|  |  |
| --- | --- |
| A picture containing drawing  Description automatically generated  Bana 7047 Final Project  BANA-7047 DATA MINING II | Abstract  Companies like Amazon, Netflix, LinkedIn, and Pandora leverage recommender systems to help users discover new and relevant items (products, videos, jobs, music), creating a delightful user experience while driving incremental revenue. In this Project, three types of machine learning algorithms viz Content Based and Collaborative Filtering Methods and Singular Value decomposition, are built to generate a recommendation system based on MovieLens Dataset  AUTHORS  Joshi, Jagruti  Kumari, Priya  Sahare, Pooja |

Contents

[**Goal** 3](#_Toc38831302)

[**Background** 3](#_Toc38831303)

[**Approach** 3](#_Toc38831304)

[**Major Results** 3](#_Toc38831305)

[**Conclusion** 4](#_Toc38831306)

[**Exploratory Data Analysis** 4](#_Toc38831307)

[Exploring Genre 4](#_Toc38831308)

[Exploring Ratings 5](#_Toc38831309)

[Exploring Tags 5](#_Toc38831310)

[**Content Based Filtering** 6](#_Toc38831311)

[Summary 6](#_Toc38831312)

[Math behind similarity calculation 7](#_Toc38831313)

[Model Result 8](#_Toc38831314)

[Pros of Content based filtering 9](#_Toc38831315)

[Cons of Content based filtering 9](#_Toc38831316)

[**Collaborative Filtering** 10](#_Toc38831317)

[Summary 10](#_Toc38831318)

[Math behind similarity calculation 10](#_Toc38831319)

[**User-User Collaborative Filtering** 11](#_Toc38831320)

[Model Result 12](#_Toc38831321)

[**Item-Item Collaborative Filtering** 12](#_Toc38831322)

[Model Result 13](#_Toc38831323)

[Pros of Collaborative filtering 13](#_Toc38831324)

[Cons of Collaborative filtering 13](#_Toc38831325)

[**Singular Value Decomposition** 13](#_Toc38831326)

[Summary 13](#_Toc38831327)

[Math behind decomposition 15](#_Toc38831328)

[Model Results 16](#_Toc38831329)

[Pros of SVD 16](#_Toc38831330)

[Cons of SVD 17](#_Toc38831331)

[**References** 17](#_Toc38831332)

# **Goal**

To build a recommendation system model to recommend movies and predict ratings based on user ratings and movie tags.

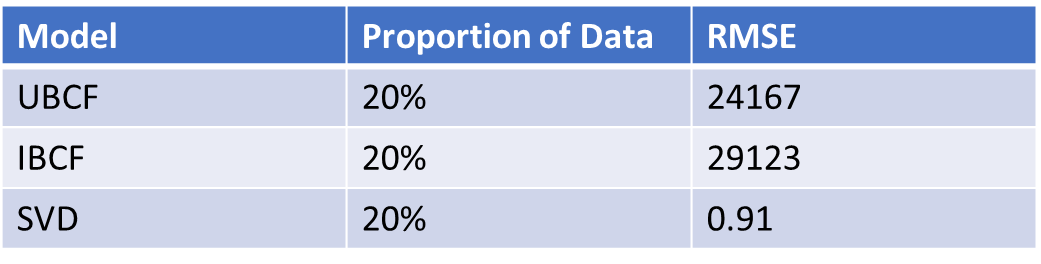
# **Background**

Most Big names in technology industry, such as Disney, YouTube, Amazon, Netflix and many other such web services, use recommender systems to enhance the viewer’s experience and increase profit. From e-commerce (suggest to buyer articles that could interest them) to online advertisement (suggest to users the right contents, matching their preferences), they are on every online platform today. Hence, we decided to build a simple recommendation system based on MovieLens Dataset, to suggest users which movies to watch next.

# **Approach**

In this project three different paradigms of recommender systems are built – Content Based Filtering, Collaborative Filtering and Singular Vector Decomposition. We have covered these recommendation systems in detail, how they work, described their theoretical basis, build them using MovieLens Dataset and compared them.

