INTRODUCTION TO DATA SCIENCE

CSL-487

MOVIE RECOMMENDATION USING MACHINE LEARNING

Group Member Name

- **Daniyal Qureshi (02-134191-055)**
- **Salman Siddiqui (02-134191-007)**
- **Faraz Ali (02-134191-114)**

COURSE INSTRUCTOR: **DR KASHIF**

LAB INSTRUCTOR: **SOOMAL FATIMA**

BAHRIA UNIVERSITY, KARACHI, PAKISTAN

DEPARTMENT OF COMPUTER SCIENCE

ABSTRACT:

Recommender systems have become ubiquitous in our lives. Yet, Currently, they are far from optimal. In this project, we attempt to understand the different kinds of recommendation systems and Compare their performance on the MovieLens dataset. We attempt to build a scalable model to perform this analysis. We start by preparing and comparing the various models on a smaller dataset of 100,000 ratings. Then, we try to scale the algorithm so that it is able to handle 20 million ratings by using Apache Spark. We find that for the smaller dataset, using user-based collaborative filtering results in the lowest Mean Squared Error on our dataset.

INTRODUCTION:

A recommendation system is a type of information filtering system which attempts to predict the preferences of a user, and make suggests based on these preferences.

There are a wide variety of applications for recommendation systems. These have become increasingly popular over the last few years and are now utilized in most online platforms that we use. The content of such platforms varies from movies, music, books and videos, to friends and stories on social media platforms, to products on e-commerce websites, to people on professional and dating websites, to search results returned on Google.

Often, these systems are able to collect information about a users choices, and can use this information to improve their suggestions in the future. For example, Facebook can monitor your interaction with various stories on your feed in order to learn what types of stories appeal to you. Sometimes, the recommender systems can make improvements based on the activities of a large number of people. For example, if Amazon observes that a large number of customers who buy the latest Apple Macbook also buy a USB-C-to-USB Adapter, they can recommend the Adapter to a new user who has just added a Macbook to his cart.Due to the advances in recommender systems, users constantly expect good recommendations. They have a low threshold for services that are not able to make appropriate suggestions. If a music

streaming app is not able to predict and play music that the user likes, then the user will simply stop using it. This has led to a high emphasis by tech companies on improving their recommendation

systems. However, the problem is more complex than it seems. Every user has different preferences and likes. In addition, even the taste of a single user can vary depending on a large number of factors, such as mood, season, or type of activity the user is doing. For example, the type of music one would like to hear while exercising differs greatly from the type of music he'd listen to when cooking dinner.

It turns out that there are (mostly) three ways to build a recommendation engine:

1. Popularity based recommendation engine
2. Content based recommendation engine
3. Collaborative filtering based recommendation engine

Now you might be thinking "That's interesting. But, what are the differences between these recommendation engines?". Let me help you out with that.

Popularity based recommendation engine:

Perhaps, this is the simplest kind of recommendation engine that you will come across. The trending list you see in YouTube or Netflix is based on this algorithm. It keeps a track of view counts for each movie/video and then lists movies based on views in descending order(highest view count to lowest view count). Pretty simple but, effective. Right?

Content based recommendation engine:

This type of recommendation systems, takes in a movie that a user currently likes as input. Then it analyzes the contents (storyline, genre, cast, director etc.) of the movie to find out other movies which have similar content. Then it ranks similar movies according to their similarity scores and recommends the most relevant movies to the user.

Collaborative filtering based recommendation engine:

This algorithm at first tries to find similar users based on their activities and preferences (for example, both the users watch same type of movies or movies directed by the same director). Now, between these users(say, A and B) if user A

has seen a movie that user B has not seen yet, then that movie gets recommended to user B and vice-versa. In other words, the recommendations get filtered based on the collaboration between similar user's preferences (thus, the name "Collaborative Filtering"). One typical application of this algorithm can be seen in the Amazon e-commerce platform, where you get to see the "Customers who viewed this item also viewed" and "Customers who bought this item also bought" list.

Content Based Recommendations:

Content Based Recommendation algorithm takes into account the likes and dislikes of the user and generates a User Profile. For generating a user profile, we take into account the item profiles(vector describing an item) and their corresponding user rating. The user profile is the weighted sum of the item profiles with weights being the ratings user rated. Once the user profile is generated, we calculate the similarity of the user profile with all the items in the dataset, which is calculated using cosine similarity between the user profile and item profile. Advantages of Content Based approach is that data of other users is not required and the recommender engine can recommend new items which are not rated currently, but the recommender algorithm doesn't recommend the items outside the category of items the user has rated.

DATASET:

We used the MovieLens movie ratings dataset for our experiments1 .The experiments are conducted on two versions of the dataset. The first version consists of 100004 ratings by 671 users across 9125. movies. The ratings allowed at intervals of 0.5 on a 5-point scale, starting from 0.5 and going to 5. All selected users had rated at least

20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided. The dataset has additional information about the movies in the form of genre and tags, however we use only the ratings given by the users to the movies and ignore the other information for the collaborative filtering techniques. For the content filtering portion, information was scrapped from the IMDB website for the corresponding movie. The links to the IMDB page for each movie is provided in a separate file.

IMPLEMENTATION:

5.1 Baseline methods

We try out the following simple baseline methods to give us an idea of the performance to expect from the

5.1.1 Global Average. The global average technique serves as a simple baseline technique. The average rating for all users across all movies is computed. This global average serves as a prediction for all the missing entries in the ratings matrix.

5.1.2 User average. All users exhibit varying rating behaviors. Some users are lenient in their ratings, whereas some are very stringent giving lower ratings to almost all movies. This user bias needs to be incorporated into the rating predictions. We compute the average rating for each user. The average rating of the user is then used as the prediction for each missing rating entry for that particular user. This method can be expected to perform slightly better than the global average since it takes into account the rating behavior of the users into account.

