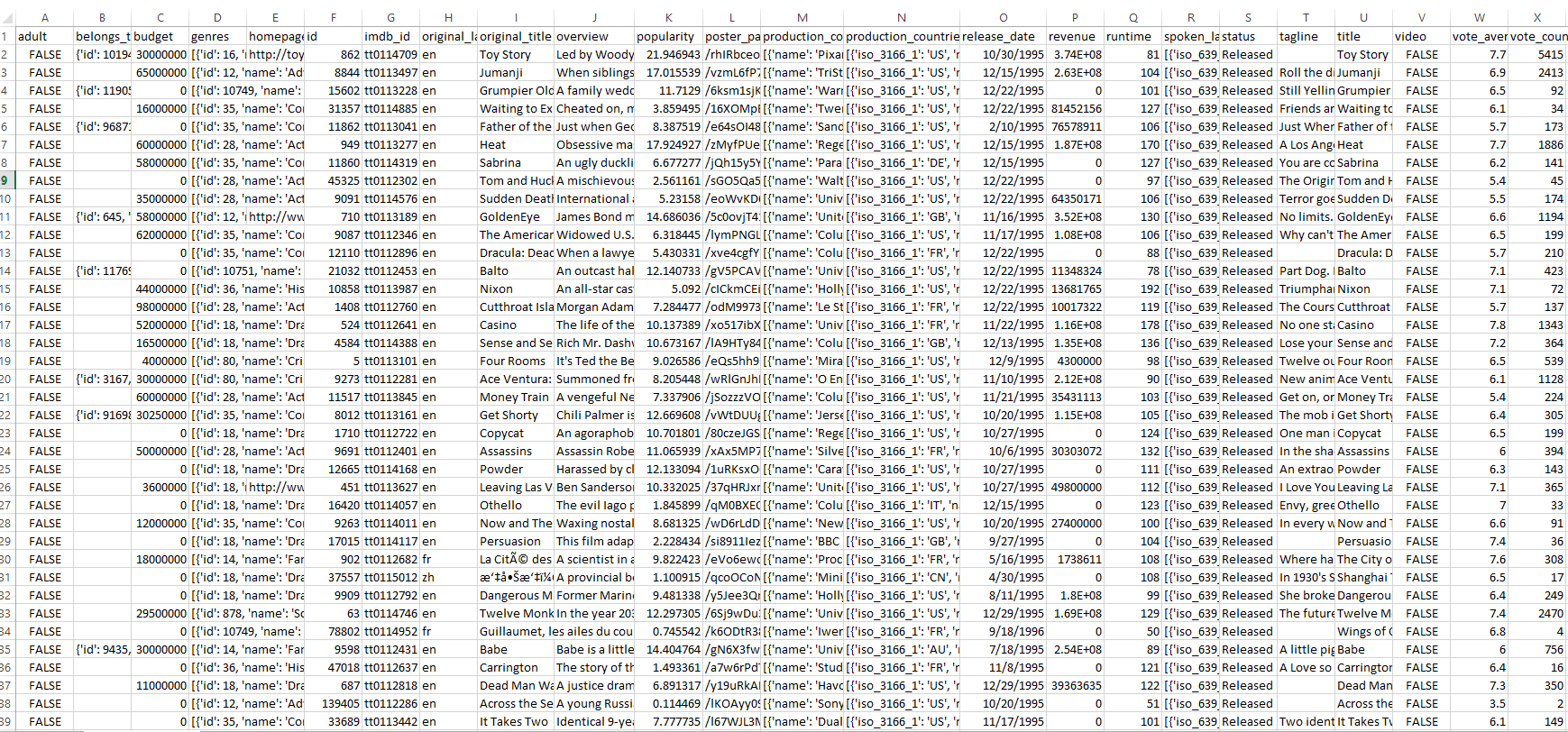
CHƯƠNG 2 – MOVIES META DATASET VÀ CÁC PHƯƠNG PHÁP

* 1. Đôi nét về tập dữ liệu Movies Meta

Tập dữ liệu movies meta được tập hợp như một phần của Dự án Capstone thứ hai, bao gồm các phim được phát hành vào hoặc trước tháng 7 năm 2017. Các điểm dữ liệu bao gồm  dàn diễn viên, đoàn phim, từ khóa cốt truyện, ngân sách, doanh thu, áp phích, ngày phát hành, ngôn ngữ, công ty sản xuất, quốc gia, số phiếu bầu của TMDB và số phiếu trung bình.

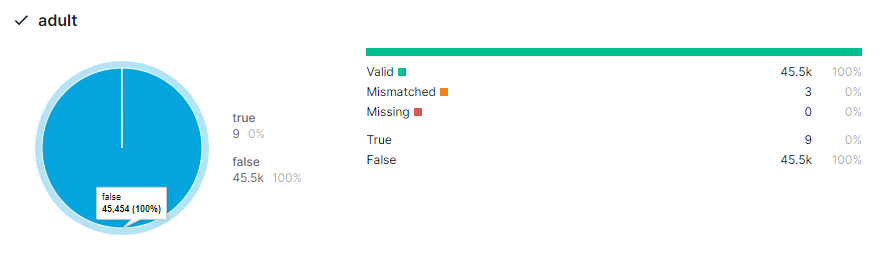
Tập dữ liệu chứa thông tin của 45.000 bộ phim có trong bộ dữ liệu Full MovieLens, bao gôm các trường là “*posters, backdrops, budget, revenue, release dates, languages, production countries and companies.*”. Movies Meta Dataset được kaggle.com tổng hợp lại từ việc thu thập dữ liệu từ TMDB và GroupLens