# **Major Results**

Table 1. Results from All Recommendation system models

# **Conclusion**

Movie recommendations are very subjective and vary from one user to another. Each model has a different approach and its own set of pros and cons. Weighing all the pros and cons, we would recommend SVD as it is a good mix of both collaborative filtering methods

# **Exploratory Data Analysis**

### Exploring Genre

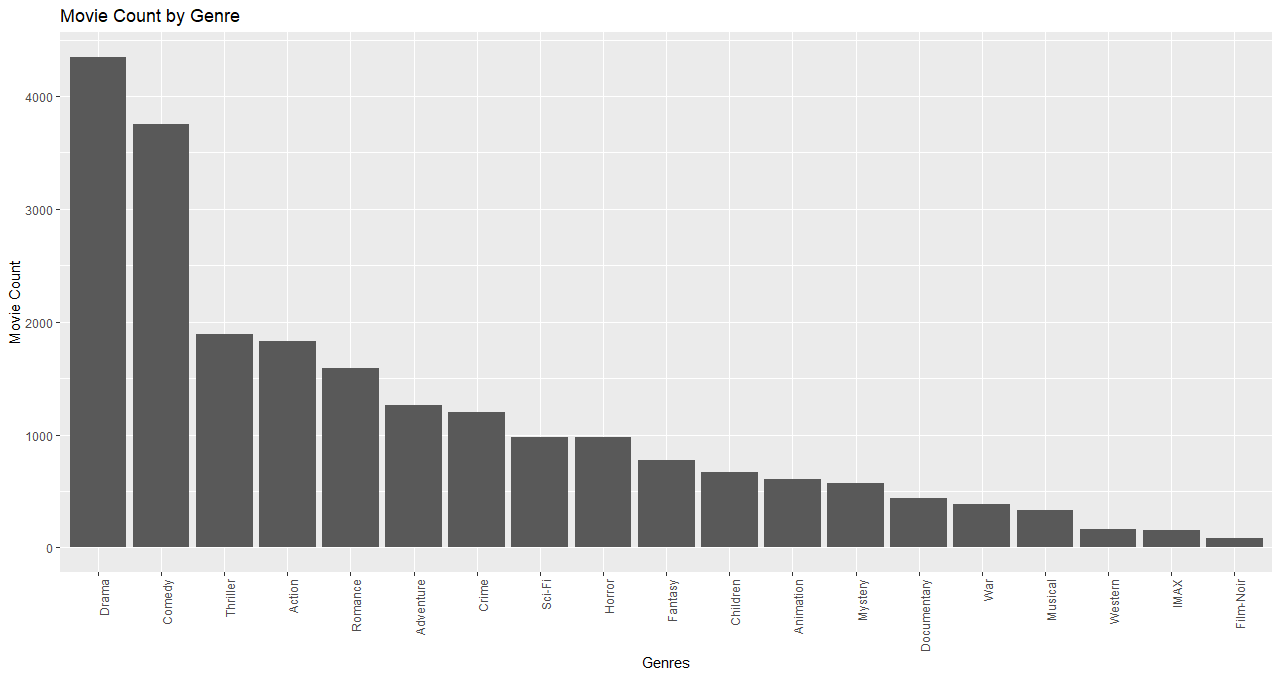
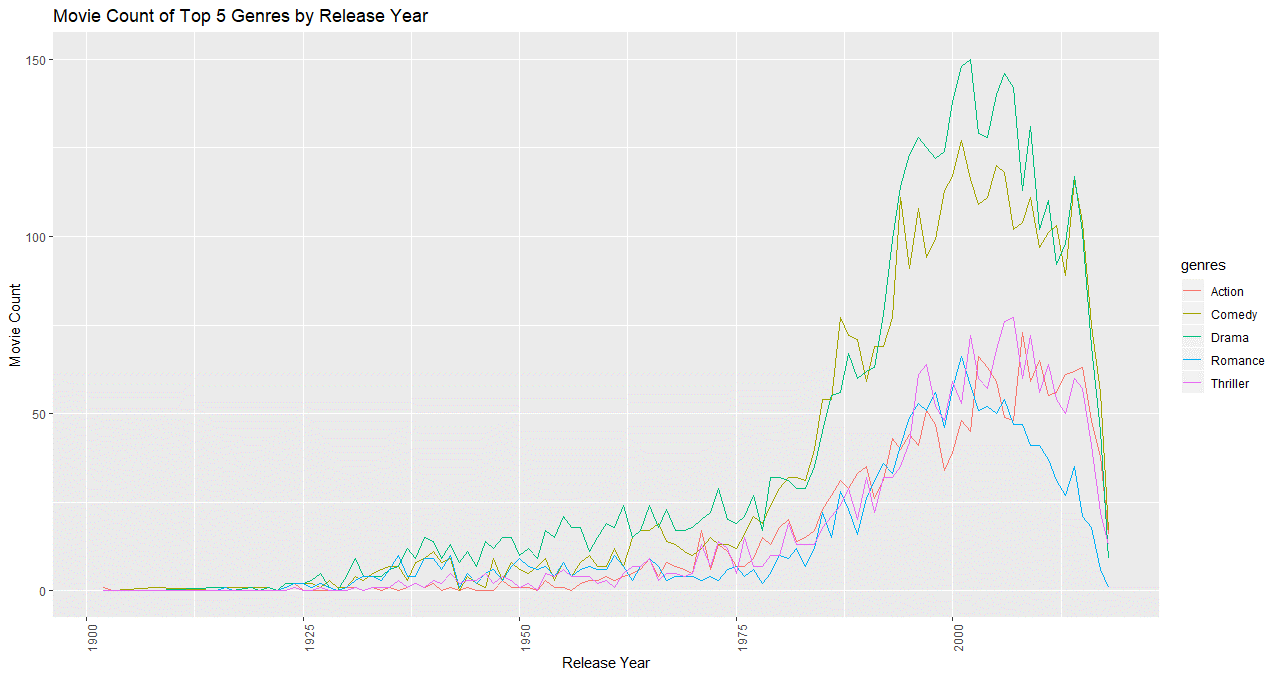
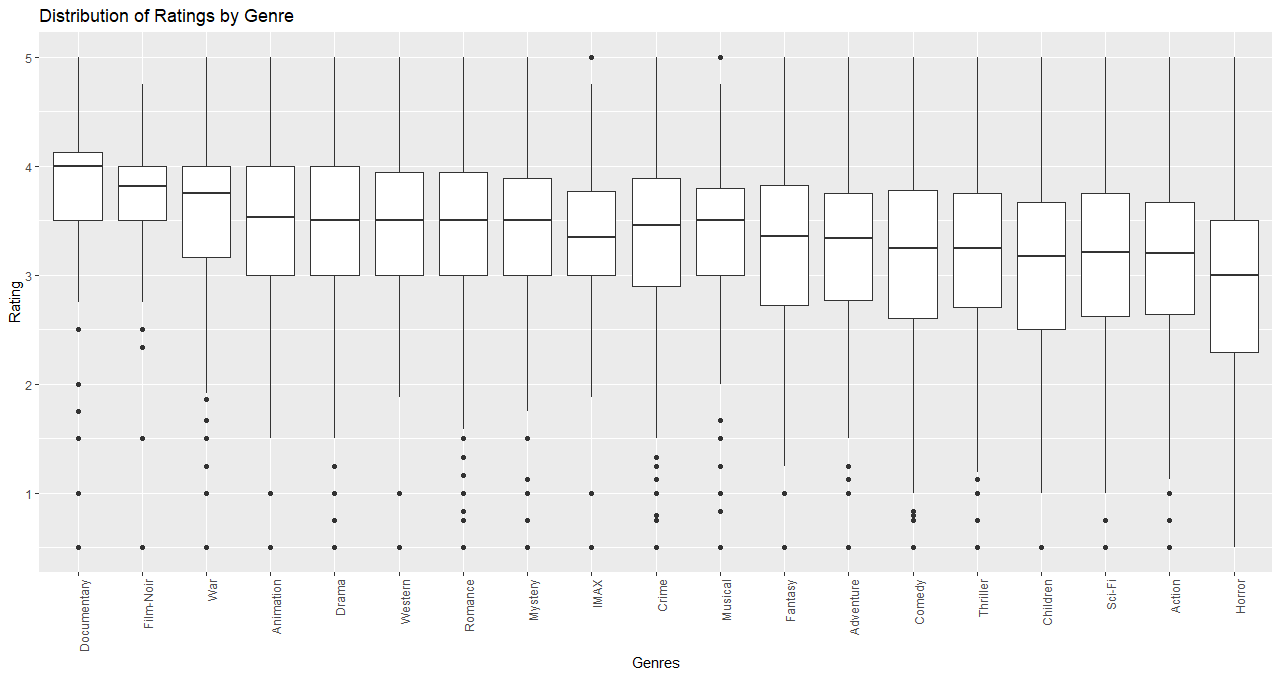
Figure 1: Movie Count by Genre

