TRƯỜNG ĐẠI HỌC CÔNG NGHỆ - ĐẠI HỌC QUỐC GIA HÀ NỘI
VIỆN TRÍ TUỆ NHÂN TẠO

BÁO CÁO MÔN HỌC KỸ THUẬT VÀ CÔNG NGHỆ DỮ LIỆU LỚN

ĐỀ TÀI

Phân tích dữ liệu lớn đề xuất sản phẩm cho người dùng

Nhóm sinh viên thực hiện: Đỗ Minh Nhật - Trần Nam Anh - Hà Như Ý

Giảng viên hướng dẫn: TS. Trần Hồng Việt

Ths. Ngô Minh Hương

Hà Nội, ngày 27 tháng 05 năm 2024

MỞ ĐẦU

Trong thời đại công nghệ số hiện nay, sự bùng nổ dữ liệu đã mở ra nhiều cơ hội và thách thức mới cho các doanh nghiệp trong việc hiểu và phục vụ khách hàng tốt hơn. Với lượng dữ liệu khổng lồ được tạo ra hàng ngày từ các hoạt động trực tuyến và mạng xã hội, việc tận dụng những dữ liệu này để đưa ra các gợi ý sản phẩm phù hợp cho người dùng đã trở thành một yếu tố then chốt trong chiến lược kinh doanh của nhiều công ty.

Dự án phân tích dữ liệu lớn nhằm đề xuất gợi ý sản phẩm cho người dùng được triển khai với mục tiêu tối ưu hóa trải nghiệm mua sắm của khách hàng và tăng cường doanh số bán hàng. Hệ thống gợi ý này sử dụng các kỹ thuật phân tích dữ liệu lớn, kết hợp với các thuật toán máy học để phân tích hành vi và sở thích của người dùng, từ đó đưa ra các đề xuất sản phẩm cá nhân hóa và phù hợp với nhu cầu của từng khách hàng.

Dự án này sẽ tập trung vào việc áp dụng các phương pháp phân tích và mô hình học máy để xây dựng một hệ thống gợi ý sản phẩm thông minh. Kết quả mong đợi là một hệ thống có khả năng học hỏi liên tục từ dữ liệu mới, cải thiện độ chính xác của các gợi ý và mang lại giá trị thực tiễn.

Báo cáo này bao gồm:

Chương 1: Tổng quan về dữ liệu lớn

Chương 2: Thuật toán gợi ý sản phẩm

Chương 3: Thực nghiệm và đánh giá

Chương 4: Kết luận và hướng phát triển

MỤC LỤC

Chương 1: Tổng quan về dữ liệu lớn

Dữ liệu Lớn

PySpark

1. Giới thiệu

PySpark là một API của Apache Spark dành cho ngôn ngữ lập trình Python. Apache Spark là một framework mã nguồn mở mạnh mẽ dùng để xử lý và phân tích dữ liệu lớn theo cách phân tán và song song. PySpark cho phép các nhà phát triển và nhà khoa học dữ liệu sử dụng Spark với Python, mang lại sự linh hoạt và tiện lợi của Python cùng với hiệu suất cao và khả năng mở rộng của Spark

.

2. Kiến trúc và thành phần của PySpark:

2.1. Kiến trúc của Apache Spark

Apache Spark có kiến trúc dựa trên cụm (cluster) bao gồm các thành phần chính:

- Driver Program: Điều khiển việc thực thi của ứng dụng Spark. Driver tạo ra SparkContext, xác định các RDDs (Resilient Distributed Datasets), và thực hiện các hành động trên chúng.
- Cluster Manager: Quản lý các tài nguyên trong cụm máy tính, chẳng hạn như YARN, Mesos hoặc Kubernetes.
- Workers: Các máy tính trong cụm thực hiện các tác vụ (tasks) được giao bởi Driver.

2.2. Thành phần của PySpark

PySpark bao gồm các thành phần chính:

1. SparkContext: Là cổng vào của Spark, dùng để kết nối với cụm Spark.
2. RDD (Resilient Distributed Datasets): Cấu trúc dữ liệu cơ bản trong Spark, giúp xử lý dữ liệu theo cách phân tán và bất biến.
3. DataFrame: API cấp cao hơn của RDD, cung cấp giao diện dữ liệu dạng bảng với các thao tác SQL và tích hợp tốt với các công cụ dữ liệu khác như Pandas.
4. Spark SQL: Cho phép thao tác và xử lý dữ liệu thông qua ngôn ngữ SQL.
5. MLlib (Machine Learning Library): Thư viện học máy của Spark, cung cấp các thuật toán và công cụ học máy phân tán.
6. GraphX: Thư viện xử lý đồ thị và phân tích đồ thị.