Hình – Movies meta dataset được tổng hợp ở trang kaggle.com

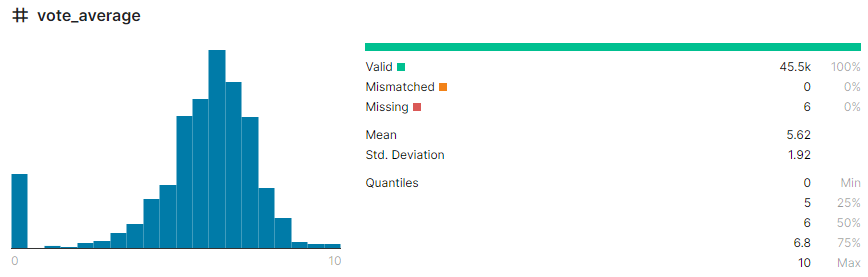


Hình – Dữ liệu trong file movies\_metadata.csv

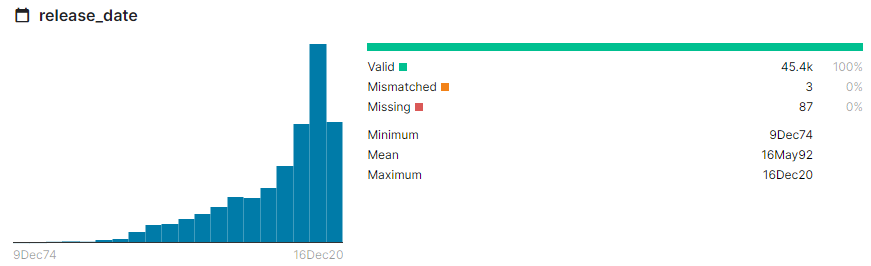
* 1. Trực quan hóa các trường dữ liệu trong movies\_metadata.csv

a

Hình – Đồ thị biểu hiện tỷ trọng cũng như số lượng khách hàng xem phim ở độ tuổi trưởng thành

b

Hình – Đồ thị thể hiện số số lượng vote trung bình của trường dữ liệu vote\_average

c

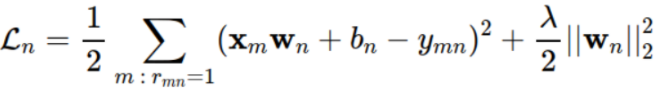
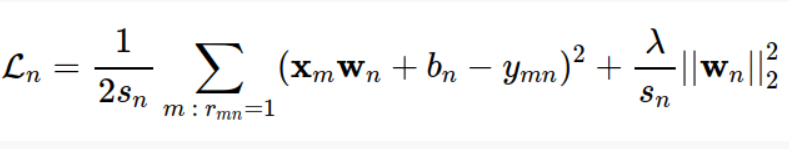
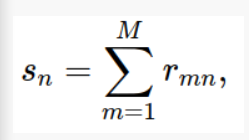
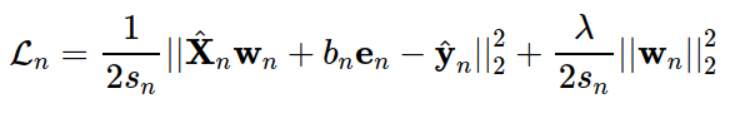
Hình – Đồ thị thể hiện dữ liệu của trường dữ liệu release\_date của movies\_metadata.csv

* 1. Các phương pháp, thuật toán, mô hình được sử dụng
     1. Phương pháp xây dựng dựa theo nội dung (Content Based Filtering)

Ở phương pháp này, các nhà phát triển, xây dựng (developer) cần phải xây dựng được Item profiles và hàm mất mát (theo mô hình tuyến tính).

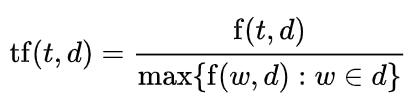
**Item Profile**: Trong các hệ thống content-based, các nội dung của mỗi item sễ được tổng thu thập lại thành một bộ hồ sơ (proflie) được biểu diễn dưới dạng toán học là một feature vector. Trong những trường hợp đơn giản, feature vector được trực tiếp trích xuất từ item.

**Hàm mất mát (Theo mô hình tuyến tính):**

* + - * Giả sử rằng, ta có:
        + Trong ma trận utility Y thì y(i,j) là mức độ quan tâm (ở đây là số sao đã rate) của user thứ i với sản phẩm thứ j 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
        + R thể hiện việc một user đã rated một item hay chưa. Cụ thể, rij = 1 nếu item thứ i đã được rated bởi user thứ j, ngược lại rij = 0 nếu item thứ i chưa được rated bởi user thứ j.
      * *Mô hình tuyến tính*:
        + 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:
        + Trong đó, 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 yn, ta có thể xây dựng hàm mất mát tương tự như sau:
        + 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 bn. Trong thực hành, trung bình cộng của lỗi thường được dùng, và mất mát nên Ln được viết lại thành:
        + 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:
        + Vì hàm mục tiêu chỉ phụ thuộc vào các items đã được rated, ta có thể rút gọn nó bằng cách đặt n 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 ma trận Y. Đồng thời, đặt n là submatrix 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 công thức:
        + Đây chính là bài toán Ridge Regression, đã có sẵn trong thư viện “sklearn.linear\_model.Ridge” của sklearn. Chúng ta sẽ sử dụng thư viện này để tìm w(n) và b(n) cho mỗi user. Còn bây giờ chúng ta sẽ xét một ví dụ về cách xây dựng mô hình cho mỗi user.
    1. Phương pháp TF-IDF vectorizer

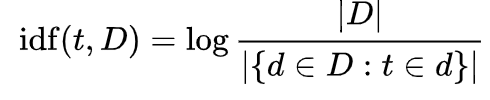
**TF-IDF** (Term Frequency – Inverse Document Frequency) là 1 kĩ thuật sử dụng trong khai phá dữ liệu văn bản. Trọng số này được sử dụng để đánh giá tầm quan trọng của một từ trong một văn bản. Giá trị cao thể hiện độ quan trọng cao và nó phụ thuộc vào số lần từ xuất hiện trong văn bản nhưng bù lại bởi tần suất của từ đó trong tập dữ liệu. Một vài biến thể của tf-idf thường được sử dụng trong các hệ thống tìm kiếm như một công cụ chính để đánh giá và sắp xếp văn bản dựa vào truy vấn của người dùng.  Tf-idf cũng được sử dụng để lọc những từ stopwords trong các bài toán như tóm tắt văn bản và phân loại văn bản.

Trong đó:

* *TF - Term Frequency (Tần suất xuất hiện của từ)*: là số lần từ xuất hiện trong văn bản. Vì các văn bản có thể có độ dài ngắn khác nhau nên một số từ có thể xuất hiện nhiều lần trong một văn bản dài hơn là một văn bản ngắn. Như vậy, term frequency thường được chia cho độ dài văn bản (tổng số từ trong một văn bản).

