Movie Recommendation System

Harsh Gahlot
Dept. of Computer Science
Indian Institute of Technology Mandi

Abbas Husain
Dept. of Computer Science
Indian Institute of Technology Mandi

Ravi Pratap Yadav
Dept. of Computer Science
Indian Institute of Technology Mandi

Sucheta Panda
Dept. of Computer Science
Indian Institute of Technology Mandi

Gaurav Shukla
Dept. of Computer Science
Indian Institute of Technology Mandi

***Abstract* —**

The requirement to extract useful data from enormous amounts of raw data to power business solutions has grown as corporate needs have become more urgent.

The same is true for digital recommendation systems, which are commonplace in consumer-facing businesses like those for books, music, apparel, movies, news articles, locations, and utilities. These systems gather data from users to enhance suggestions in the future. Through the use of collaborative filtering algorithms, this work seeks to describe the implementation of a movie recommendation system.

By effectively identifying patterns and knowledge, big data is utilized to manage data efficiency. However, there are several users with various tastes, therefore with this enormous amount of data, we must provide the user with relevant content that will aid in making the best choice. We must rely on an automated machine to complete these tasks, which helps to lessen the work required from each human. This is where the recommendation system enters the picture. Recommendation systems are currently widely used in our daily lives for things like movies, news, books, shopping, and music. Additionally, huge corporations like Netflix, Amazon, LinkedIn, and YouTube use recommendation systems. To improve the accuracy and precision of the advice for a person based on their needs, numerous technologies and algorithms are applied. The three primary classifications of recommendation systems are content-based systems, collaborative filtering systems, and hybrid recommendation systems.

Keywords* — *Recommendation System, User Based Recommendation, Item Based Recommendation, ALS Algorithm

1 Introduction

A recommendation system, sometimes known as a recommendation engine, is a paradigm for information filtering that aims to anticipate user preferences and offer suggestions in accordance with these preferences. These technologies are now widely used in a variety of industries, including those that deal with utilities, books, music, movies, television, apparel, and restaurants. These systems gather data on a user's preferences and behavior, which they then employ to enhance their recommendations in the future.

Recommendation systems are being used by a lot of businesses to improve customer interaction and the purchasing experience. The most significant advantages of recommendation systems are client happiness and income. Customers frequently look at the suggestions made based on their prior transaction because they believe they will find better selections there. The consumer will be happy with their purchase if these recommendations are tailored to their needs. As a result, the client would utilize this programme once more. Due to the significant money gained by clients who use these applications frequently, many e-commerce businesses are turning to improve their recommendation engines.

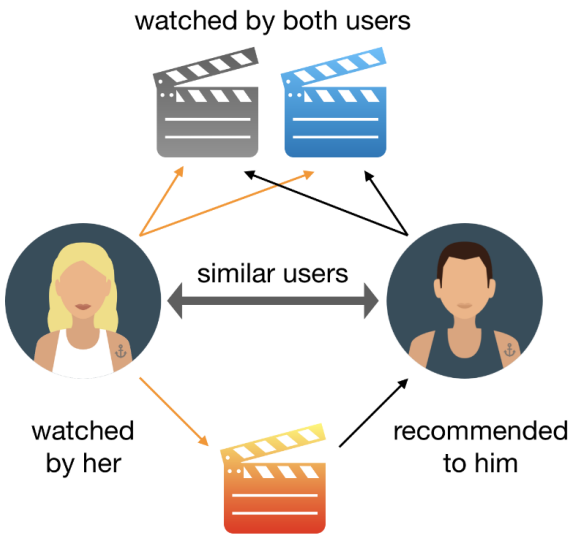
Although recommendation systems are ubiquitous, it can be difficult to create systems that offer useful and relevant ideas. Every individual has unique tastes and interests. Furthermore, a user's selection is influenced by a variety of factors, including their mood, the occasion, the reason behind their purchase, etc. The user is likely to cease using a website or app if it cannot forecast their preferences and offer recommendations that are suited for them. As a result, businesses must constantly enhance their recommendation systems.

The purpose of this paper is to create a movie recommendation system that offers ideas to the user based on previous movie ratings provided by different users. We used cooperative filtering strategies to create this system.

2 Overview

A. Recommendation System

We are all familiar with the recommendation engine where the web site tells us along the line of if you like that product then you probably like this product, we see this type of recommendation in many businesses, what we are going to see is one way that the company generates these recommendations, when we purchase something online, or watch a movie we are often given a chance to rate that product or the movie on a scale of 1 to 5. Based on our feedback from these types of rating system companies can learn a lot about us and our preferences and can make predictions about our taste and offer recommendations based on rating from users similar to us. Let us see an example of a movie recommendation system. Let us say we have a customer A who likes ... dislikes ..Then we can find another customer who likes ... dislikes ... Because they have such similar tastes, if one of them gave a high rating to a movie that the other customer has not yet seen we can infer that the other customer is likely to enjoy that movie as well. Based on that logic we can offer that as a recommendation.