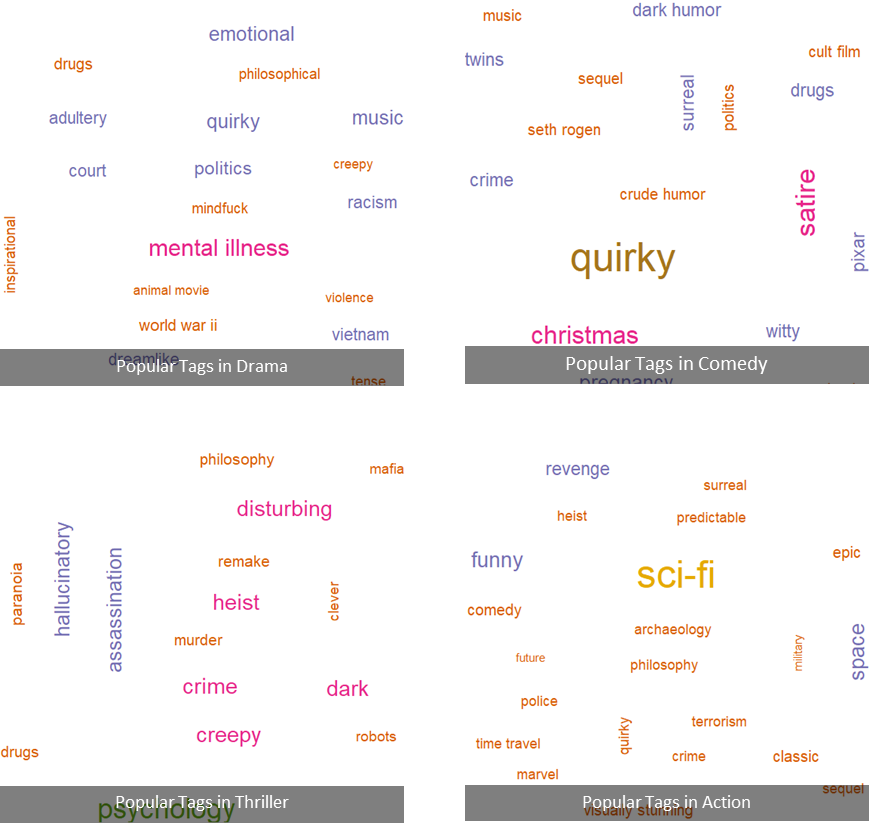
Figure 2: Movie Count of Top 5 Genres by Release Year