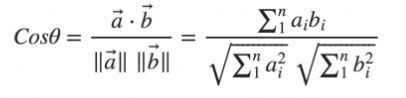
Hình – công thức xác định TF

Với:

* tf(t, d): tần suất xuất hiện của từ t trong văn bản d
* f(t, d): Số lần xuất hiện của từ t trong văn bản d
* max({f(w, d) : w ∈ d}): Số lần xuất hiện của từ có số lần xuất hiện nhiều nhất trong văn bản d
* ***IDF*** *- Inverse Document Frequency (Nghịch đảo tần suất của văn bản)*: giúp đánh giá tầm quan trọng của một từ. Khi tính toán TF, tất cả các từ được coi như có độ quan trọng bằng nhau. Nhưng một số từ như “is”, “of” và “that” thường xuất hiện rất nhiều lần nhưng độ quan trọng là không cao. Như thế chúng ta cần giảm độ quan trọng của những từ này xuống.

Hình – Công thức xác định IDF

* + 1. Consine similarity

Chỉ số Cosine similarity được sử dụng để xác định mức độ tương tự của các tài liệu bất kể kích thước của các đối tượng cần so sánh. Về mặt toán học, chỉ số tương tự cosine sẽ đo cosin của góc giữa hai vectơ được chiếu trong không gian đa chiều. Trong ngữ cảnh này, hai vectơ đó chính là các mảng chứa số lượng từ của hai tài liệu (các tài liệu càng gần nhau theo góc độ, thì Độ tương đồng Cosine càng cao.).

Hình – Công thức tính Consine Similarity

* + 1. Demographic filtering

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.

Hình – Demographic Filtering

CHƯƠNG 3 – CHƯƠNG TRÌNH DEMO

* 1. Chương trình demo không áp dụng PySpark
     1. Quá trình chuẩn bị

Để có thể triển khai chương trình thực nghiệm không sử dụng PySpark, cần phải chuẩn bị trước file dữ liệu cần theo tác (ở đồ án này sử dụng file *movies\_metadata.csv* được tổng hợp trên website kaggle.com). Sau đó, tiến hành khai báo các thư viện cần thiết cho quá trình xây dụng chương trình gồm có *pandas, numpy, time,…*

import numpy as np

import pandas as pd

import time

import os

%matplotlib inline

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

from ast import literal\_eval

from sklearn.feature\_extraction.text import TfidfVectorizer, CountVectorizer

from sklearn.metrics.pairwise import linear\_kernel, cosine\_similarity

from nltk.stem.snowball import SnowballStemmer

from nltk.stem.wordnet import WordNetLemmatizer

from nltk.corpus import wordnet

pd.options.display.max\_columns=None

import warnings; warnings.simplefilter('ignore')

from google.colab import drive

Sau khi đã khai báo xong các thư viện cần thiết, tiến hành load dữ liệu từ dataset *movies\_metadata.csv,* đồng thời khởi tạo biến *begin* để phục vụ cho quá trình so sánh

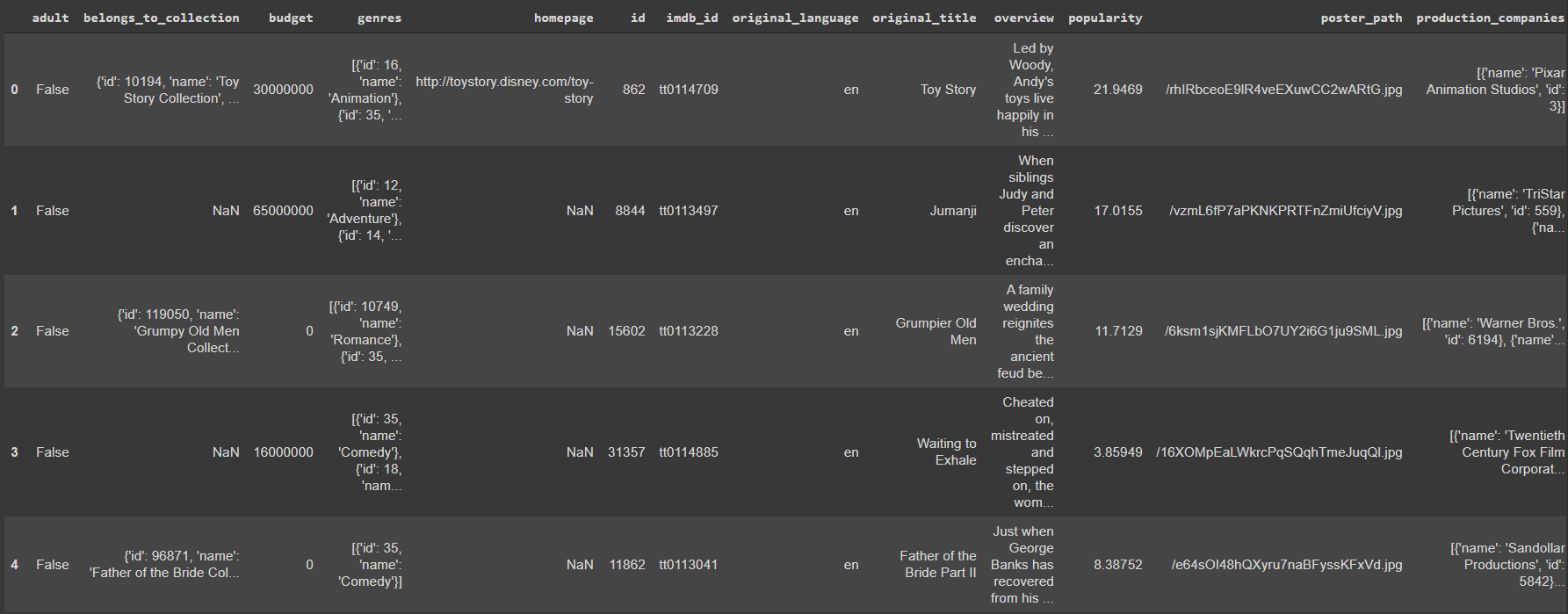
drive.mount('/content/drive',force\_remount=True)

begin = time.time()

data1 = pd.read\_csv("/content/drive/MyDrive/BIGDATA/DOAN/NEWDATASET/movies\_metadata.csv")

data1.head()

Và đây là kết quả sau khi hoàn thành hai bước vừa rồi:



Xác định các giá trị null có trong *movies\_metadata.csv* và tiến hành clean dữ liệu lấy các loại dữ liệu thành một array sau đó lưu vào dataframe:

# Identify the total number of null values in the data

data1.isnull().sum()

# Data cleaning

data1['genres'] = data1['genres'].fillna('[]').apply(literal\_eval).apply(lambda x: [i['name'] for i in x] if isinstance(x, list) else [])