2.3. Ứng dụng của PySpark

- Xử lý và Phân tích Dữ liệu Lớn: PySpark được sử dụng rộng rãi để xử lý và phân tích dữ liệu lớn trong nhiều lĩnh vực khác nhau như thương mại điện tử, tài chính, y tế và công nghiệp. Các công ty như Netflix, Amazon và eBay sử dụng PySpark để phân tích hành vi người dùng, dự đoán xu hướng và tối ưu hóa dịch vụ của họ.

- Học Máy và Trí Tuệ Nhân Tạo: Với thư viện MLlib, PySpark cung cấp các công cụ và thuật toán học máy phân tán, cho phép triển khai các mô hình học máy trên các tập dữ liệu lớn. Điều này giúp cải thiện độ chính xác và hiệu quả của các mô hình dự đoán.

- Xử lý Dữ liệu Thời Gian Thực: PySpark hỗ trợ xử lý dữ liệu thời gian thực thông qua Spark Streaming, cho phép xử lý và phân tích luồng dữ liệu liên tục từ các nguồn dữ liệu như Kafka, Flume và Kinesis.

Chương 2: Thuật toán gợi ý sản phẩm

2.1. Thuật toán content-based filtering

2.1.1. Giới thiệu

Hệ thống này đề xuất nội dung cho người dùng dựa trên nội dung mà người dùng đó thích. Ví dụ, người dùng A thích xem bộ phim có thể loại hài hước, vui vẻ, trong tương lai hệ thống sẽ đề xuất các nội dung liên quan đến bộ phim có thể loại hài hước, vui vẻ. Cách tiếp cận này yêu cầu phải có vector đặc trưng của mỗi item, không chỉ vậy đánh giá của mỗi người dùng và item cũng rất quan trọng vì nếu không có user rating của user đối với item ta không thể nào biết được user đó có sở thích là gì để có thể đề xuất nội dung tương tự. Tuy nhiên, đôi lúc trong thực tế việc xác định feature vector của item có thể gặp khó khăn do khó xác định được nhóm cục thể các item.

2.1.2. Method.

2.1.2.1 Xây dựng Utility Matrix.

Việc xây dựng Utility Matrix thường rất quan trọng, nếu không có Utility Matrix dường như ta không thể gợi ý được sản phẩm tới người dùng, ngoại trừ cách gợi ý các sản phẩm phổ biến nhất. Do đó, việc xây dựng Utility Matrix trong bài toán recommendation là cực kỳ quan trọng. Có 2 thực thể chính trong Recommendation System đó là users và items. Trong đó, mỗi user sẽ có mức độ quan tâm tới từng item. Trong thực tế, mức độ quan tâm đó thường được phân tích dựa trên rating của user đối với item, hoặc like/dislike của user đối với item đó.

	A	B	C	D	E	F
Mưa nửa đêm	5	5	0	0	1	?
Cỏ úa	5	?	?	0	?	?
Vùng lá me bay	?	4	1	?	?	1
Con cò bé bé	1	1	4	4	4	?
Em yêu trường em	1	0	5	?	?	?

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

Hình 1: Utility Matrix mô tả mức độ yêu thích của user với item.

Nguồn: https://machinelearningcoban.com/2017/05/17/contentbasedrecommendersys/

2.1.2.2 Xây dựng Item Profiles.

Trong Content-based Recommendation System, hệ thống đề xuất dựa trên nội dung. Chúng ta cần xây dựng một bộ hồ sơ cho mỗi item còn được gọi là profile item. Profile item này thường được biểu diễn dưới dạng vector thường được gọi là feature vector. Trong trường hợp đơn giảm, item profile được trích xuất từ item. Ví dụ: một bộ phim khi sản xuất sẽ thường bao gồm các nội dung như: Đạo diễn, Năm phát hành, Thể loại,... từ thông tin đó ta có thể xây dựng feature vector cho từng bộ phim.

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

Hình 2: Feature vector của bài hát được đặt ở cuối cùng. Các entry trong utility matrix *(i, j)* đại diện cho rating của item thứ *i* được user thứ *j* rated, trong đó *(i, j)* là vị trí hàng thứ *i* cột thứ *j*. Nguồn: https://machinelearningcoban.com/2017/05/17/contentbasedrecommendersys/

Ví dụ ở hình 2 phía trên, ta xây dựng feature vector x_i với mỗi bài hát. Trong đó, chiều thứ nhất của vector x_i là số thực đo mức độ bài hát liên quan đến Bolero và chiều thứ 2 của vector x_i là số thực đo mức độ bài hát liên quan đến Thiếu nhi. Sau khi xây dựng trong feature vector, ta có thể chỉ ra rằng trong hình 1 user A thích nghe nhạc Bolero và không thích nghe nhạc thiếu nhi. User C thích nghe nhạc thiếu nhi và không thích nghe nhạc Bolero. Khi đó hệ thống sẽ ưu tiên đề xuất cho user A nhạc Bolero, còn user B sẽ được hệ thống đề xuất nhạc thiếu nhi.