### Exploring Ratings

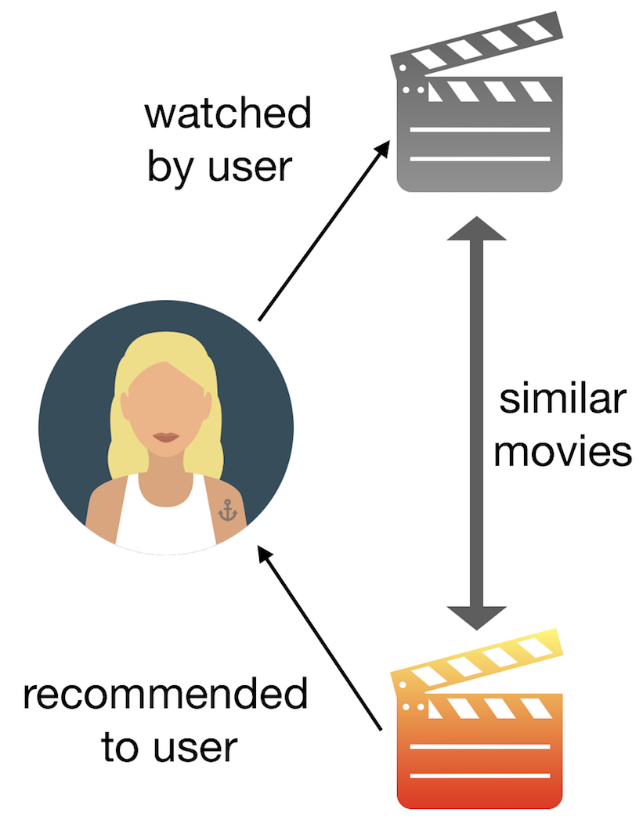
Figure 3: Distribtution of Ratings by Genre

### Exploring Tags

Figure 4: Popular Tags in the Top 4 Genres

# **Content Based Filtering**

## Summary

The basic idea of Content based filtering is that if a user likes an item, he/she will also like a similar item. Content refers to a set of attributes/features that describes an item. For our movie recommendation engine, the recommendations of movies can be based on highest similarity based of features, such as genres, actors, directors, year of release.