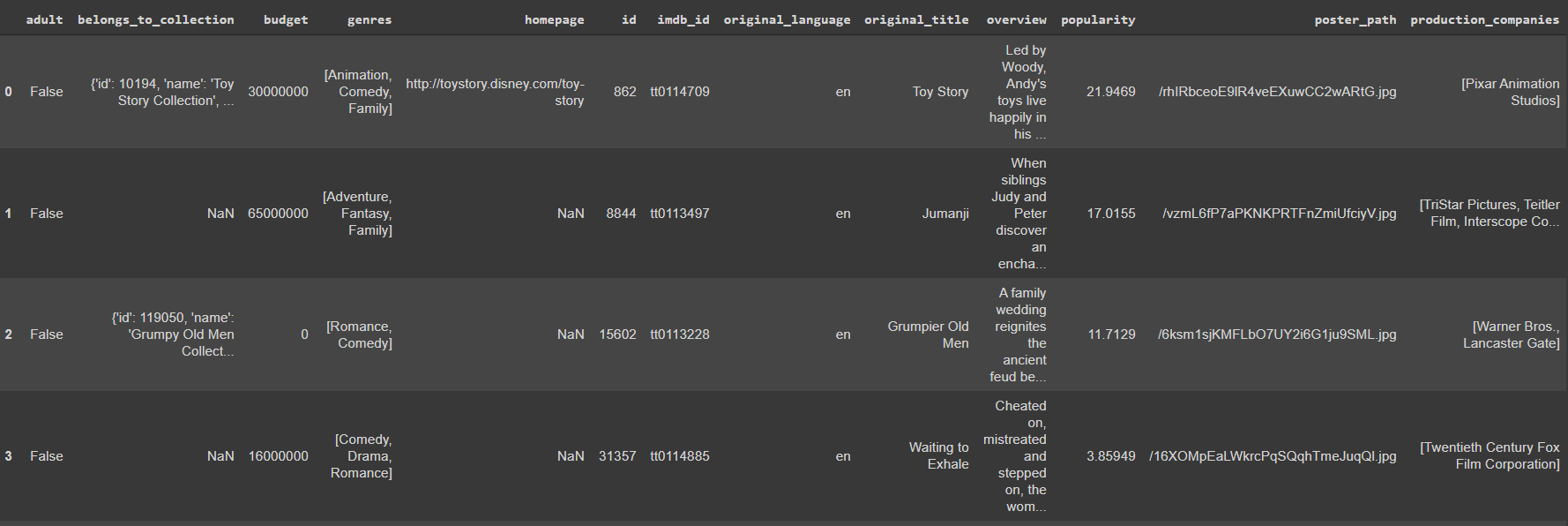
data1['production\_companies']= data1['production\_companies'].fillna('[]').apply(literal\_eval).apply(lambda x: [i['name'] for i in x] if isinstance(x, list) else [])

data1['production\_countries'] = data1['production\_countries'].fillna('[]').apply(literal\_eval).apply(lambda x: [i['name'] for i in x] if isinstance(x, list) else [])

data1['spoken\_languages'] = data1['spoken\_languages'].fillna('[]').apply(literal\_eval).apply(lambda x: [i['name'] for i in x] if isinstance(x, list) else [])

data1['year'] = pd.to\_datetime(data1['release\_date'], errors='coerce').apply(lambda x: str(x).split('-')[0] if x != np.nan else np.nan)

Kết quả sau khi xác định tổng số giá trị null trong dataset:

Và bước clean data:

* + 1. Quá trình Demographic Filtering và chọn ra weight rank

Sử dụng xếp hạng IMDB để đưa ra bảng xếp hạng các bộ phim hàng đầu, sau đó sử dụng công thức xếp hạng có trọng số của IMDB ( <https://www.imdb.com> ) để xây dụng biểu đồ. Công thức được biểu diễn dạng toán học như sau:

Weighted Rank (WR) = (v ÷ (v+m)) × R + (m ÷ (v+m)) × C

Trong đó:

* R = trung bình cho phim = rating
* v là số phiếu bầu cho bộ phim = vote
* m là số phiêu bầu tối thiểu cần thiết được liệt kê trong top 250
* C là số phiếu bầu cho toàn bộ báo cáo

# xác định m => để một bộ phim có mặt trong bảng xếp hạng, nó phải có số phiếu bình chọn >= 96% các bộ phim trong danh sách.

vote\_counts = data1[data1['vote\_count'].notnull()]['vote\_count'].astype('int')

vote\_averages = data1[data1['vote\_average'].notnull()]['vote\_average'].astype('int')

C = vote\_averages.mean()

# Dùng phân vị tìm ra được điểm m có giá trị > 96% toàn bộ dữ liệu còn lại

m = vote\_counts.quantile(0.96)

print('Giá trị trung bình của voting averages =',C)

print('Số phiếu bầu tối thiểu cho một bộ phim cần xem xét =', m)

Output nhận được là:

Biến đổi dữ liệu sang cấu trúc 6 feature và 1819 dòng

qualified = data1[(data1['vote\_count'] >= m) & (data1['vote\_count'].notnull()) & (data1['vote\_average'].notnull())][['title', 'year', 'vote\_count', 'vote\_average', 'popularity', 'genres']]

qualified['vote\_count'] = qualified['vote\_count'].astype('int')

qualified['vote\_average'] = qualified['vote\_average'].astype('int')

print('Cấu trúc mới của hệ dữ liệu là = ',qualified.shape)

Tiến hành xếp hạng cho 250 bộ phim theo thứ tự giảm dần

def weighted\_rating(x):

    v = x['vote\_count']

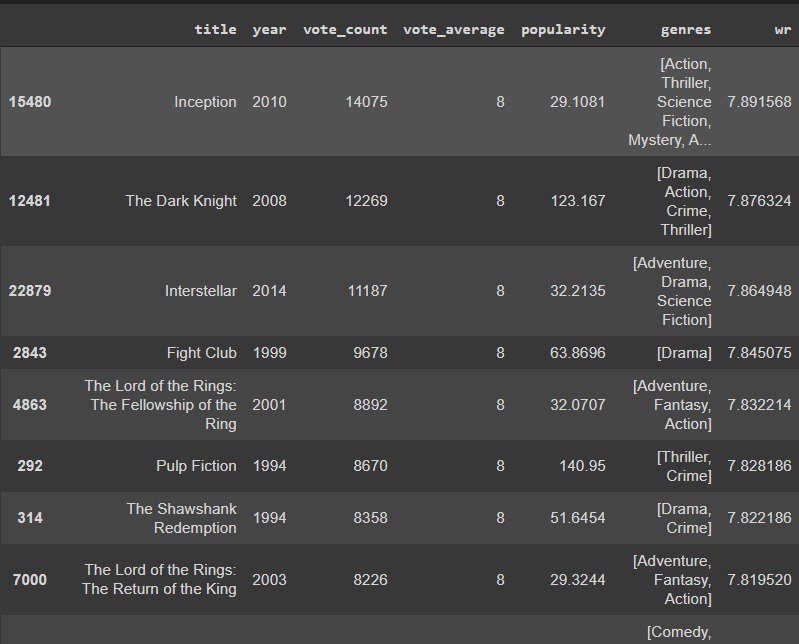
    R = x['vote\_average']

