DEPARTMENT OF COMPUTER SCIENCE,
UNIVERSITY OF DELHI

Movie Recommendation System

By:

Pooja Sahu

About the Project

- Introduction
- Statement of the Problem
- Objective
- Related Work
- Methodology
- Summary and Conclusion

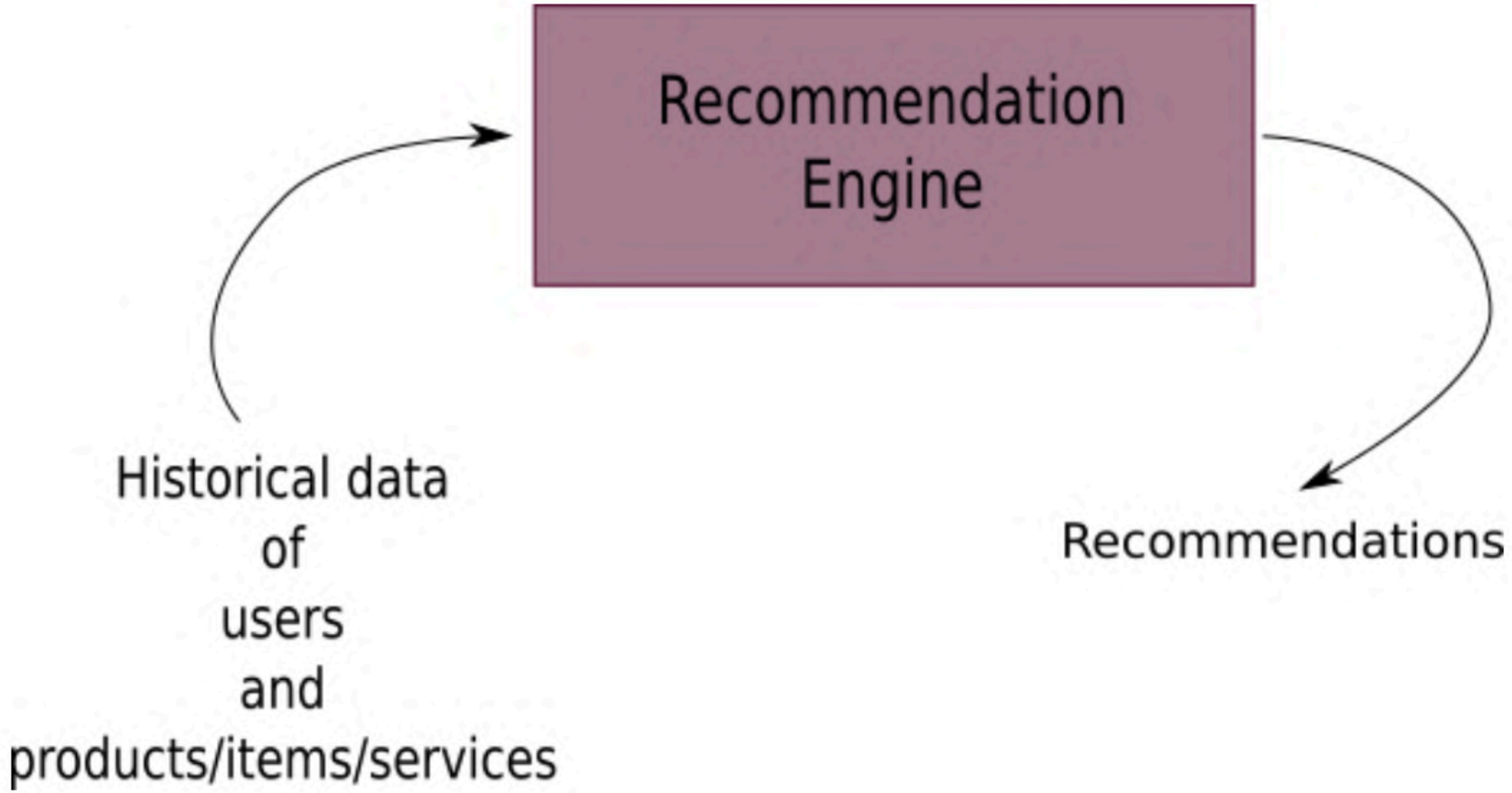
What is recommedation system?

