Graph based Recommendation System

Movie Recommendation System

Lipsa Mishra
MMATH (MDSAI)
University of Waterloo
Waterloo Ontario Canada
lmishra@uwaterloo.ca

ABSTRACT

In this project, I have implemented a movie recommendation system using User-Based Collaborative Filtering technique. The Movielens dataset is taken from the grouplens site. The recommendation system is implemented using GraphX, a Apache Spark's API for graphs and graph-parallel computation. The goal of this project is to implement collaborative filtering using GraphX to recommend top 'n' movies to the user. Movielens dataset is represented as a weighted bipartite graph where edges from user to movies are weighted by rating. A graph based resource allocation algorithm is followed to produce similarity measures between every pair of users. Then recommend the top 'n' movies to the user with the highest average rating.

1. INTRODUCTION

The amount of content in the internet like social media sites, shopping sites, movies, TV shows and so on are growing rapidly. Hence, it is very difficult for an user to decide what to select from these endless choices. We need a filtering system that would filter out the contents based on the user's preferences and choices and help the user to decide. Recommendation systems are built to solve this problem. It is used in many applications to suggest products, services, and information to potential customers. It is mostly embedded in the business web-based system. It collects the key information (purchases, ratings, browsing activities, and so on), from the users either from the sales transaction database or through a real-time system for making recommendations. The data collected are analyzed by the recommendation system's engine, which designs a customer preference model. The recommendations are then presented to the customers to support marketing and purchasing decision making. [1]

In this project, a User-Based Collaborative Filtering algorithm is implemented that uses existing data to give better suggestions to a user, based on other users having similar choices. We will be building a bipartite graph from the Movielens dataset to support graph traversal for collaborative filtering system.

KEYWORDS

Recommendation; collaborative filtering; similarity; graph representation; graph algorithms

2. DATA:

The dataset is obtained from the website http://www.grouplens.org/. GroupLens Research has collected and made available rating datasets from the MovieLens website (http://movielens.org). The datasets were collected over various periods of time,

depending on the size of the set [6]. In this project, the two main files that are used to build the recommendation system using collaborative filtering are movies.csv and ratings.csv.

Movies.csv:

The movies.csv has the following structure:

MovieID	title	genre
1	Toy StoryToy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children
9	Sudden Death (1995)	Action
10	GoldenEye (1995)	Action|Adventure|Thriller

Ratings.csv:

The ratings.csv has the following structure:

userId	movieId	ratings	timestamp
1	1	4	964982703
1	3	4	964981247
1	6	4	964982224
1	47	5	964983815
1	50	5	964982931
1	70	3	964982400
1	101	5	964980868
1	110	4	964982176
1	151	5	964984041
1	157	5	964984100

For testing purposes, the model is first implemented on the smaller dataset which consists of 100,000 ratings of 1700 movies given by 1000 users. Finally, the model is tested on a larger dataset which consists of 1000,000 ratings of 4000 movies given by 6000 users.

Below are the two datasets for the testing purpose:

Dataset size	No.of users	No.of Movies	No.of Ratings
100k Dataset	1000	1700	100000
1M Dataset	6000	4000	100000

2.1 REPRESENTATION OF THE DATA:

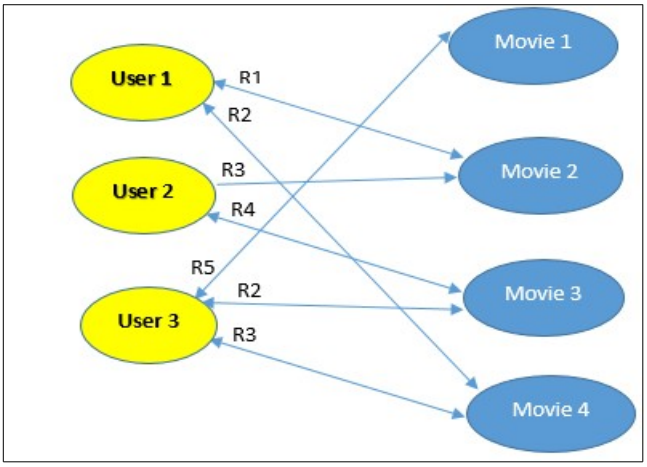
The movielens dataset is transformed into a bipartite graph to implement collaborative filtering. GraphX is used to transform the given dataset into a bipartite graph. GraphX extends the Spark RDD by introducing a new graph abstraction: a directed multigraph with properties attached to each vertex and edge [7]. The bipartite graph is divided into two distinct set of vertices (Users and Movies). There can be a link or edge (ratings) between the vertices. But the link between the same set of vertices is inadmissible. In this project, the vertices are Users and Movies and edge is the rating provided by the user to a given movie.[2] Below are the tabular and graphical representation of the data:

Tabular Representation of data:

User/ Movie	Movie 1	Movie 2	Movie 3	Movie 4
User 1		Rating 1		Rating 2
User 2		Rating 3	Rating 4	
User 3	Rating 5		Rating 2	Rating 3

The values in the above table represents that a rating is provided by the user for a given movie. The zero in the table represents that that User has not rated the given movie.