2.1.2.3 Xây dựng Objective Function.

Với mỗi user trong toàn bộ user ta cần tìm một bộ tham số θ_i bao gồm (w_i, b_i) sao cho ta có thể xấp xỉ hàm mục tiêu y. Gọi hàm xấp xỉ mục tiêu là $\hat{y}_{mn}$, ta có:

$$\hat{y}_{mn} = x_m w_n + b_n$$

Với x_i là vector hàng, w_j là vector cột.

Ở đây ta có thể xây dựng loss function tuỳ ý. Một trong các cách đơn giản thường được sử dụng đó là Square Error.

$$L_n = \frac{1}{2} \sum_{m: r_{mn}=1} (x_m w_n + b_n - y_{mn})^2$$

Trong đó x_m la feature vector của item, w_n là weight matrix đại diện cho sở thích của user với từng từng feature trong feature vector, b_n là bias vector, y_{mn} là target value (đại diện cho rating của item thứ m được user thứ n rated). $\{m : r_{mn} = 1\}$là tập các vị trí (m, n) trong matrix mà user thứ n đã rating cho item thứ m.

2.1.2.4 Phương pháp tiếp cận sử dụng MapReduce

Bước 1: Tính toán user profile (sở thích của user, ví dụ: xem phim hài hước, vui nhộn)

- *Mapper*: (key, value) => (user_id, features_of_item_id_have_ interacted) (ở đây đối với item mà người dùng thích feature vector là chính nó, đối với item mà người dùng không thích feature vector sẽ bằng với giá trị âm của feature vector đó feature vector = -feature_vector).
- *Reducer* : (key, value) => (user_id, sum(features_of_item_id_have_interacted)).

Bước 2: Tính cosine similarity giữa user và movie:

- *Mapper*: (key, value) => (user_id, (movie_id, score)).
- *Reducer*: Chọn một thresh hold có giá trị bằng A, nếu giá trị của score >= A giữ lại các cặp (user_id, (movie_id, score)).

Bước 3: Lọc các cặp có giá trị score nhỏ hơn threshold, giữ lại các cặp lớn hơn threshold tương ứng với các cặp có thể đề xuất kết bạn

- *Mapper*: Truyền output của Reducer trong *Bước 2* cho Reducer.
- *Reducer*: Reduce về dạng (key,value) = (user_id, movie_id). Khi đó user_id sẽ được đề xuất xem các movie_id.

2.2 Collaborative Filtering

2.2.1 Introduction

Trong Content-based Recommendation System, chung ta đã biết cách tạo ra một hệ thống đề xuất dựa trên các đặc trưng của mỗi item. Đặc trưng của Content-based Recommendation System là đề xuất dựa trên đặc trưng của mỗi item, nhưng câu hỏi đặt ra là nếu giả sử user không rating bất cứ item nào hoặc rất it làm sao ta có thể đề xuất cho user tới item phù hợp. Trong thực tế, việc user rating item là rất ít, không phải người nào xem youtube cũng bấm like/dislike, không phải người nào sau khi mua hàng cũng đánh giá sản phẩm. Không chỉ vậy, trong thực tế có rất nhiều các user có thể có các sở thích giống nhau ví dụ như user A thích item X, item Y, item Z, user B thích item X, item Y, vậy có khả năng rất cao là user B cũng thích item Z. Câu hỏi đặt ra là:

1. Làm thế nào ta có thể xác định được sự giống nhau giữa các cặp users.
2. Khi đã biết các cặp users gần giống nhau làm thế nào để dự đoán mức độ quan tâm của user này dựa trên user khác mà gần giống với user này

2.2.2 Method

2.2.2.1. 1D Embedding: Giả sử chúng tôi gán cho mỗi bộ phim một số thực trong khoảng [-1, 1] mô tả bộ phim đố cho trẻ em hay cho người lớn với giá trị càng gần với -1 đại diện cho bộ phim dánh cho trẻ em và bộ phim càng gần với 1 dành cho người lớn. Tương tự trong khoảng [-1, 1] ta cũng gán mỗi user một giá trị nằm trong khoảng [-1,1] đại diện cho tuổi của user. Nếu user càng ít tuổi ta gán giá trị càng gần với -1, ngược lại user càng nhiều tuổi ta gán cho user đó giá trị càng gần với 1.