- A recommendation system is any system that automatically suggests content, products or serivces which should interest customers based on their preferences

Why recommendation system is useful for organization?

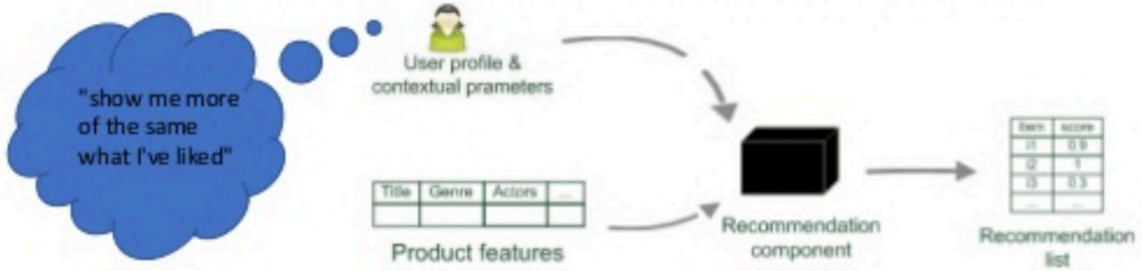
- Help to increase the site’s page views, dwell time, click-through rate, and retention
- Help generate more advertising revenue
- Increase upselling and crossselling

How do we build recommendation engine?



Approach 1 : Content Based

- **The requirement**
  - some information about the available items such as the genre ("content")
  - some sort of *user profile* describing what the user likes (the preferences)
- "**Similarity**" is computed from item attributes, e.g.,
  - Similarity of movies by actors, director, genre
  - Similarity of text by words, topics
  - Similarity of music by genre, year
- **The task:**
  - learn user preferences
  - locate/recommend items that are "similar" to the user preferences



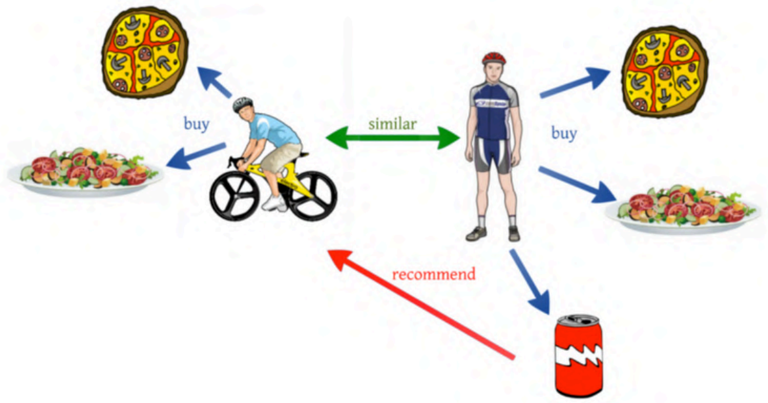
Approach 2 : Collaborative Filtering (CF) Based

Types of CF

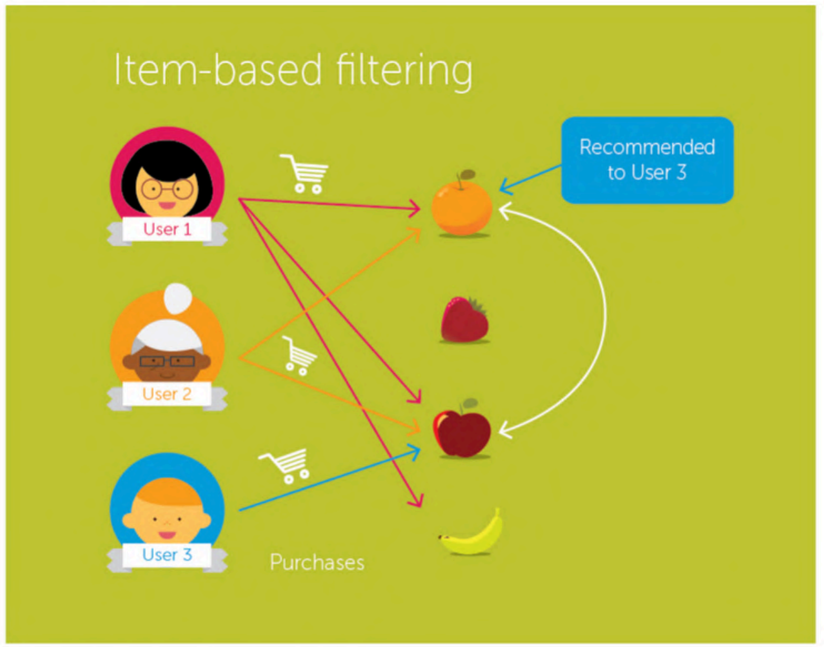
1. User-user Collaborative Filtering
2. Item-item Collaborative Filtering