5.1.3 Movie average. Some movies are rated highly by almost

all users whereas some movies receive poor ratings from everyone. Another simple baseline which can be expected to perform slightly better than the global average is the movie average method. In this technique, each missing rating entry for a movie j is assigned the average rating for the movie j.

5.1.4 Adjusted Average. This simple method tries to incorporate some information about the user i and the movie j when making a prediction for the entry rij . We predict a missing entry for user i and movie j, by assigning it the global average value adjusted for the user bias and movie bias. The adjusted average rating is given by the formula below.

rij = дlobal_avд + (u_avд(i) − дij) + (m_avд(j) − дij)

The user bias is given by the difference between the average user rating and the global average rating. The movie bias is given by the difference between the average movie rating and the global average rating. Consider the following example which demonstrates the working of the adjusted average method. Let the global average rating be 3.7. The user A has an average rating of 4.1. Thus the bias of the user is (4.1 - 3.7) I.e the user rates 0.4 stars more than the global average. The movie Fast and Furious has an average rating of 3.1 stars. Thus the bias for the movie is -0.6. the adjusted average method will predict that user A will give the movie Fast and Furious a rating of 3.7 + 0.4 - 0.6 = 3.5

RESULTS AND DISCUSSION

We implement the nearest neighbors and latent factor methods for

Collaborative filtering. The smaller version of the dataset can be processed by using dense matrices and non-vectorized operations. However, if we try to use the simple implementations on the larger 20 million dataset, the code breaks. If we store the ratings matrix for the 20 million dataset in a dense format, it would take up around $140000 \times 27000 \times 8 = 28$GB of memory, assuming 8 bytes per matrix entry. Thus we need to use sparse matrix representations to be able to handle the bigger dataset effectively.

POSSIBLE FUTURE WORK:

There are plenty of way to expand on the work done in this project. Firstly, the content based method can be expanded to include more criteria to help categorize the movies. The most obvious ideas is to add features to suggest movies with common actors, directors

or writers. In addition, movies released within the same time period could also receive a boost in likelihood for recommendation. Similarly, the movies total gross could be used to identify a users taste in terms of whether he/she prefers large release blockbusters, or smaller indie films. However, the above ideas may lead to overfitting, given that a users taste can be highly varied, and we only have a guarantee that 20 movies (less than 0.2%) have been reviewed by the user.In addition, we could try to develop hybrid methods that try to combine the advantages of both content-based methods and collaborative filtering into one recommendation system.

\\Code:

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
###### helper functions.#######
def get_title_from_index(index):
        return df[df.index == index]["title"].values[0]

def get_index_from_title(title):
        return df[df.title == title]["index"].values[0]
##################################################

##Step 1: Read CSV File
df = pd.read_csv("movie_dataset.csv")
# print (df.columns)

##Step 2: Select Features
```

```
features = ['keywords','cast','genres','director']

##Step 3: Create a column in DF which combines all selected features
for feature in features:
    df[feature] = df[feature].fillna('')

def combine_features(row):
    try:
        return row['keywords'] +" "+row['cast']+" "+row["genres"]+" "+row["director"]
    except:
        print ("Error:", row)

df["combined_features"] = df.apply(combine_features,axis=1)



##Step 4: Create count matrix
cv = CountVectorizer()

count_matrix = cv.fit_transform(df["combined_features"])

##Step 5: Compute the Cosine Similarity based on the count_matrix
cosine_sim = cosine_similarity(count_matrix)
ask = input ("Input a movie from the list:  ")
movie_user_likes = ask

## Step 6: Get index of this movie from its title
```

```
movie_index = get_index_from_title(movie_user_likes)

similar_movies =  list(enumerate(cosine_sim[movie_index]))

## Step 7: Get a list of similar movies in descending order of similarity score
sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)

## Step 8: Print titles of first 50 movies
i=0
for element in sorted_similar_movies:
        print (get_title_from_index(element[0]))
        i=i+1
        if i>50:
            break
```

SCREENSHOT:

```
Input a movie from the list:  Titanic
```

Result:

```
Titanic
Revolutionary Road
Me You and Five Bucks
All the King's Men
The Day the Earth Stood Still
Almost Famous
The Beach
Cheri
Little Black Book
Vanilla Sky
The Bridge of San Luis Rey
In the Valley of Elah
The Aviator
Fabled
The Great Gatsby
The Life of David Gale
The Reader
Gossip
Pink Narcissus
Blood Diamond
The Departed
The Box
Eternal Sunshine of the Spotless Mind
The Yards
Becoming Jane
The Remains of the Day
Romance & Cigarettes
```

REFERENCES:

[1] A Survey of Collaborative Filtering Techniques; Su et al; https://www.hindawi.com/journals/aai/2009/421425/

[2] Google News Personalization: Scalable Online Collaborative Filtering; Das et al; https://www2007.org/papers/paper570.pdf

[3] Intro to Recommender Systems: Collaborative Filtering; http://blog.ethanrosenthal.com/2015/11/02/intro-to-collaborative-filtering/

[4] Collaborative Filtering Recommender Systems; Stanford Student project; http://cs229.stanford.edu/proj2014/Rahul%20Makhijani,%20Saleh%20Samaneh,%20Megh%20Mehta,%20Collaborative%20Filtering%20Recommender%20Systems.pdf

[5] MMDS course slides; Jeffrey Ullman; http://infolab.stanford.edu/ ullman/mmds/ch9.pdf

[6] Recommending items to more than a billion people; Kabiljo et al; https://code.facebook.com/posts/861999383875667/recommending-items-to-more-than-a-billion-people/