Steps to predict movies for a given user:

1. Identify the features to compare with the past browsing history or movies that were rated.

2. Compare the features with other items/ movies by calculating cosine similarity between the items.

3. For a given user, sort the items based on cosine similarity score

4. Finally, recommended the movies associated with the top scores!

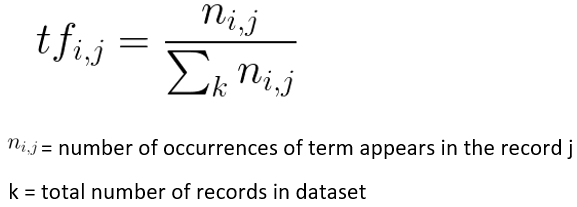
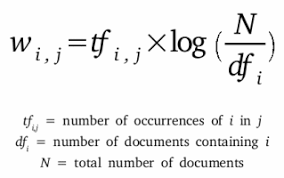
## Math behind similarity calculation

The features to be compared are often words. In order to compare them, we rely on frequency of their occurrence in an item as well as the complete dataset. Mathematically, they are known as Term Frequency (TF) and Inverse Document Frequency (IDF).

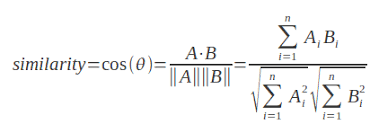
• Term Frequency (TF) – Frequency of a word in given item

• Inverse Document Frequency (IDF) - inverse of the document frequency among the whole dataset

TF-IDF has certain advantages as it determines the importance of words in each item by negating the effect of high frequency words. Taking log value further dampens the effect of high frequency words so that the values are better comparable. The resulting weighted Term frequency is given by the equation:

Such scores for a record form a vector which is normalized determine the similarity of the items using a Vector Space Model. We calculate the cosine similarity score between these vectors in n-dimensional space.

Value of angle or similarity is given by value of cosine between the vectors. Higher cosine value indicates lower value of the angle which signifies more similarity. Thus, we for a given user ID we, recommend movies associated with highest cosine similarity scores.

## Model Result

Table 2. Recommendation for a user who has recently watched Avengers Infinity War



## Pros of Content based filtering

* Able to recommend new as well as unpopular movies
* Quality of recommendation may improve over time
* Able to cater to users with unique tastes
* Independent of data for other users
* Provides explanation by mentioning the content features

## Cons of Content based filtering

* In some domains like music and videos, feature generation is and issue
* Recommends movies based on their likes and preferences only
* Highly dependent on content of the features or descriptive data
* High bias towards the content of features

# **Collaborative Filtering**

## Summary

There are two basic approaches in CF: user-based collaborative filtering and item-based collaborative filtering, respectively.

In both cases this recommendation engine has two steps:

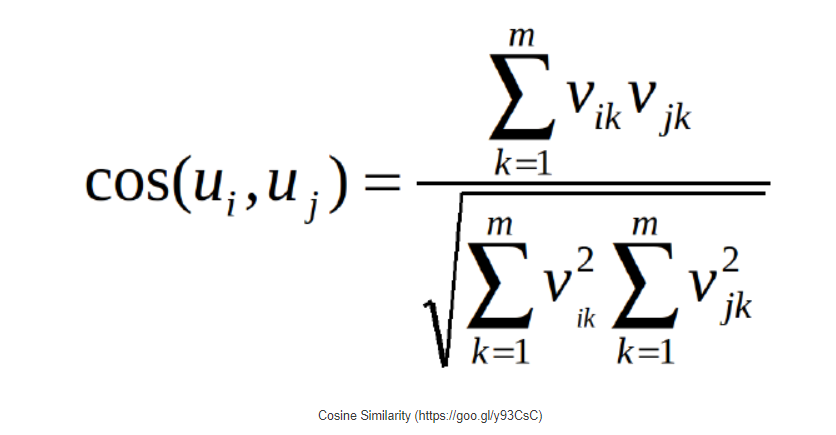
1. Find out how many users/items in the database are like the given user/item.
2. Assess other users/items to predict what grade you would give the user of this product, given the total weight of the users/items that are more like this one.