    return (v/(v+m) \* R) + (m/(m+v) \* C)

qualified['wr'] = qualified.apply(weighted\_rating, axis=1)

qualified = qualified.sort\_values('wr', ascending=False).head(250

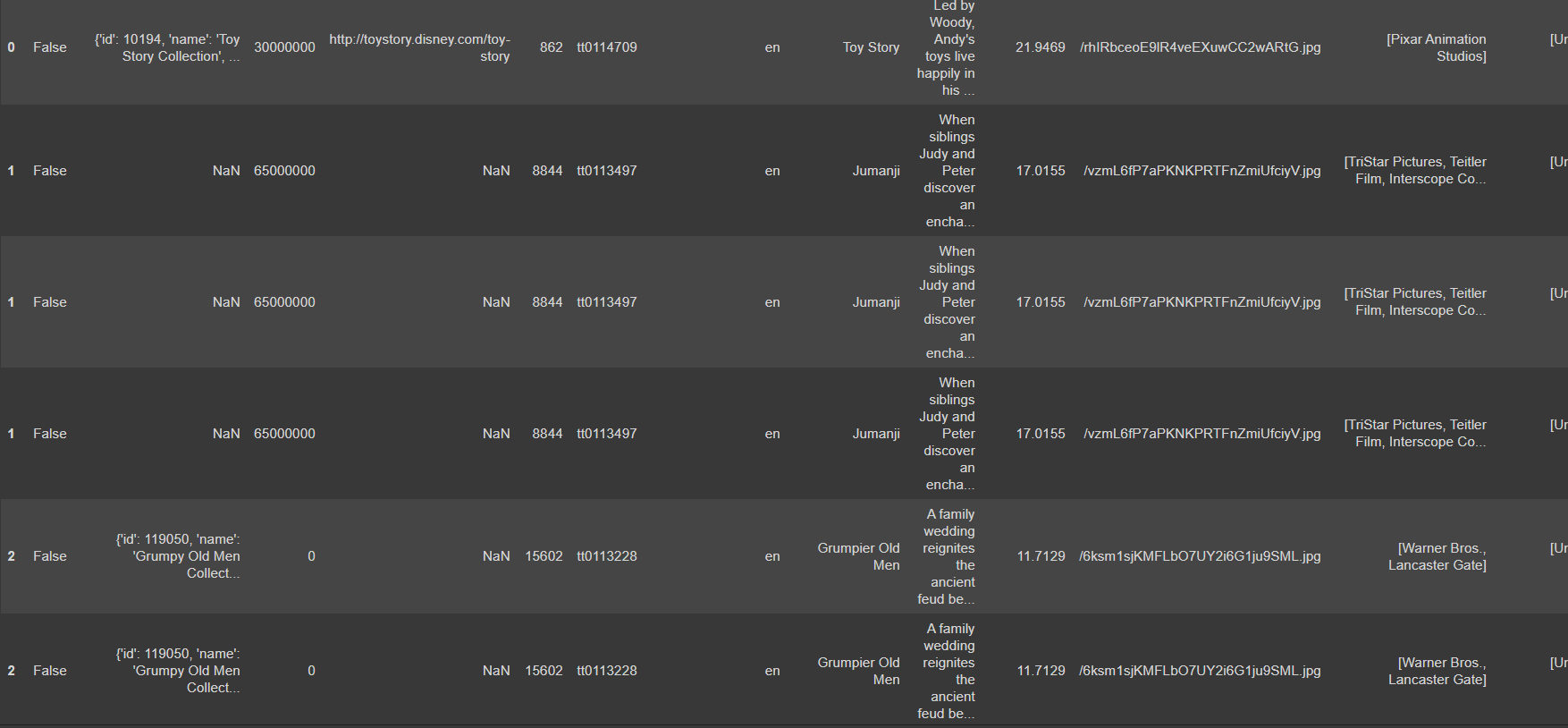
Tiến hành chọn ra top 15 được xếp hạng cao nhất, rồi tổng hợp tất cả cá tâp data dựa trên trường Genres

qualified.head(15)

s = data1.apply(lambda x: pd.Series(x['genres']),axis=1).stack().reset\_index(level=1, drop=True)

s.name = 'genre'

gen\_md = data1.drop('genres', axis=1).join(s)

gen\_md.head(8)

def build\_chart(genre, percentile=0.90):

    df = gen\_md[gen\_md['genre'] == genre]

    vote\_counts = df[df['vote\_count'].notnull()]['vote\_count'].astype('int')

    vote\_averages = df[df['vote\_average'].notnull()]['vote\_average'].astype('int')

    C = vote\_averages.mean()

    m = vote\_counts.quantile(percentile)

    qualified = df[(df['vote\_count'] >= m) & (df['vote\_count'].notnull()) & (df['vote\_average'].notnull())][['title', 'year', 'vote\_count', 'vote\_average', 'popularity']]

    qualified['vote\_count'] = qualified['vote\_count'].astype('int')

    qualified['vote\_average'] = qualified['vote\_average'].astype('int')

    # Công thức tính WR

    qualified['weight\_rank'] = qualified.apply(lambda x: (x['vote\_count']/(x['vote\_count']+m) \* x['vote\_average']) + (m/(m+x['vote\_count']) \* C), axis=1)

    qualified = qualified.sort\_values('weight\_rank', ascending=False).head(250)

    return qualified

build\_chart('Horror').head(15)

Output nhận được:

Tương tự ta thực hiện build\_chart cho trường “*comedy*”, “*Romance*”. Tiếp nối sau đó, tiến hành merge từng tagline và overview:

data1 = data1.drop([19730, 29503, 35587])

data1['id'] = data1['id'].astype('int')

links\_small = pd.read\_csv('/content/drive/MyDrive/BIGDATA/DOAN/NEWDATASET/links\_small.csv')

links\_small = links\_small[links\_small['tmdbId'].notnull()]['tmdbId'].astype('int')

smd = data1[data1['id'].isin(links\_small)]

smd['tagline'] = smd['tagline'].fillna('')