- User-Item Collaborative Filtering: "Users who are similar to you also liked ..."
- Item-Item Collaborative Filtering: "Users who liked this item also liked ..."

** User-User CF**



** Item-item CF**



Related Work

https://ieeexplore.ieee.org/document/9566476

Movie Recommendation Systems Using Actor-Based Matrix Computations in South Korea

- The paper focuses on a content-based movie recommendation system in South Korea, utilizing data from the Korean Film Council, Naver movie site, and Korea Box Office Information System.
- Data preprocessing steps include removing duplicates, excluding certain genres, and incorporating additional movie-related information, with a final dataset combining actor, director, genre, plot, and viewer-related details.
- Correlation Analysis and User Evaluation:
- The study involves computing rank correlation between movies and genres, as well as the correlation between actors and genres using Pearson's correlation coefficient.
- Results reveal notable differences in correlation patterns between movies and genres compared to actors and movie genres. Additionally, a user survey is conducted to assess the recommendation system, with participants expressing preferences based on movie genres or actors.

Movie Recommendation System Based on SVD Collaborative Filtering

https://ieeexplore.ieee.org/document/10144401

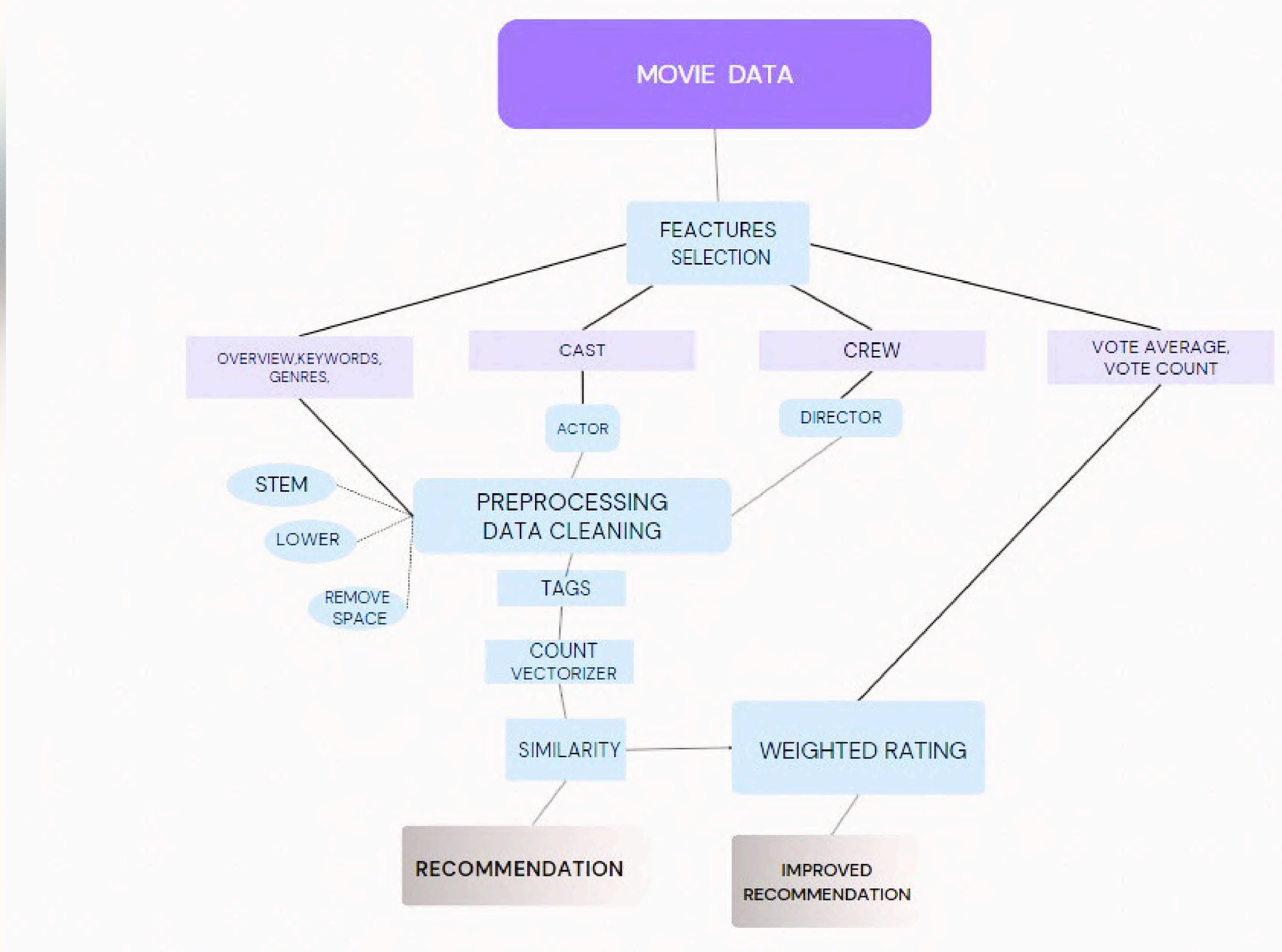
- The second paper discusses Collaborative Filtering (CF), specifically User-CF and Item-CF, as mainstream recommendation algorithms, focusing on personalized movie recommendations by identifying users with similar interests.
- Singular Value Decomposition (SVD) is highlighted as a superior algorithm in the Movie Recommendation System based on the MovieLens dataset, outperforming Item-CF and User-CF, especially in scenarios with sparse data.

The Movies Dataset

https://grouplens.org/datasets/movielens/latest/li

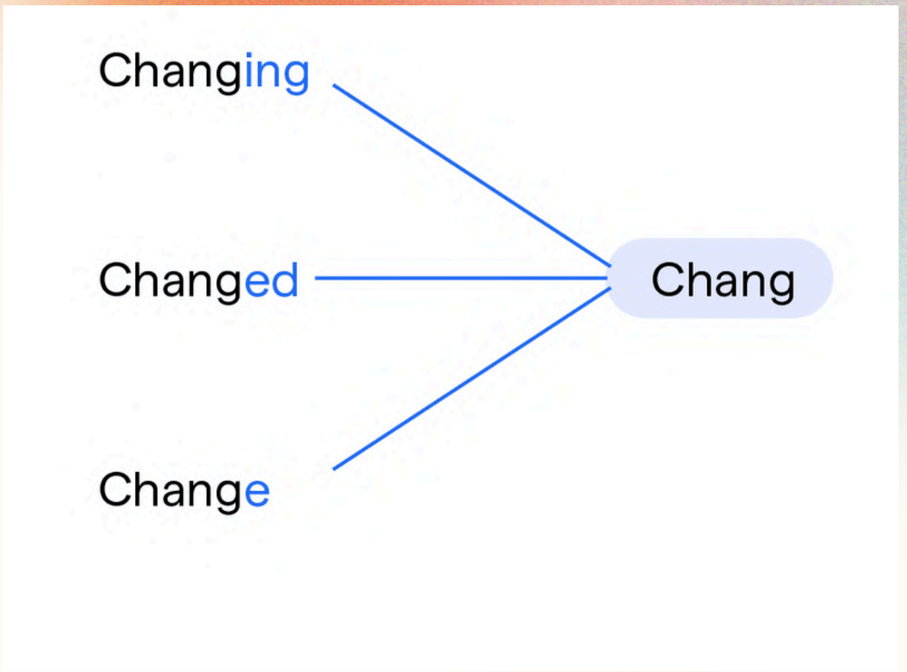
movies_metadata.csv	• The main Movies Metadata file. Contains information on 45,000 movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.
keywords.csv	• Contains the movie plot keywords for our MovieLens movies. Available in the form of a stringified JSON Object.
credits.csv	• Consists of Cast and Crew Information for all our movies. Available in the form of a stringified JSON Object.
links.csv	• The file that contains the TMDB and IMDB IDs of all the movies featured.
links_small.csv	• Contains the TMDB and IMDB IDs of a small subset of 9,000 movies.
ratings_small.csv	• The subset of 100,000 ratings from 700 users on 9,000 movies.