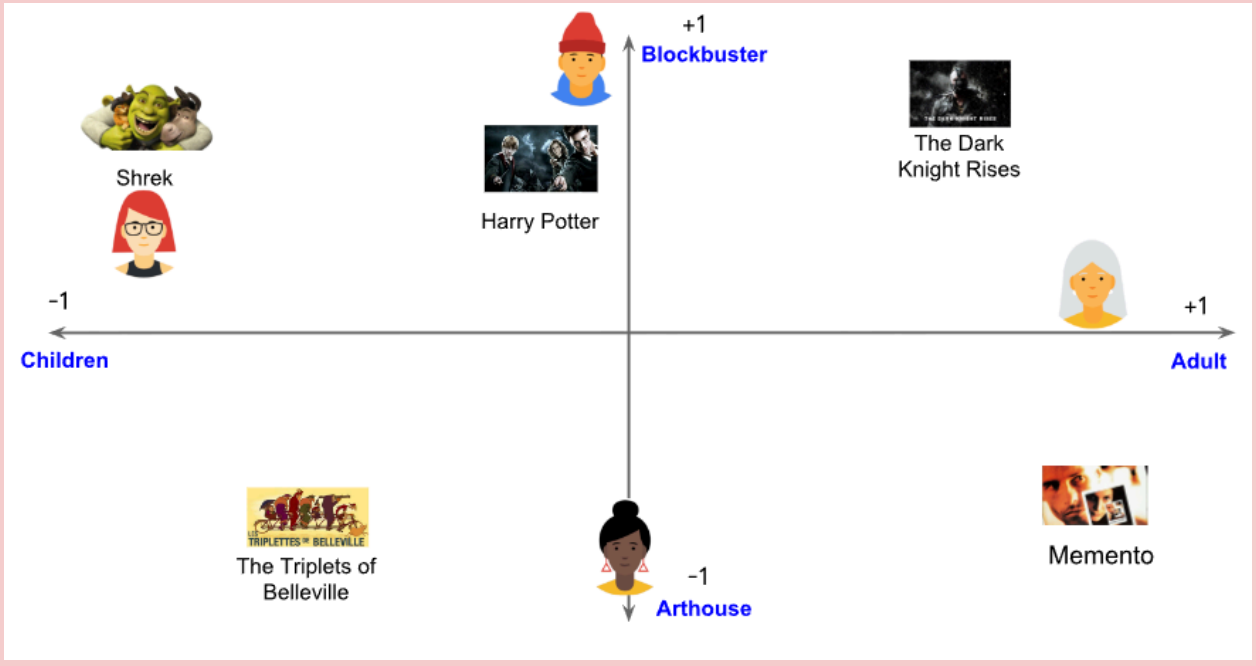
Hình 3: Các dấu checkmark cho ta biết bộ phim mà một user cụ thể đã xem. Từ các dấu checkmark đó ta có thể dễ dàng suy ra được sử thích của người dùng. Khi đó ta có thể biết được những người có độ tuổi trưởng thành thường không thích xem phim dành cho trẻ con và đề xuất phim cho người trưởng thành.

Nguồn: https://developers.google.com/machine-learning/recommendation/collaborative/basics

2.2.2.2. 2D Embedding

Trong trường hợp một feature không thể đại diện rõ cho từng user và item, ví dụ: ta muốn phân biệt rõ hơn user đầu tiên và user thứ 2 trong Hình 3. Ta có thể thêm 1 feature vào

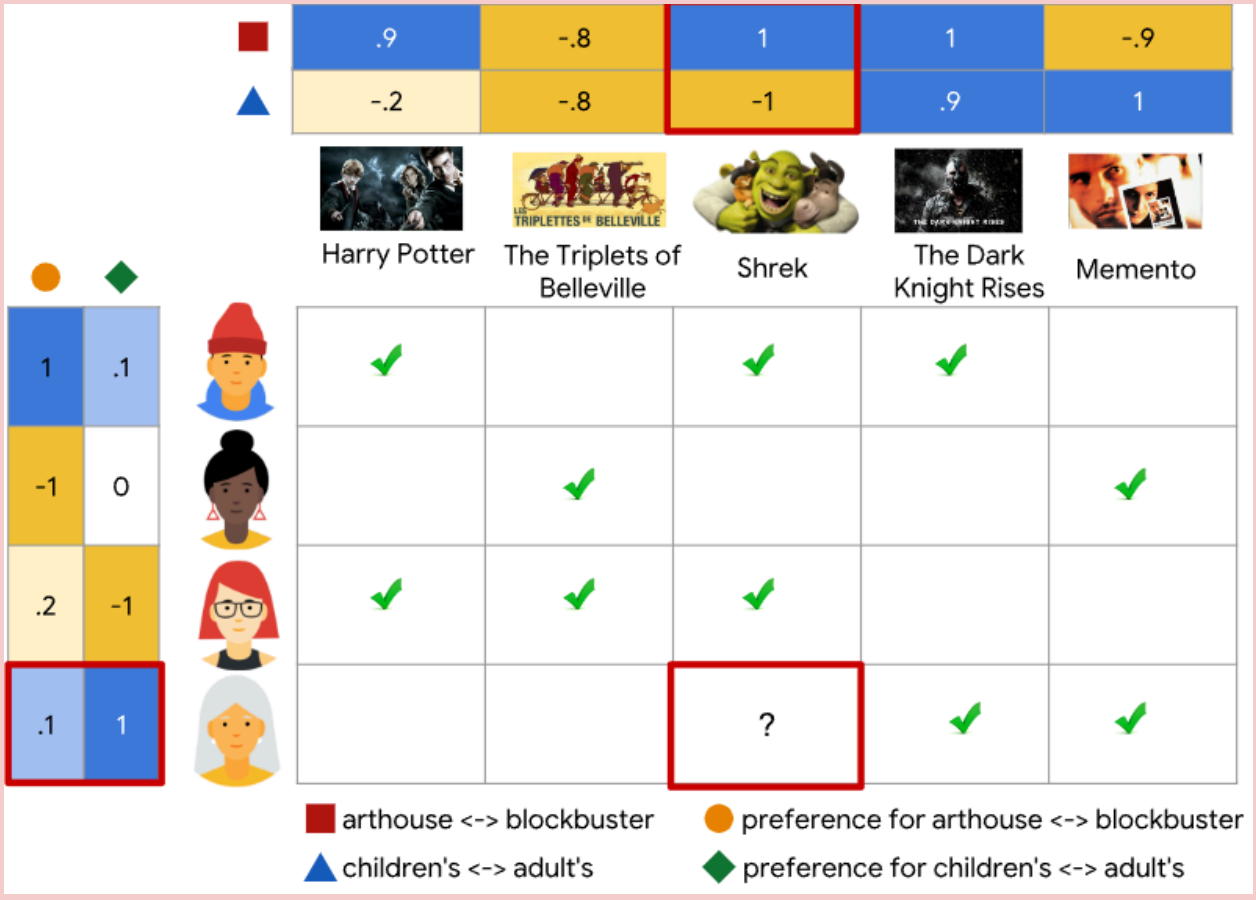
trong feature vector.