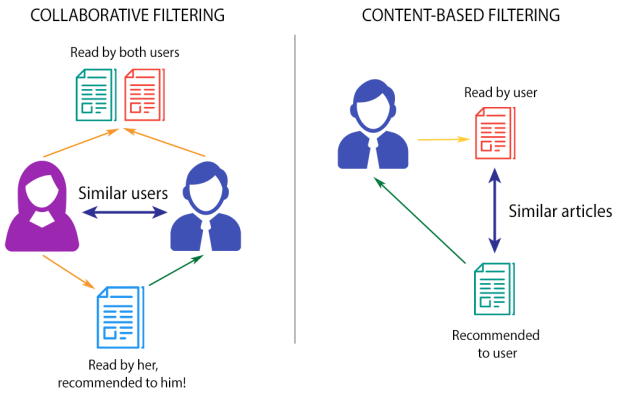


We give a brief overview of the various kinds of recommendation systems. Recommendation systems can be divided into three categories: collaborative filtering, content-based filtering, and hybrid systems.

Collaborative Filtering: Systems that use collaborative filtering examine user behavior and preferences to forecast what they will want based on similarities to other users. Collaborative filtering systems come in two varieties: user-based recommender and item-based recommender.

Content-Based Filtering: When making suggestions, content-based systems take into account both the user's preferences and the description and features of an item.

Hybrid System: Hybrid recommendation systems combine content-based and collaborative filtering techniques. These kinds of systems undertake collaborative and content-based predictions independently before combining the findings to offer recommendations.



B. PySpark

Apache Spark is an open source, distributed computing platform and collection of tools for real-time, massive data processing, and PySpark is its Python API.

In essence, Apache Spark is a computational engine that handles big data sets by processing them concurrently and in batches. PySpark was created to facilitate the integration of Python and Spark, which is written in Scala. PySpark makes use of the Py4j library to assist you in interacting with Resilient Distributed Datasets (RDDs), in addition to offering a Spark API.

The Spark dataframe is the main type of data utilized in PySpark. This object, which functions similarly to dataframes in R and Pandas, can be thought of as a table dispersed throughout a cluster. You must operate on Spark dataframes rather than other Python data types if you want to use PySpark for distributed computation.

Operations in PySpark are postponed until a result is truly needed in the pipeline. If we want to import a data set from Amazon S3 and apply a lot of transformations to the dataframe, for instance, we can specify these actions, but they won't take effect right away. Instead, a graph of transformations is kept and applied as a single pipeline operation once the data are truly needed, such as when publishing the results back to S3. By avoiding loading the entire dataframe into memory, this method allows for more efficient processing across a cluster of workstations.

3 Collaborative Filtering Model

We suggest a collaborative approach for suggesting movies that relies heavily on user ratings to make recommendations. Utilizing both user-based and item-based collaborative filtering techniques, we built this system using PySpark and the similarity/correlation between objects. We go into detail about the two collaborative filtering approaches in this section.

A. User-Based Filtering

In the subject of building tailored systems, user preferences are crucial. This method makes the assumption that, when looked at historically, the user's preferences are not random. Users first rate various catalog items on a scale of 1 to 5. Both implicit and explicit ratings are possible. When a person expressly ranks an item on a scale or gives it a thumbs-up or thumbs-down, the rating is known as an explicit rating. Explicit ratings are frequently difficult to collect because not all users are keen on leaving comments. We collect implicit ratings based on their actions in various instances. For instance, a user's repeated purchases of a product show a favorable preference. In relation to movie systems, we can infer that a user has some likeability to the film if they watch the entire thing. Be aware that there are no precise guidelines for establishing implicit ratings.

Next, we identify a specific number of nearest neighbors for each user. Using the Pearson Connection algorithm, we determine the correlation between user ratings. To recommend items to consumers, it is assumed that if two users' ratings are significantly connected, they must value similar goods and services.

B. Item-based filtering

Item-based filtering, as opposed to user-based filtering, places more emphasis on the similarities between the goods that users prefer than on the individuals themselves. It is calculated in advance which things are the most comparable. The user is then given recommendations for goods that are most comparable to the target item.