Graphical Representation Of Data:



An edge between a user and a movie in the above graph represents that a rating is provided by the user for a given movie. If there is no edge between the user and a given movie, then the user has not rated the given movie. [7]

The actual data set has 4000 movies and 6000 customer. So the bipartite graph matrix is of size (4000 * 6000).

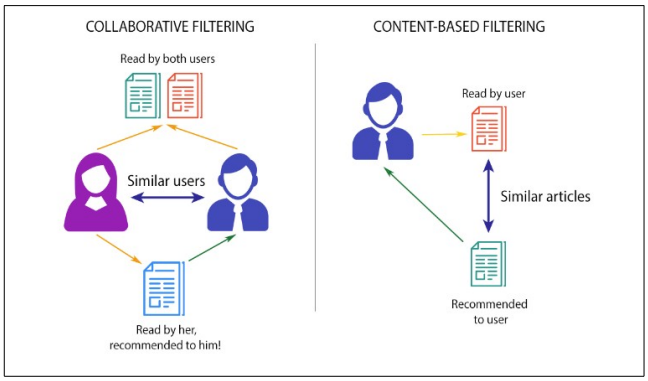
3. RELATED WORK:

Below are the **Filtering Methods** that are used to build a recommendation System:

A. <u>Content Based filtering</u>:

This filtering method is based on the description of an item and a profile of the user's preferred choices. In this system, keywords are used to describe the items. Besides, a user profile is built to state the type of item this user likes. In other words, the algorithms try to recommend products which are similar to the ones that a user has liked in the past. The idea of content based filtering is that if you like an item you will also like a 'similar' item. For example, when we are

recommending the same kind of item like a movie or song recommendation. This approach has its roots in information retrieval and information filtering research. But this filtering technique is not applicable to this project.



B. <u>Collaborative Based filtering:</u>

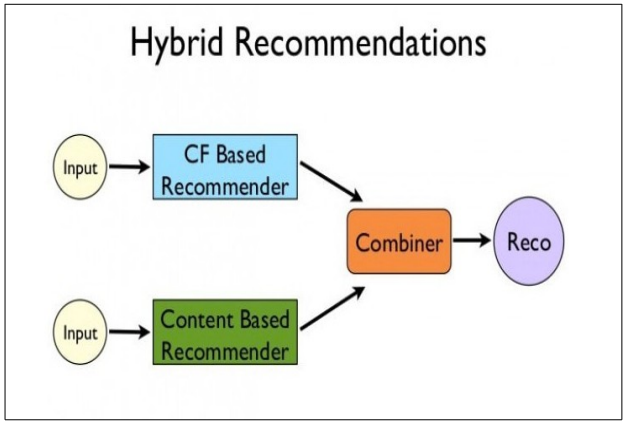
This filtering method is usually based on collecting and analyzing information based on user's behaviors, their activities or preferences. It then predicts items based on the similarity with other users. A key advantage of the collaborative filtering approach is that it does not rely on machine analyzable content and thus, it is capable of accurately recommending complex items such as movies without requiring an "understanding" of the item itself. Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past. In this project, collaborative filtering is used to recommend movies.

For example, if a person A likes item 1, 2, 3 and B like 2, 3, 4 then they have similar interests and A should like item 4 and B should like item 1. This is the basic concept behind collaborative filtering.

C. <u>Hybrid Based filtering:</u>

It is a combination of Content and Collaborative filtering. Hybrid approaches can be implemented by making content-based and collaborative-based predictions separately and then combining them.

For example, Netflix is a good example of the use of hybrid recommender systems. The website makes recommendations by comparing the watching and searching habits of similar users (i.e., collaborative filtering) as well as by offering movies that share characteristics with films that a user has rated highly (content-based filtering).



4. METHODOLOGY:

In this project, Collaborative filtering is implemented on the movielens dataset. It is basically of two types: (1) **User based collaborative filtering:** User-based collaborative recommendation aims to calculate some

similarity metric between all pairs of users and then predict a particular user U's rating for an item 'i' by collecting and processing the ratings of U's "neighborhood" (all the other users with high similarity as compared to user U) for item 'i'. [5]

(2) **Item based collaborative filtering:** Item-based collaborative recommendation seeks to calculate some similarity metric between all pairs of items and then predict a particular user U's rating for an item (i) by collecting and processing the ratings of item(i)'s "neighborhood" (all the other items with high similarity as compared to i that u has rated). [5]

In this project, user-based collaborative filtering is used. In this type of collaborative filtering approach, when the user tries to try to recommend a movie , then the algorithm tries to find other similar users who have watched almost the same movies as our current user. Here, cosine similarity metrics is used to find such similar users.