Hình 4: Biểu diễn người dùng không 2D embedding space.

Nguồn:
https://developers.google.com/machine-learning/recommendation/collaborative/basics

Chúng ta đặt người dùng vào cùng một không gian nhúng để làm rõ hơn feedback matrix: đối với mỗi cặp (user, item), chúng ta muốn dot product của user embedding và lượt item embedding gần bằng 1 khi người dùng đã xem phim, và ngược lại là 0 khi người dùng chưa xem phim. Từ các embedding space ta có thể rút ra embedding vector cho từng user và item, chúng được biểu diễn bằng qua ma trận trong hình sau:



Hình 5: Feedback matrix các dấu checkmark đại diện cho bộ phim mà user đã xem

Nguồn:
https://developers.google.com/machine-learning/recommendation/collaborative/basics

Sau khi tạo được feedback matrix, cùng với embedding của mỗi user và item, ta có thể tính toán sự giống nhau giữa các user đề từ đó nếu 2 user có mức độ tương đồng cao, hệ thống có thể đề xuất các bộ phim mà người dùng A chưa xem nhưng người dùng B đã xem và người dùng B thích bộ phim.

Để tính toán mức độ tương đồng có rất nhiều cách tính toán, một trong các cách tính toán phổ biến là cosine similarity:

$$cos(x,\ y)\ =\ \frac{x \cdot y}{||x||||y||}$$

Trước khi chuyển sang chi tiết cụ thể trong phương pháp, trong các lần trước ta đã đề cập về vấn đề sự thưa của ma trận rating của cặp (user, item). Đây luôn là vấn đề trong bài

toán Recommendation, ở đây để giải quyết vấn đề một cách thiên kiến (bias), chúng tôi đề xuất 2 phương pháp:

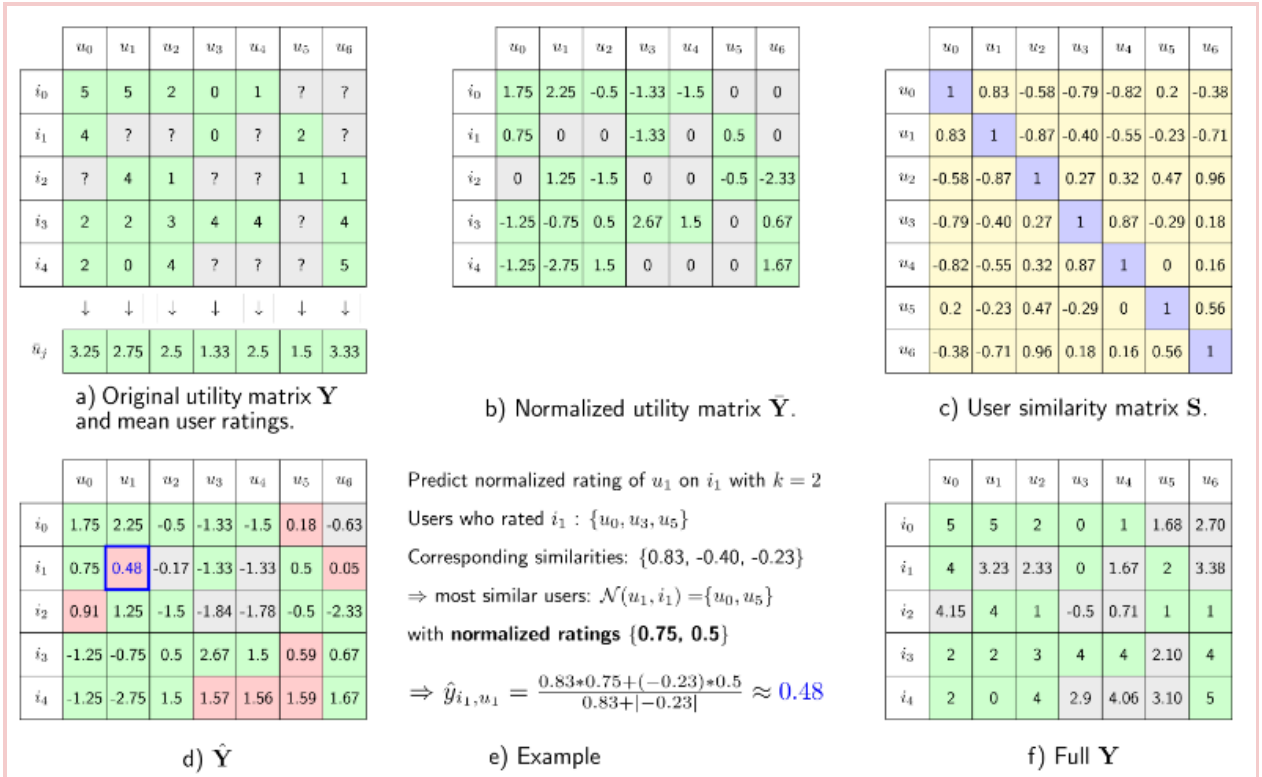
1. Thay các giá trị missing rating của cặp (user, item) bằng với giá trị trung bình của các rating mà user đó đã đánh giá. Nhưng khi chọn vấn đề này, có một vài vấn đề khó khăn.

    1. Nếu user đó đã đánh giá rất ít item, ví dụ: user rating 1 item và rating item đó 5 sao, khi đó ta nhận thấy việc giả sử này là không thực tế, và có thể dẫn đến rất nhiều sai sót.

    2. User đó không đánh giá item nào, khi đó ta sẽ không biết điền giá trị gì.

2. Ta sẽ điền vào tất cả cặp giá trị (user, item) missing = 2.5 hoặc 0 tương đương với giá trị trung tính (neutral).

Example: User-user Collaborative filtering



Hình 2: Ví dụ mô tả User-user Collaborative Filtering. a) Utility Matrix ban đầu. b) Utility Matrix đã được chuẩn hoá. c) User similarity matrix. d) Dự đoán các (normalized) *ratings* còn thiếu. e) Ví dụ về cách dự đoán normalized rating của u_1 cho i_1. f) Dự đoán các (denormalized) *ratings* còn thiếu.

Hình 6: Mô tả phương pháp User-user Collaborative Filtering.

Nguồn: https://machinelearningcoban.com/2017/05/24/collaborativefiltering/#-user-user-collaborative-filtering

a) Là ma trận ban đầu, hàng dưới cùng $\hat{u}_j$ biểu diễn giá trị trung bình của các cột.

b) Normalize matrix với $u_j = u_j - \hat{u}_j$.

c) Tính toán similarity giữa các user sử dụng **cosine similarity.**

d) Là ma trận dự đoán các giá trị missing rating, trong đó các giá trị missing rating được dự đoán bằng cách, chọn ra k (trong bài toán này chọn $k = 2$) user có mức độ similarity cao nhất đối với user muốn dự đoán và user đó đã rating item, sau đó ta dự đoán missing rating như trong hình **e).**

f) Là ma trận sau khi tính toán xong và thay $u_j = u_j + \hat{u}_j$

2.2.2.3 Phương pháp tiếp cận sử dụng MapReduce:

Bước 1: Chia dữ liệu thành các cặp khóa - giá trị:

Mapper: (key, value) −> (user_id, movie:rating_score) | map mỗi user với movie:rating_score tương ứng.
Reducer: (key, value) −> (user_id, (movie1: rating_score, movie2:rating_score, ..)).

Bước 2: Tạo ma trận xuất hiện đồng thời:

Mapper: Tạo ma trận xuất hiện đồng thời, ma trận xuất hiện đồng thời ở đây có thể hiểu là đối với mỗi cặp movie xuất hiện đồng thời hay đều được xem bởi user_id, thì với mỗi cặp movies đó ta tăng giá trị lên 1: (key, value) = (movie1:movie2, <1,1,1,... >)
Reducer: Tổng các giá count: (key, value) = (movie1:movie2, sum(<1,1,1...>)).