## Math behind similarity calculation

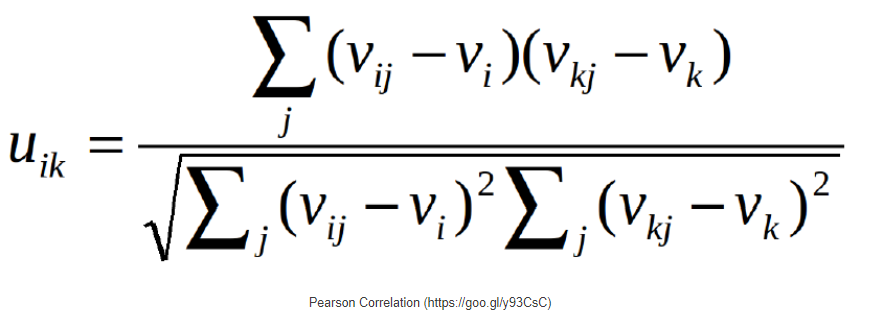
In either scenario, we build a similarity matrix.

There are 3 distance similarity metrics that are usually used in collaborative filtering:

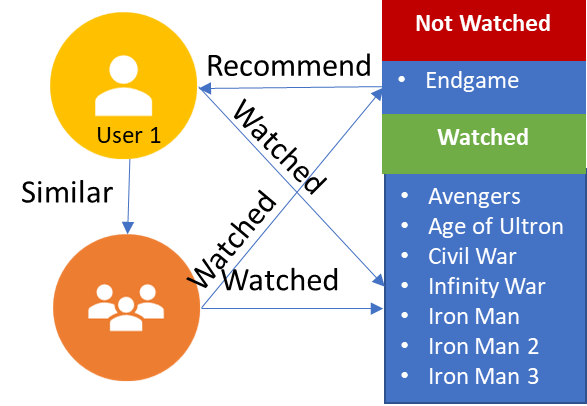
1. Jaccard Similarity: Similarity is based on the number of users which have rated item A and B divided by the number of users who have rated either A or B. It is typically used where we do not have a numeric rating but just a Boolean value like a product being bought or an add being clicked.
2. Cosine Similarity: (as in the Content-Based system) Similarity is the cosine of the angle between the 2 vectors of the item vectors of A and B. Closer the vectors, smaller will be the angle and larger the cosine.



1. Pearson Similarity: Similarity is the Pearson coefficient between the two vectors. For the purpose of diversity, I will use Pearson Similarity in this implementation.



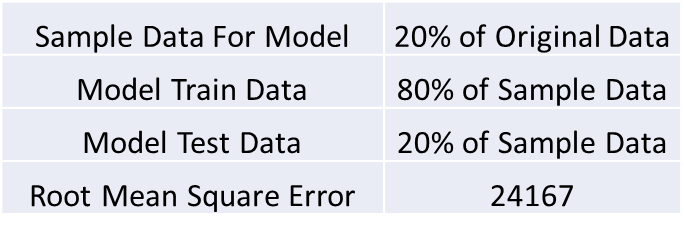
## **User-User Collaborative Filtering**

 Here we find look alike users based on similarity and recommend movies which first user’s look-alike has chosen in past. In order to make a new recommendation to a user, user-user method roughly tries to identify users with the most similar “interactions profile” (nearest neighbors) in order to suggest items that are the most popular among these neighbors (and that are “new” to our user). This method is said to be “user-centered” as it represents users based on their interactions with items and evaluate distances between users This algorithm is very effective but takes a lot of time and resources.

Hence, we have taken only 20% of the original data to build this recommender system.

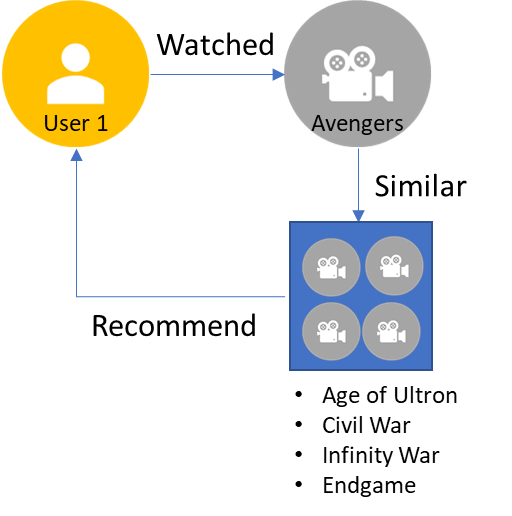
### Model Result

Table 3. Model Result for User-User Collaborative Filtering



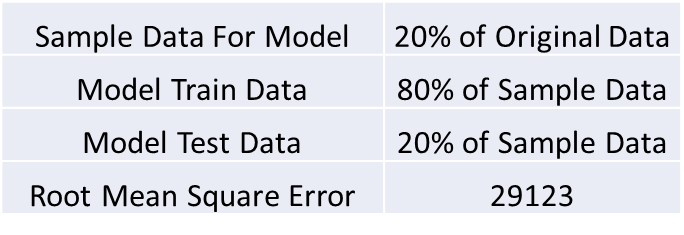
## **Item-Item Collaborative Filtering**

Instead of measuring the similarity between users, the item-based CF recommends items based on their similarity with the items that the target user rated. Likewise, the similarity can be computed with Pearson Correlation or Cosine Similarity.