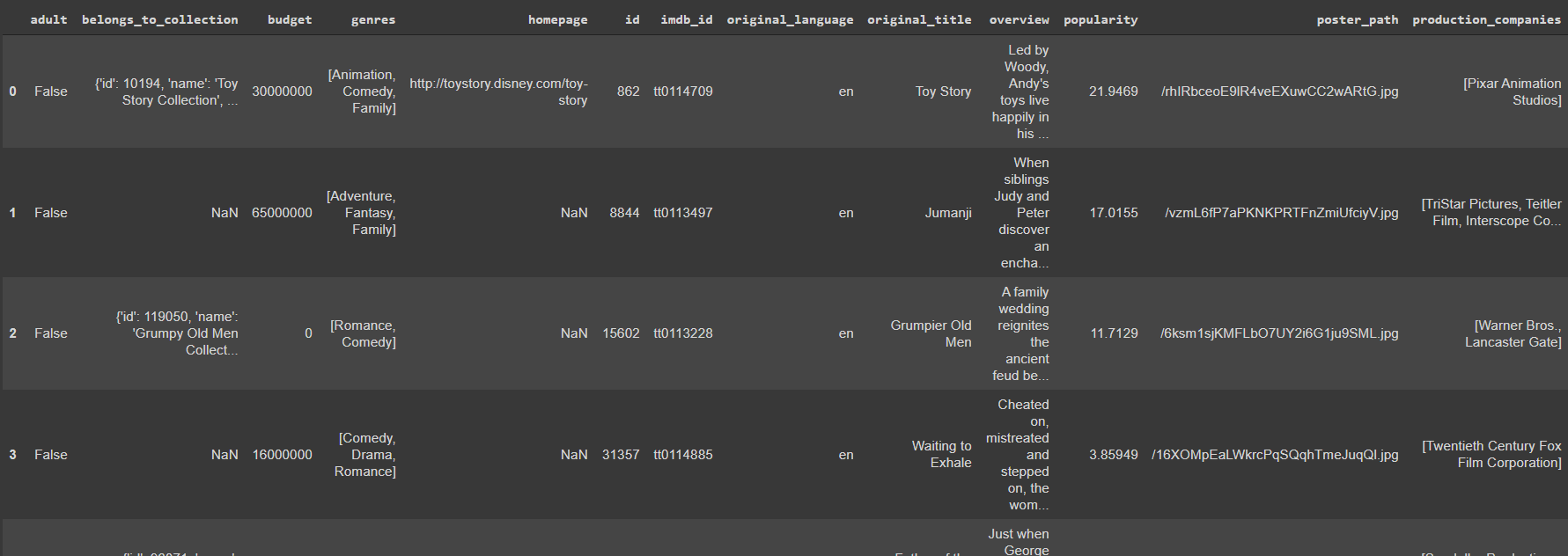
smd['overview'] = smd['overview'].fillna('')

smd['description'] = smd['overview'] + smd['tagline']

smd['description'] = smd['description'].fillna('')

len(smd)

smd.head()

Output sẽ là:

* + 1. TF-IDF Vectorizer

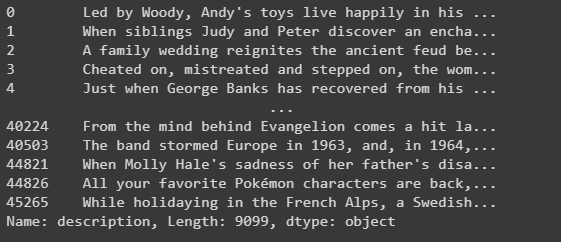
Bước tiếp theo đó chính là sử dụng TF-IDF Vectorizer nhằm mục đích chuẩn bị cho việc tính chỉ số tương tự consine

print(smd['description'])

tf = TfidfVectorizer(analyzer='word',ngram\_range=(1, 2),min\_df=0, stop\_words='english')

tfidf\_matrix = tf.fit\_transform(smd['description'])

Output nhận được của bước này là;



* + 1. Consine Similarity

Chuẩn bị các biến cần thiết:

cosine\_sim = linear\_kernel(tfidf\_matrix, tfidf\_matrix)

smd = smd.reset\_index()

titles = smd['title']

indices = pd.Series(smd.index, index=smd['title'])

print(indices["Toy Story"])

Tiến hành tạo hàm *get\_recommendations(title)* để nhận vào các title mà khách hàng thường xuyên hay đã từng truy cập và đưa ra kết quả là các bộ phim có liên quan đến title được nhập đó

def get\_recommendations(title):

    movies = np.array(smd[smd['title'].str.contains(title)].title.index)

    print(f"Có {len(movies)} được tìm thấy")

    for i,m in enumerate(movies):

      print(f"{i+1} --- {smd.loc[m]['title']}")

    id\_select = int(input("Chọn một bộ phim: "))

    movie = movies[id\_select - 1]

    print(f"Phim được chọn: {smd.loc[movie].title}\n----------------------------\n Phim được đề xuất")

    idx = indices[smd.loc[movie].title]

    # Tính cosine similarity

    sim\_scores = list(enumerate(cosine\_sim[idx]))

    # sort

    sim\_scores = sorted(sim\_scores, key=lambda x: x[1], reverse=True)

    # bỏ cái đầu vì trùng

    sim\_scores = sim\_scores[1:11]

    movie\_indices = [[i[1],smd.loc[i[0]].title] for i in sim\_scores]

    result\_df = pd.DataFrame(movie\_indices,columns=['Similarity','Movies'])

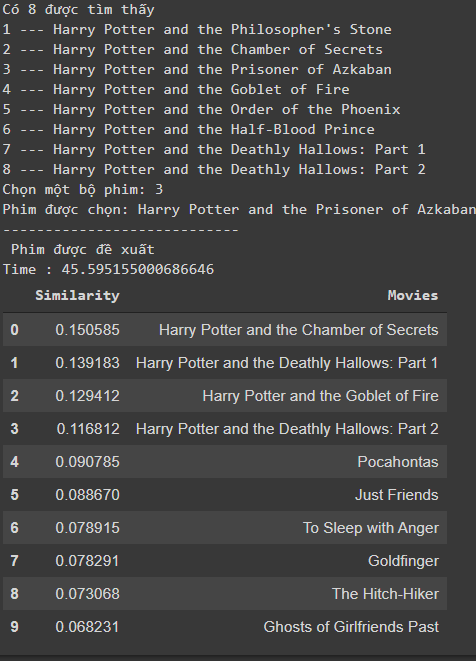
    end = time.time()

    result\_time = end - begin

    print("Time :" ,result\_time)

    return result\_df

get\_recommendations('Harry Potter')