WorkFlow OF content based and improved Recommendation system



Stemming

Stemming is a natural language processing technique that is used to reduce words to their base form, also known as the root form. The process of stemming is used to normalize text and make it easier to process.



Count Vectorization

For each document, a vector is created where each element represents the count of the corresponding term in the document. The result is a matrix where rows correspond to documents, and columns correspond to terms in the vocabulary.
Here's a simple example:
Suppose you have 4 documents:
Document 1:"The greatest thing of life is love."
Document 2: "Love is great, it's great to be loved."
Document 3: "Is love the greatest thing?"
Document 4:"I love lasagna for 1000 times."

	for	great	greatest	lasagna	life	love	loved	the	thing	times
sentence 1	0	0	1	0	1	1	0	1	1	0
sentence 2	0	2	0	0	0	1	1	0	0	0
sentence 3	0	0	1	0	0	1	0	1	1	0
sentence 4	1	0	0	1	0	1	0	0	0	1

Cosine similarity

Cosine similarity is a metric used to measure how similar two vectors are, especially in the context of text data and document similarity. It calculates the cosine of the angle between two vectors, providing a measure of similarity irrespective of their magnitude

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

$$\cos\theta = \frac{\overrightarrow{a}.\overrightarrow{b}}{\|\overrightarrow{a}\|\|\overrightarrow{b}\|}$$

$$\|\overrightarrow{a}\| = \sqrt{a_1^2 + a_2^2 + a_3^2 + \ldots + a_n^2} \|\overrightarrow{b}\| = \sqrt{b_1^2 + b_2^2 + b_3^2 + \ldots + b_n^2}$$

Output of content based model

Input: recommend(movie) takes a movie title (movie) as input for which the user wants recommendations.
Output: The function prints the titles of 10 other movies that are most similar to the input movie.

```
def recommend(movie):
    index=new_df[new_df['original_title']==movie].index[0]
    distance=similarity[index]
    movie_list=sorted(list(enumerate(distance)),reverse=True,key=lambda x:x[1])[1:11]
    for i in movie_list:
        print(new_df.iloc[i[0]].original_title)
```

```
recommend("Avatar")
```

```
Aliens vs Predator: Requiem
Jupiter Ascending
Starman
Independence Day
Titan A.E.
Aliens
Battle: Los Angeles
Small Soldiers
Contamination
Babylon 5: Thirdspace
Meet Dave
10.000 BC
```