 This algorithm is far less resource consuming than user-user collaborative filtering. And with fixed number of movies, movie-movie look alike matrix is fixed over time. It avoids the problem posed by dynamic user preference as item-based CF is more static. However, several problems remain for this method. The main issue is scalability.

### Model Result

Table 4. Model Result for Item-Item Collaborative Filtering



## Pros of Collaborative filtering

* No domain knowledge or feature selection is required since interactions yield recommendations
* Implicit user feedback is sufficient
* Can recommend different movies and develop new interests among users
* Does not require contextual features

## Cons of Collaborative filtering

* Cold start problem in case a new user with no history or item with no ratings
* Needs high user: item ratio of at least 1:10
* Can be difficult to provide explanation
* Prone to have a high popularity bias even more so by spam users

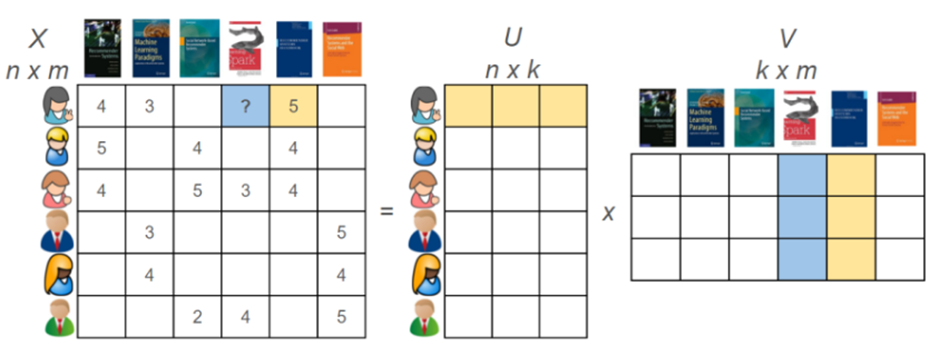
# **Singular Value Decomposition**

## Summary

* The SVD technique was introduced into the recommendation system domain by Brandyn Webb, much more famously known as Simon Funk during the Netflix Prize challenge of 2006.
* The basic essence of SVD is to decomposes a matrix of any shape into a product of 3 matrices with notable mathematical properties: X = U S VT
* This decomposition of ratings matrix results in an ordered matrix of a user feature matrix and an item feature matrix which encapsulate the variance associated with every direction of the matrix.
* The assumption here is that larger variances indicate less redundancy and less correlation and hold features of data.

Steps to predict movies for a given user:

1. Create a ratings matrix X with user Id and ratings assigned by each user Id to the respective movies.
2. Decompose the matrix A into orthogonal matrices (Q-1=QT) with orthonormal eigenvectors (matrix are directions of maximum spread or variance of data) chosen from AAᵀ and AᵀA, respectively. S is a diagonal matrix with r elements equal to the root of the positive eigenvalues (scalar by which eigen vector can be scaled) of AAᵀ or Aᵀ A (both matrices have the same positive eigenvalues any). The diagonal elements are composed of singular values.
3. Predict the rating by simply looking up the entry for the appropriate user/movie pair in the matrix A.
4. Finally, recommend movies with a higher rating to the user!