Output nhận được với title là “Harry Potter như sau”

* 1. Chương trình demo có áp dụng PySpark
     1. Quá trình chuẩn bị

Để có thể triển khai chương trình thực nghiệm không sử dụng PySpark, cần phải chuẩn bị trước file dữ liệu cần theo tác (ở đồ án này sử dụng file *movies\_metadata.csv* được tổng hợp trên website kaggle.com). Tiến hành thực hiện khai báo các thư viện cần thiết và load dataset (*movies\_metadata.csv*):

* + - * Khai báo các thư viện cần thiết

import pyspark

import pandas as pd

import numpy as np

from pyspark import SparkConf, SparkContext

from pyspark.sql import SparkSession

from google.colab import drive

import time

from pyspark.sql import SQLContext

drive.mount('/content/drive')

import collections

from pyspark.mllib.linalg import \*

from pyspark.ml.feature import HashingTF, IDF, Tokenizer, StopWordsRemover

from pyspark.mllib.linalg import SparseVector

from scipy.spatial import distance

import json

* + - * Load dataset, chuẩn bị biến time, spark để phục vụ cho quá trình song song hóa và quá trình so sánh thời gian thực thi đằng sau:

spark = SparkSession.builder.appName('MOVIES\_RECOMMENDATION\_SYSTEM').getOrCreate()

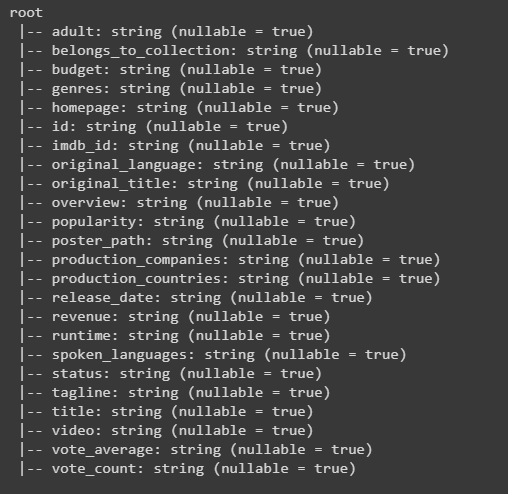
# load data

smd = spark.read.csv('/content/drive/MyDrive/BIGDATA/DOAN/NEWDATASET/movies\_metadata.csv', header = True, inferSchema = True)

begin1 = time.time()

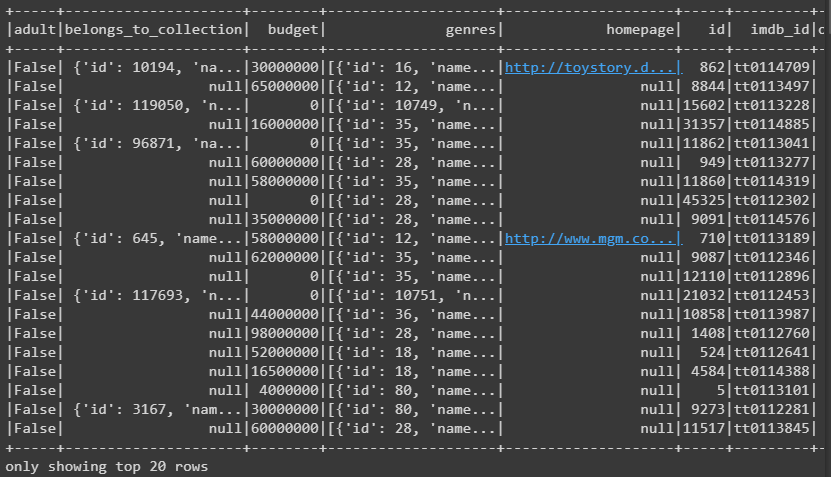
smd = smd.withColumn("id", smd.id.cast("int"))

smd.printSchema()

* + - * Output nhận được của quá trình này như sau:

smd = smd.fillna({'tagline':'', 'overview':""})

smd.show()

Output nhận được là:

from pyspark.sql.functions import concat, lit

col\_list = ['tagline','overview']

smd = smd.withColumn('description',concat(\*col\_list))

* + 1. TF-IDF Vectorizer

Quá trình thực hiện TF-IDF Vectorizer được thực hiện như sau:

from pyspark.ml.feature import CountVectorizer,RegexTokenizer

regexTokenizer = RegexTokenizer(inputCol="description", outputCol="words", pattern="\\W")

wordsData = regexTokenizer.transform(smd)

hashingTF = HashingTF(inputCol="words", outputCol="tfidf", numFeatures=1000)

featurizedData = hashingTF.transform(wordsData)

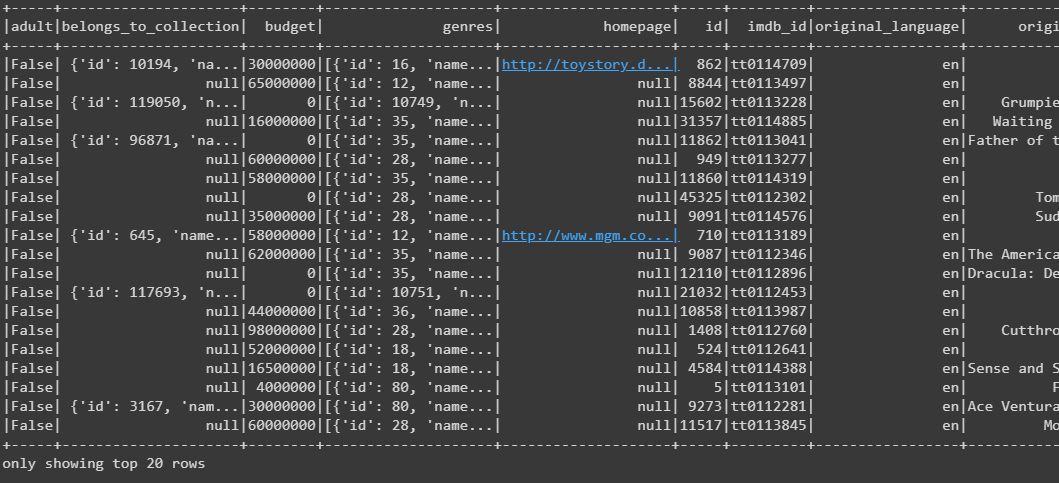
idf = IDF(inputCol="tfidf", outputCol="tfidf\_features")

idf\_model = idf.fit(featurizedData)

wordsData = idf\_model.transform(featurizedData)

wordsData.show()

end1 = time.time() - begin1

và với cách thực hiện như trên, hệ thống sẽ trả về kết quả sau:

* + 1. Consine Similarity

Các hàm cần thiết cho bước này gồm có:

def cosin(x, y):

  return x.dot(y)/(x.norm(2)\*y.norm(2))

def recommend(y):

  begin2 = time.time()

  b = wordsData.rdd.map(lambda x: (x["tfidf\_features"], x['original\_title']))

  result = b.map(lambda x:([cosin(x[0],y)], x[1]))

  return result.takeOrdered(11, key = lambda x: -x[0][0]),begin2

def get\_recommendations(title):

  movies = wordsData.where(f"UPPER(original\_title) like UPPER('%{title}%')").select(["original\_title","id","tfidf\_features"]).distinct().collect()

  if len(movies) > 0:

    print(f'Có {len(movies)} bộ phim được tìm thấy')

  for i,m in enumerate(movies):

      print(f"{i+1} --- {m.original\_title}")

  id\_select = int(input("Chọn một bộ phim: "))

  movie = movies[id\_select - 1]

  print(f"Phim được chọn: {movie[0]}\n----------------------------\n Phim được đề xuất")

  recommend\_result,begin2 = recommend(movie[2])

  end2 = time.time() - begin2

  print("Time:",end1 + end2)

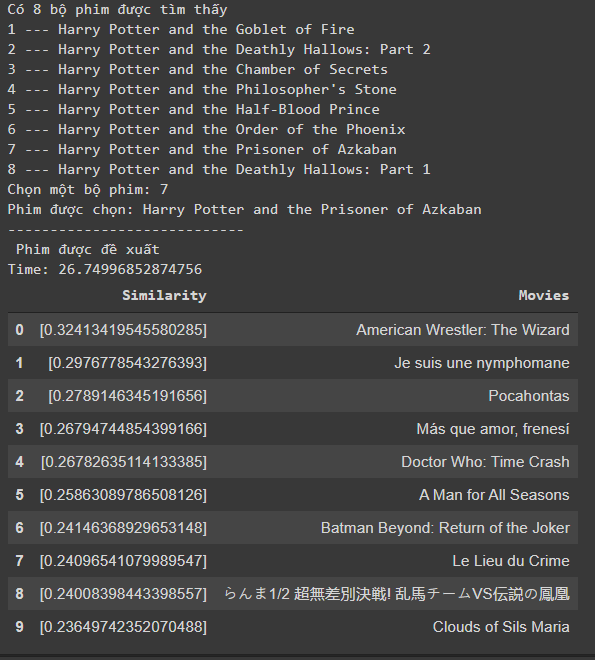
  recommend\_result = recommend\_result[1:11]

  result\_df = pd.DataFrame(recommend\_result,columns=['Similarity','Movies'])

  return result\_df

Sau khi đã chuẩn bị đầy đủ các hàm cần thiết để tính chỉ số tương tự cosine, ta tiến hành thực hiện tính toán và đưa ra kết quả là các bộ phim kiến nghị phù hợp với title đầu vào:

get\_recommendations('Harry Potter')

Và kết quả cuối cùng nhận được như sau:

***Nhận xét*:**

Dựa vào biến time, ta có thể thấy rằng nhờ có sử dụng pyspark cũng như việc xong xong hóa các bước có thể mà thời gian xử lý, tính toàn và kết quả trả về được thực thi một cách nhanh chóng và tiết kiệm chi phí nhiều hơn. Nếu chỉ xét chi phí cụ thể là thời gian, thì căn cứ vào kết quả mà biến *time* đã trả về, ta có thể dễ dàng biết được rằng hệ thống khuyến nghị (Recommder Systems) có kết hợp PySpark nhanh hơn hẳn khi không sử dụng hay triển khai kết hợp cùng PySpark (*timeunusePySpark* = 45.595155000686646 > *timeuse PySpark =* 26.74996852874756).

Ngoài ra, tập dataset được áp dụng khi sử dụng pyspark lớn hơn gấp 4 lần so với khi không chạy spyspark. Đây là một trường hợp xảy ra khi không áp dụng kỹ thuật pyspark dẫn đến tiến trình khi thực thi trên chương trình thường, việc tính toán tuần tự dẫn đến bị tràn bộ nhớ (RAM) và bị ngắt giữa chừng làm tiến trình xảy ra lỗi.

TÀI LIỆU THAM KHẢO

**Tiếng Việt**

1. <https://machinelearningcoban.com/2017/05/17/contentbasedrecommendersys/>
2. <https://techmaster.vn/posts/35386/cach-xay-dung-recommender-system-rs-phan-1>
3. <https://viblo.asia/p/lam-the-nao-de-xay-dung-mot-recommender-system-rs-phan-3-E375zbeW5GW>
4. <https://nguyenvanhieu.vn/tf-idf-la-gi/>
5. <https://tailieu.vn/doc/luan-van-thac-si-ung-dung-he-thong-tu-van-recommender-systems-trong-linh-vuc-thuong-mai-dien-tu-1602259.html>
6. <https://itzone.com.vn/vi/article/tim-hieu-ve-content-based-filtering-phuong-phap-goi-y-dua-theo-noi-dung-phan-1/>

**Tiếng Anh**

1. <https://blog.avenuecode.com/how-to-build-a-recommender-system-in-less-than-1-hour#:~:text=Content%20Based%20Recommender%20System,similar%20preferences%20(Meteren%2C%20et>.
2. <https://machinelearningcoban.com/2017/05/17/contentbasedrecommendersys/>
3. <https://en.wikipedia.org/wiki/Recommender_system#:~:text=A%20recommender%20system%2C%20or%20a,would%20give%20to%20an%20item>.
4. <https://developers.google.com/machine-learning/recommendation>
5. <https://www.machinelearningplus.com/nlp/cosine-similarity/>

**Dataset**

<https://www.kaggle.com/rounakbanik/the-movies-dataset?select=movies_metadata.csv>