Bước 3: Normalize giá trị quan hệ:

Mapper: Định dạng lại output từ các cặp (movie1:movie2, sum()) , (movie1:movie3, sum()) −> (movie1, <movie2:relation, movie3: relation, ... >) (relation = sum).
Reducer: Normalize các giá relation thành relation/ sum(relation, ..) với mới key như user 1 trong mapper ở trên sẽ tồn tại nhiều giá trị relation, ta normalize nó bằng cách sum = tổng cách giá trị relation đó lại rồi lấy relation_n = relation/sum => (movie2, <movie1:realtion_n, ... >).

Bước 4: Tính toán giá trị user rating cho movie qua relation giữa các movies:

Bao gồm 2 quá trình Mapper:

Mapper1: Truyền output buớc 3 của Reducer trước cho Reducer.

Mapper2: Map từ input ban đầu (user_id, item_id, rating) thành (movie_id, user_id:rating).

Reducer: Nhận vào input bao gồm (movie2_id, <movie1_id=relation, .. item_id:rating, ... >) => (key, value) = (user:movie, relation*rating).

Bước 5: Tính tổng giá trị user rating cho movie:

Mapper : truyền output của Reducer trước trong bước 4 to Reducer.

Reducer: Tính tổng cách giá trị relation*rating,* (key, value) => (user:movie, sum( )) [trong đó sum( ) là giá trị rating dự đoán của user dành cho movie].

Bước 6: Lọc các giá trị rating của user dành cho movie nhỏ hơn threshold:

Mapper: Lọc các cặp (key, value) có value nhỏ hơn threshold *A*. Giữ lại các cặp giá tri (key, value) = (user:movie, rating) lớn hơn hoặc bằng threshold *A*.

Reducer: Reduce các cặp (key, value) = (user:movie, rating) thành (user, movie) các movie này sẽ được đề xuất cho user.

Chương 3 : Thử nghiệm và đánh giá

3.1. Bộ dữ liệu:

Sử dụng bộ dữ liệu MovieLens 2M gồm chứa 2.204.506 dữ liệu với các cột là User_Id, Movie_Name, Rating, Genre và Year.

3.2. Thuật toán Content-based Filtering:

3.2.1. Kết quả

```
+--------------------+--------------------+--------------------+
|              movie1|              movie2|          similarity|
+--------------------+--------------------+--------------------+
|            Chocolat|               Kenny|9.998180365258422E-4|
|       Dirty Dancing|               House|9.997434257065828E-4|
|          The Guilty|         Underground|9.996743847901846E-4|
|            Chocolat|     Kiss Me Goodbye|9.993493658537972E-4|
|Fanny and Alexand...|                Fear|9.992990437192216E-5|
|Bridge on the Riv...|           Dead Heat| 9.99286559551358E-4|
|            Chocolat|            Splendor|9.990127999092737E-4|
|       Dirty Dancing|                 Pan|9.987154880592904E-5|
|             My Girl|NeverEnding Story...|9.983789130123886E-4|
|            Chocolat|     Little Monsters|9.980520477442072E-4|
|Bad Education (La...|      My Blue Heaven| 9.97938881797442E-4|
|Barbarian Invasio...|            Iron Man|9.977944928804664E-6|
|               Akira|                   M| 9.97339344213557E-5|
|            Adelheid|                   M| 9.97124897464755E-4|
|       Amateur Teens|                   M| 9.97124897464755E-4|
|       I Am (Jestem)|                   M| 9.97124897464755E-4|
|                AWOL|                   M| 9.97124897464755E-4|
|    Adieu Philippine|                   M| 9.97124897464755E-4|
|              Homage|                   M| 9.97124897464755E-4|
|A Magnificent Hau...|                   M| 9.97124897464755E-4|
+--------------------+--------------------+--------------------+
only showing top 20 rows
```

3.2.2. Phân tích và đánh giá:

Thuật toán cho ra kết quả khá thấp vì có nhiều hạn chế

- Giới hạn về feature matrix: Thuật toán yêu cầu các item được mô tả chi tiết và cụ thể. Nếu thông tin mô tả của các item không đầy đủ hoặc thiếu chính xác, hệ thống sẽ gặp khó khăn trong việc gợi ý chính xác.

Ở đây, bộ dữ liệu thử nghiệm chỉ trích xuất thông tin thể loại phim mà không có các đặc trưng khác như đạo diễn, diễn viên… dẫn đến thiếu thông tin, thuật toán hoạt động không hiệu quả.

- Thiếu khả năng khám phá: Hệ thống chỉ dựa trên các sản phẩm mà người dùng đã tương tác từ trước. Điều này dẫn đến việc đưa ra gợi ý các sản phẩm mới mà người dùng chưa từng khám phá là rất khó. Hay nói cách khác, người dùng có thể bỏ lỡ các sản phẩm tiềm năng hoặc các sản phẩm mới ra mắt mà họ có thể quan tâm nhưng chưa từng tương tác trước đó.