4 ALS ALGORITHM

Another matrix factorization approach that runs in parallel is called Alternating Least Square (ALS). ALS is a collaborative filtering algorithm that is built for large-scale problems and is implemented in Apache Spark ML. With its simplicity and capacity to expand to very big datasets, ALS does a decent job of addressing the scalability and sparsity of the Ratings data.Matrix factorization: what is it? A collection of mathematical operations for matrices in linear algebra is called matrix factorization. Matrix factorization, to be precise, is the factorization of a matrix into a product of matrices. When it comes to collaborative filtering, matrix factorization techniques function by breaking down the user-item interaction matrix into the sum of two rectangular matrices with lesser dimensions. The user matrix is one example of a matrix where the rows correspond to users and the columns to latent factors.
How does matrix factorization solve our problem?

- In order to forecast more accurate personalized movie ratings for users, the algorithm learns to factorize the rating matrix into user and movie representation.
- Matrix factorization makes it possible for lesser-known films to have rich latent representations on par with those of popular films, which enhances the recommendation capabilities of recommenders.The anticipated rating user u will give item I is calculated using the sparse user-item interaction matrix as following Diagram 1

$$\tilde{r}_{ui} = \sum_{f=0}^{nfactors} H_{u,f} W_{f,i}$$

Diagram 1

The anticipated rating user u will give item I is calculated using the sparse user-item interaction matrix as following

$$\arg\min_{H,W} \|R - \tilde{R}\|_{\mathrm{F}} + \alpha\|H\| + \beta\|W\|$$

where H is user matrix, W is item matrix

Diagram 2.

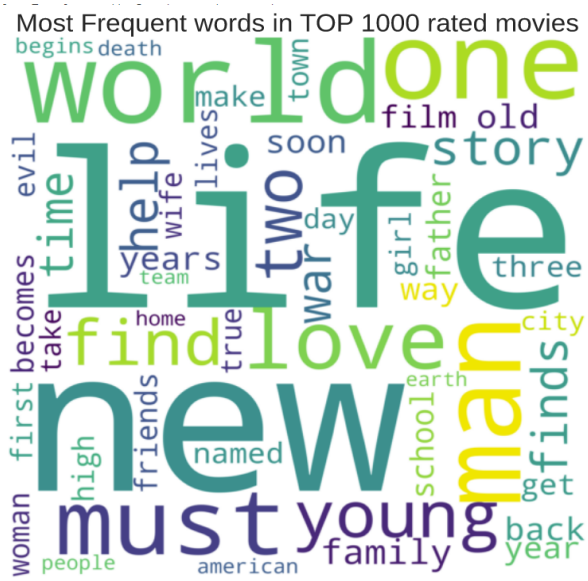
The following hyper-parameters are crucial for Alternating Least Square (ALS):

- maxIter: the maximum number of iterations to run (defaults to 10)
- rank: the number of latent factors in the model (defaults to 10)
- regParam: the regularization parameter in ALS (defaults to 1.0)

Different from other training methods, ALS minimizes two loss functions in an alternating fashion. First, it runs gradient descent with the item matrix while holding the user matrix fixed, and then it does the same with the item matrix while holding the user matrix fixed.The next paper will discuss the intrinsic and extrinsic elements that have been effectively demonstrated to promote and secure QoL in ALS in accordance with the four ethical standards. Beneficence, non-maleficence, autonomy, and justice are viewed as essential components of patient-centered care.Depending on the underlying etiology and the time of clinical onset, ALS can be described in a variety of ways. The two kinds of ALS, familial and sporadic, differ in terms of the underlying cause. Spinal ALS and bulbar ALS are the two terminologies used to categorize the disease in terms of its clinical start.Hold-Out and Cross-Validation are two techniques used in data science to assess models.

Natural Language Processing (NLP)

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can. It is one of the most popular subcategories of AI. It is the study of how machines analyze natural languages and produce meaningful information about the text. First, we do training on how to read and understand the text. We trained our model using NLP. According to ratings of the user, the most frequent words in top 1000 rated movies are observed. Then if anyone wants to get the recommendation, he/she will find the movie among these 1000 most frequent words.

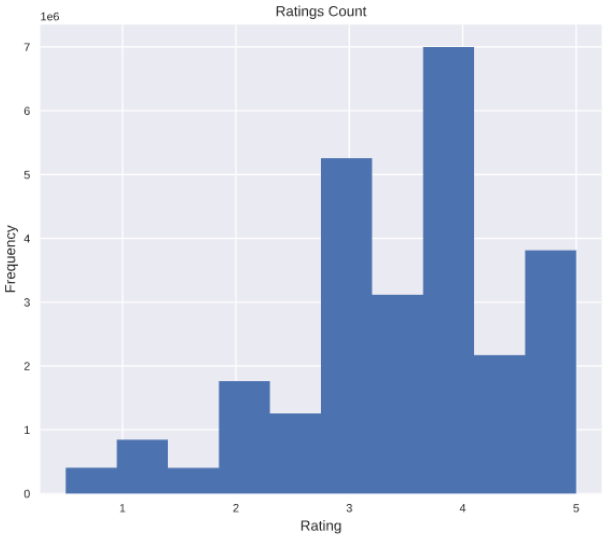


As we can see from the figure, the most frequent words used in the movies are having a larger size as compared to the less frequently used words in the movies. For example, “life” having the largest size is the most frequently used word in the top 1000 rated movies and if anyone wants recommendation , then a movie having the word “life” can be recommended followed by other frequently used words.