output

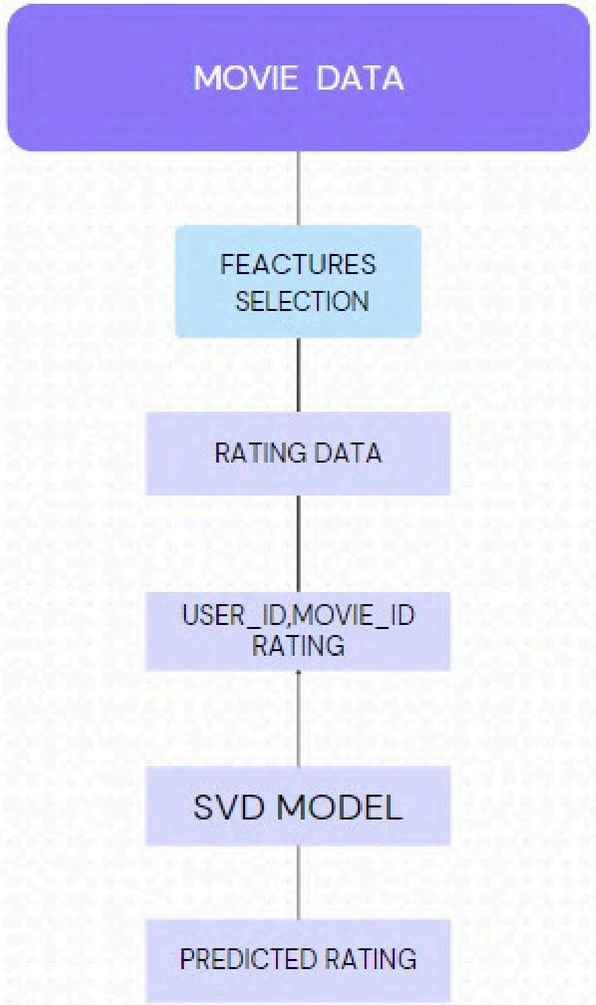
Improved Content Based

It Uses Vote Count and vote average calculates weighted rating which can be calculated as:

$$WeightedRating(WR) = (v.\mathrm{R}/(v+m))+(m.\mathrm{C}/(v+m))$$

where,
v is the number of votes for the movie
m is the minimum votes required to be listed in the chart
R is the average rating of the movie
C is the mean vote across the whole report

Collaborative Filtering Based



Our content based engine suffers from some severe limitations.

- It is only capable of **suggesting movies which are close to a certain movie.** That is, it is **not capable of capturing tastes** and providing recommendations across genres.
- Also, the engine that **we built is not really personal in that it doesn't capture the personal tastes and biases of a user.** Anyone querying our engine for recommendations based on a movie will receive the same recommendations for that movie, regardless of who (s)he is.
- Therefore, in this section, we will use **Collaborative Filtering** to make recommendations to Movie Watchers. Collaborative Filtering is based on the idea that users similar to a me can be used to predict how much I will like a particular product or service those users have used/experienced but I have not

SVD Model

SVD (Singular Value Decomposition) is a popular matrix factorization technique commonly used in collaborative filtering-based movie recommendation systems. In a movie recommendation system, SVD is applied to decompose the user-item interaction matrix into three matrices:

- User Matrix (U): Represents users and their latent factors.
- Item Matrix (V): Represents items (movies) and their latent factors.
- Singular Value Matrix (Σ): Represents the singular values.