- Vấn đề cold start của người dùng: Đối với người dùng mới hoặc người dùng chưa từng đánh giá bất kỳ sản phẩm nào, hệ thống sẽ ở tình trạng không có dữ liệu lịch sử tương tác của người dùng. Lúc này, Hệ thống không có đủ thông tin xây dựng hồ sơ sở thích dẫn đến việc gợi ý không chính xác hoặc không có gợi ý.

Mặc dù có nhiều hạn chế, thuật toán vẫn được sử dụng rộng rãi vì các ưu điểm sau:

- **Không phụ thuộc vào dữ liệu người dùng khác, cần ít dữ liệu huấn luyện:** Thuật toán chỉ dựa trên thông tin về sản phẩm và sở thích cá nhân của người dùng mà không cần thông tin của những người dùng khác.
- **Khả năng đề xuất cá nhân hóa:** Thuật toán có thể tạo ra các đề xuất cá nhân hóa dựa trên sở thích và hành vi của từng người dùng cụ thể.
- **Giải quyết vấn đề cold start sản phẩm mới:** Ngay cả khi có sản phẩm hoàn toàn mới, thuật toán vẫn có thể dựa vào các mô tả để đưa gợi ý đến người dùng.

3.3. Thuật toán Collaborative Filtering với ALS:

3.3.1. Kết quả:

```
+-------+--------+------+----------+
|User_Id|Movie_Id|Rating|prediction|
+-------+--------+------+----------+
|      1|    1423|   4.0| 3.6295834|
|      1|    1825|   4.0| 3.6823902|
|      1|    1954|   2.0| 3.3341224|
|      1|    5117|   3.0| 3.7898393|
|      1|    6906|   5.0| 3.7488658|
|      1|   11246|   2.5| 3.5693054|
|      1|   11825|   3.5| 3.7801185|
|      1|   13239|   5.0| 3.9083328|
|      1|   16482|   3.5| 3.9562864|
|      1|   18081|   5.0| 3.4491777|
|      1|   20391|   5.0| 3.9695456|
|      2|    1007|   4.0| 3.6244717|
|      2|    1105|   0.5| 3.3854523|
|      2|    2403|   3.0| 3.2399204|
|      2|    3163|   0.5| 3.3822029|
|      2|    3379|   4.5| 3.5900645|
|      2|    4015|   3.5| 3.3698997|
|      2|    5015|   2.0| 3.6297216|
|      2|    5432|   0.5| 3.1871965|
|      2|    5861|   3.5| 3.7668617|
+-------+--------+------+----------+
only showing top 20 rows

Root-mean-square error = 0.8212500716818497
+-------+--------------------+
```

3.3.2. Phân tích và đánh giá:

Ưu điểm:

Khả năng khám phá: Dựa trên sự tương đồng giữa các người dùng, thuật toán có thể đề xuất sản phẩm đa dạng, không tập trung vào một loạt hẹp. Đặc biệt, nó có khả năng đề xuất đến những sản phẩm mà người dùng chưa từng biết đến.

Không cần thông tin nội dung: Thuật toán không cần thông tin chi tiết về nội dung sản phẩm. Ứng dụng cao trong trường hợp các item như video, movie không có các thông tin chi tiết.

Hạn chế:

Vấn đề Sparsity: Ma trận tương tác giữa người dùng và sản phẩm rất thưa thớt, dẫn đến việc tìm kiếm sự tương đồng giữa các người dùng hoặc giữa các sản phẩm khó khăn và kém hiệu quả

Vấn đề Cold Start:

Người dùng mới: ALS gặp khó khăn trong việc gợi ý sản phẩm cho người dùng mới, do không có dữ liệu lịch sử về sở thích của họ.

Sản phẩm mới: Tương tự, các sản phẩm mới không có đánh giá hoặc tương tác từ người dùng sẽ khó được gợi ý bởi hệ thống.

Yêu cầu tài nguyên tính toán lớn: Quá trình huấn luyện ALS yêu cầu tài nguyên tính toán lớn, đặc biệt với các tập dữ liệu lớn. Mặc dù ALS có thể mở rộng tốt trên các hệ thống phân tán, nhưng khi kích thước dữ liệu tăng lên, chi phí tính toán và lưu trữ cũng tăng theo.

Chương 4: Kết luận và hướng phát triển

4.1. Kết luận:

Trong báo cáo này, chúng tôi đã trình bày các thuật toán gợi ý người dùng bao gồm Content-Based Filtering và Collaborative Filtering cũng như triển khai chúng với các kỹ thuật dữ liệu lớn.

Nhóm đã đạt được một số kết quả như sau:

- Hiểu tổng quan về Big Data, Hadoop, MapReduce, PySpark
- Hiểu chi tiết thuật toán gợi ý: Content-Based Filtering và Collaborative Filtering
- Triển khai ý tưởng thuật toán với PySpark
- Xây dựng demo cho đề tài
- Đánh giá demo

Tuy nhiên, vẫn còn một số hạn chế:

- Tập dữ liệu vẫn ở mức nhỏ

4.2. Hướng phát triển:

Tài liệu tham khảo