Cosine Similarity:

It is a method to measure the distance between two non zero vectors in an inner product space. For user-based collaborative filtering, the two users similarity is measured as the cosine of the angle between the two user's vectors. [4]

For example, if two persons b and c have similar movie preferences and we have ratings of two movies:

User/Movies	Iron Man (2008)	Pride and Prejudice (2005)
b	4	3
c	5	5

Each user's ratings can be represented in a separate vector:

$$\vec{b} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \qquad \vec{c} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

Cosine Similarity helps to measure the similarity between the two vectors b and c with the below formula:

$$similarity = \cos\theta = \frac{b.c}{\|b\|\|c\|}$$

$b.c \Rightarrow$ *Is the Dot product of the two vectors*

$\|b\|\|c\| \Rightarrow$ *Is the product of each vector's magnitude*

RECOMMENDATION ALGORITHM:

1. Find the list of movies 'M' rated by an user U1 in the movielens dataset.

2. Find the list of all the users who rated at least one of these movies M.

- Using aggregateMessage property of GraphX

- Returns a list of unique user Ids.

3. Find U2 users, who have the most similar choice in movies as user U1.

- Using Densevector() to convert the ratings of the users to a vector.

- Use Cosine Similarity to calculate the similarity between the two vectors

4. Find the list of movies M that users U1 has not rated yet but users U2 has rated them.

- Returns movieIds and count

5. Finally, recommend the top 'n' movies by the users U2 to the user U1 with the highest average rating.

- Returns score, movieId, title and genre

5. RESULTS:

For 1Million MovieLens dataset:

Below are the top 10 movies that are recommended for User U1 in the movielens dataset of 1000,000 ratings:

```
#-recommended: 10
---- recommended movies ----
(4.609) [2905] Sanjuro (1962)  Action Adventure
(4.561) [2019] Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954)  Action Drama
(4.555) [318] Shawshank Redemption, The (1994)  Drama
(4.525) [858] Godfather, The (1972)  Action Crime Drama
(4.521) [745] Close Shave, A (1995)  Animation Comedy Thriller
(4.517) [50] Usual Suspects, The (1995)  Crime Thriller
(4.510) [527] Schindler's List (1993)  Drama War
(4.508) [1148] Wrong Trousers, The (1993)  Animation Comedy
(4.491) [922] Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)  Film-Noir
(4.478) [1198] Raiders of the Lost Ark (1981)  Action Adventure
4.0percentage relevance of user is (40.000)
```

Figure 1: Top 10 recommended movies with the percentage of accuracy

6. EVALUATION:

As the recommendation system is built using User-based collaborative filtering technique, the evaluation metrics like MAE (Mean Absolute Error) and RMSE (Root Mean Square Error) cannot be used to evaluate the accuracy of the recommendation System. In this project, the recommendation system is evaluated based on the genre of the movies. The following steps are used to evaluate the recommendation system:

1. Find the list of genres of the movies for each user

2. Find the most preferred genre for each user

3. In the top 10 recommended movies for a user:

- Find the list of movies that has the same genre as the preferred genre for a user, calculated in step 2

- Find the total count of those movies

4. Find the accuracy percentage using the following formula:

Accuracy percentage (%) = (Total count calculated in step 3)/(Total count of Recommended movies) * 100

7. CONCLUSION:

In this experiment, the evaluation system used gives us 40 % of accuracy. The reason of the low accuracy is: few of the movies have multiple genres and our evaluation metric only considers the most preferred genre for the user. If multiple genres are considered, then this recommendation system would give a better accuracy. For testing, it would be more realistic to have data where we have information about what movies did the user choose after this recommendation. This way we can implement precision and recall metrics easily.

REFERENCES:

[1] Zhou, T., et al., Bipartite network projection and personal recommendation, Physical Review E, 2007. 76(046115)

[2] Shang, M., Fu, Y., Chen, D., Personal Recommendation Using Weighted BiPartite Graph Projection, Apperceiving Computing and Intelligence Analysis, 2008. ICACIA 2008. International Conference on , vol., no., pp.198,202, 13-15 Dec. 2008

[3] Sumedh Sawant (2013), Collaborative Filtering using Weighted BiPartite Graph Projection: A Recommendation System for Yelp http://snap.stanford.edu/class/cs224w-2013/projects2013/cs224w-038-final.pdf

[4] Bellogin, A., & Parapar, J. (2012). Using graph partitioning techniques for neighbor selection in user-based collaborative filtering. In Proceedings of the sixth ACM conference on Recommender systems (pp. 213–216).

[5] Pinela, Carlas (2017), Recommender Systems — User-Based and Item-Based Collaborative Filtering https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f

[6] GroupLens Movielens Dataset (2019) Social Computing Research at the University of Minnesota https://grouplens.org/datasets/movielens/

[7] ApacheSpark – GraphX- GraphX Programming Guide :http://spark.apache.org/docs/latest/graphx-programming-guide.html