The user-item interaction matrix M is decomposed as M is nearly equal to U.Σ.VT

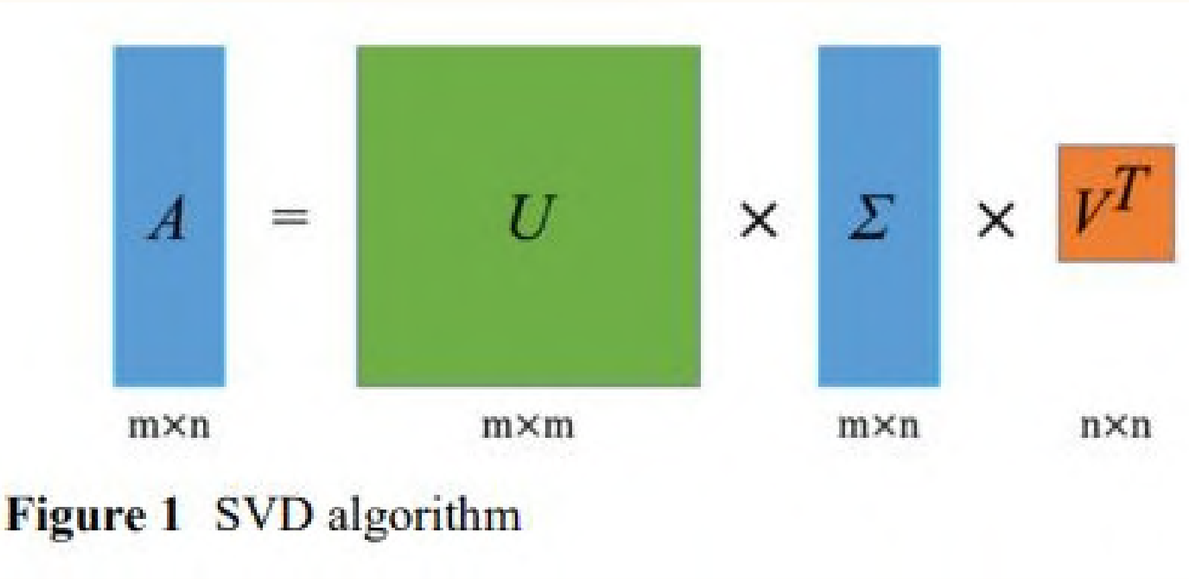
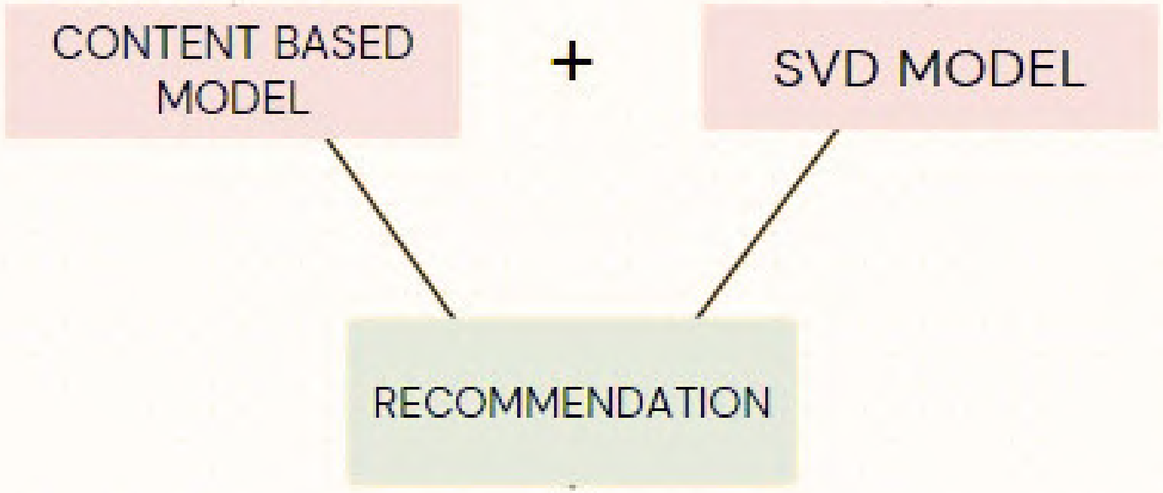


Figure 1 SVD algorithm

Hybrid Movie Recommendation System

Build a simple hybrid recommender that brings together techniques we have implemented in the content based and collaborative filter based engines.



Summary

In this Project, we have built 4 different recommendation engines based on different ideas and algorithms. They are as follows:

Content Based Recommender

We built two content based engines; one that took movie overview and taglines as input and the other which took metadata such as cast, crew, genre and keywords to come up with predictions. We also devised a simple filter to give greater preference to movies with more votes and higher ratings.

Improved Content Based Recommender

This system used overall TMDB Vote Count and Vote Averages to build Top Movies Charts, in general and for a specific genre. The IMDB Weighted Rating System was used to calculate ratings on which the sorting was finally performed.

Summary

Collaborative Filtering

We used the powerful Surprise Library to build a collaborative filter based on single value decomposition. The RMSE obtained was less than 1 and the engine gave estimated ratings for a given user and movie.

Hybrid Engine.

We brought together ideas from content and collaborative filtering to build an engine that gave movie suggestions to a particular user based on the estimated ratings that it had internally calculated for that user.

Conclusion

We see that for our hybrid recommender, we get different recommendations for different users although the movie is the same. Hence, our recommendations are more personalised and tailored towards particular users.

Thank You for listening!