## Math behind decomposition

Let the rating matrix X of size n x m be decomposed as

A = U S VT

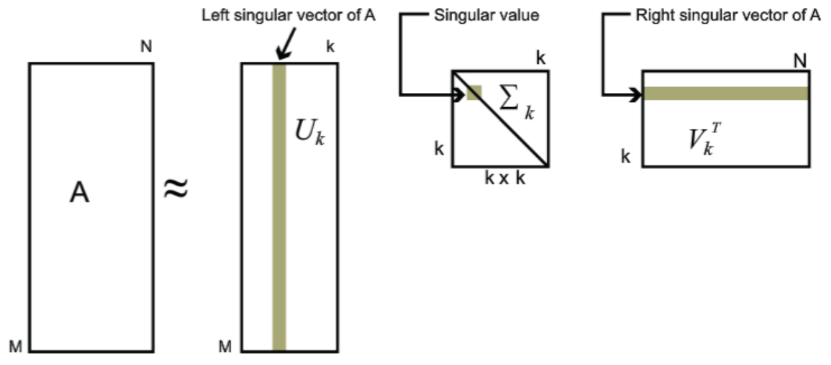
Where U = matrix of n users and a set k of items

V= matrix of k items and m movies

S = Diagonal matrix with values representing the singular values or the weights or the importance values of different features in the matrix

k= number of latent features to be discovered

Rank of a matrix is a measure of the unique information stored in a matrix. Higher the rank, more the information



The aim is to get a lower rank matrix of order k that retains only the top k features enclosing the most important underlying taste and preference vectors by setting the smallest singular values to 0 since these are assumed to correspond to noise. We then calculate energy defined by the sum of squares of singular values and select an optimum value of k (number of singular values) which, ideally retains 90 % of energy.

## Model Results

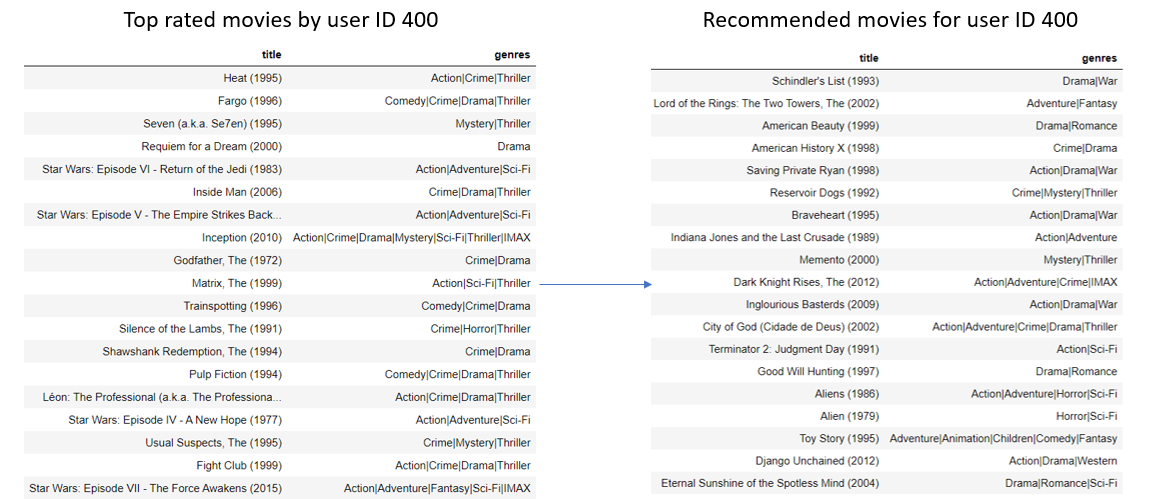


Table 5. Model Results for SVD



## Pros of SVD

* Captures essence of all the input features
* Optimal low-rank approximation
* No cold start problems
* Able to recommend new items to users

## Cons of SVD

* + Computationally expensive
  + Needs attention to missing data
  + Lack of sparsity since singular vectors are usually dense
  + Assumes that the data is normally distributed
  + Low interpretability since a singular vector is a linear combination of features

# **References**

1. <https://www.dataminingapps.com/2020/02/singular-value-decomposition-in-recommender-systems/>
2. <https://alyssaq.github.io/2015/20150426-simple-movie-recommender-using-svd/>
3. <https://heartbeat.fritz.ai/recommender-systems-with-python-part-iii-collaborative-filtering-singular-value-decomposition-5b5dcb3f242b>
4. <https://www.analyticsvidhya.com/blog/2019/08/5-applications-singular-value-decomposition-svd-data-science/>
5. <http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/svd.html>
6. <https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/>
7. <https://towardsdatascience.com/the-4-recommendation-engines-that-can-predict-your-movie-tastes-109dc4e10c52>
8. <https://muffynomster.wordpress.com/2015/06/07/building-a-movie-recommendation-engine-with-r/>
9. <https://blog.statsbot.co/recommendation-system-algorithms-ba67f39ac9a3>
10. <https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75>
11. <https://sifter.org/~simon/journal/20061211.html>