5 Results

Given how prevalent recommendation systems are now, there are several ways to create and enhance them.
They are available in internet stores, streaming music and movie services.
We created an EDA on The Movies Dataset in this Kernel and obtained the information needed to create a recommendation system using Alternating Least Square (ALS). With this system, it will be feasible to create a user-based system or an item-based system.

We used Pandas Dataframes for this project, Altair for visuals, BeautifulText for printable text, and PySpark for ALS. Basically we have 7 files of different data in our dataset. From the data in the metadata file displayed, we utilized it to create a recommendation engine based on user ratings because we could observe that just ratings take a lot of memory. The dataset contains important data like userId, movieId, and ratings.
The ratings dataset does not include any missing values, therefore no special handling, such as data imputation or removing NA rows, was required.
Exploratory Data Analysis was performed using the additional datasets. According to our preliminary data, the majority of films received ratings of 4, on a scale from 1 to 5. Less movies (in relation to the entire dataset) were given low ratings. Another interesting fact was that none of the highest-rated movies have a rating higher than 9.0.



We examined the 1,000 most highly rated movies and the terms that frequently appear in their synopses. The next step was using Natural Language Processing (NLP) with the NLTK module to convert everything to lower case, add word tokens, eliminate stopwords, and create a Word Cloud. After setting up the spark environment, the next step in the process was to split the dataset into 70% train data and remaining 30% test data. And now the model was trained on the training data. The evaluation of the model was done by applying the model on the test dataset using Mean Absolute Error (MAE).

Finally, the User Based Recommendation System produced the Best recommendation for each user, with a test accuracy of 65.75%.

Best recommendation for each user (User Based Recommendation System). The movieId, the first element in the recommendations vector, is the same as the ratings dataframe.

```
model.recommendForAllUsers(1).show(5)
```

```
+------+--------------------+
|userId|     recommendations|
+------+--------------------+
|     1|[{101862, 6.585939}]|
|     2|[{164937, 4.963108}]|
|     3| [{158832, 4.19634}]|
|     4|[{164937, 5.639908}]|
|     5|[{164937, 6.23275...|
+------+--------------------+
only showing top 5 rows
```

Most recommended user for each movie (Item Based Recommendation System). Again, the movieId is the same as the ratings dataframe.

```
[ ]  model.recommendForAllItems(1).show(5)
```

```
+-------+--------------------+
|movieId|     recommendations|
+-------+--------------------+
|      1|[{172953, 5.41259...|
|      2|[{212857, 5.07832...|
|      3|[{179759, 5.21454...|
|      4|[{41755, 4.6923423}]|
|      5|[{179759, 5.21794...|
+-------+--------------------+
only showing top 5 rows
```

References

[1] Ching-Seh (Mike) Wu, Deepti Garg, Unnathi Bhandary, Movie Recommendation System Using Collaborative Filtering, vol. 47, pp. 98–115, 2015.

[2] A. V. Dev and A. Mohan , "Recommendation system for big data applications based on set similarity of user preferences," 2016 International Conference on Next Generation Intelligent Systems (ICNGIS), Kottayam, 2016, pp. 1-6. doi: 1O.1109/INGIS.2016.7854058

[3] Kumar, Manoj & Yadav, D.K. & Singh, Ankur & Kr, Vijay, "A Movie Recommender System: MOVREC ", 2015 International Journal of Computer Applications. 124. 7-11. 10.5120 /ijca2015904111.

[4] . Sarwar, G. Karypis, 1. Konstan , 1. Riedl, "Item-based collaborative filtering recommendation algorithms", Proceedings of the 10th international conference on World Wide Web, pp. 285-295, 2001.

[5] J. Zhang, Y. Wang, Z. Yuan and Q. Jin, "Personalized real-time movie recommendation system: Practical prototype and evaluation", April 2020Ut fermentum a magna ut eleifend. Integer convallis suscipit ante eu varius.

[6] M. Ahmed, M. T. Imtiaz and R. Khan, "Movie recommendation system using clustering and pattern recognition network," 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), 2018

[7] Bhalse, N., & Thakur, R. (2021). Algorithm for movie recommendation system using collaborative filtering. Materials Today: Proceedings.

Github Link : https://github.com/husainam/movieRecommendation