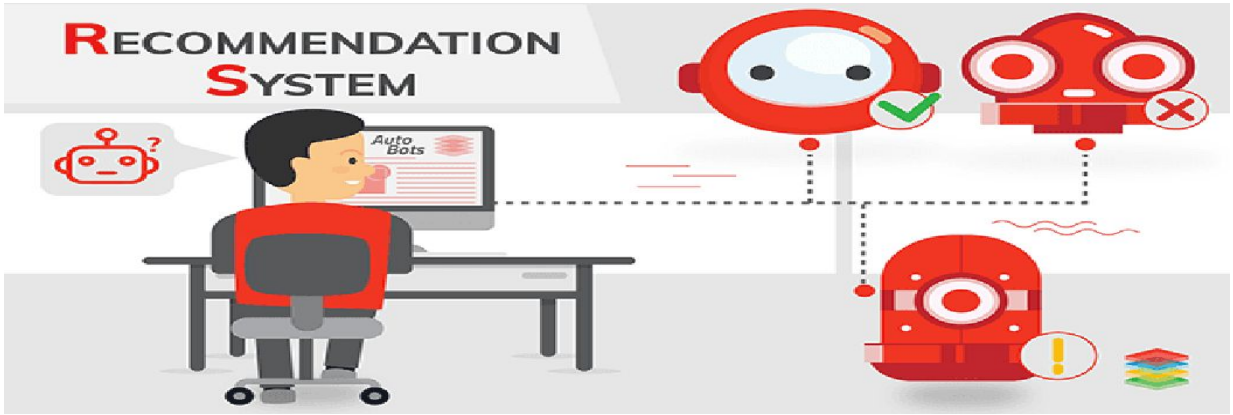
Movie Recommendation System

GROUP-19

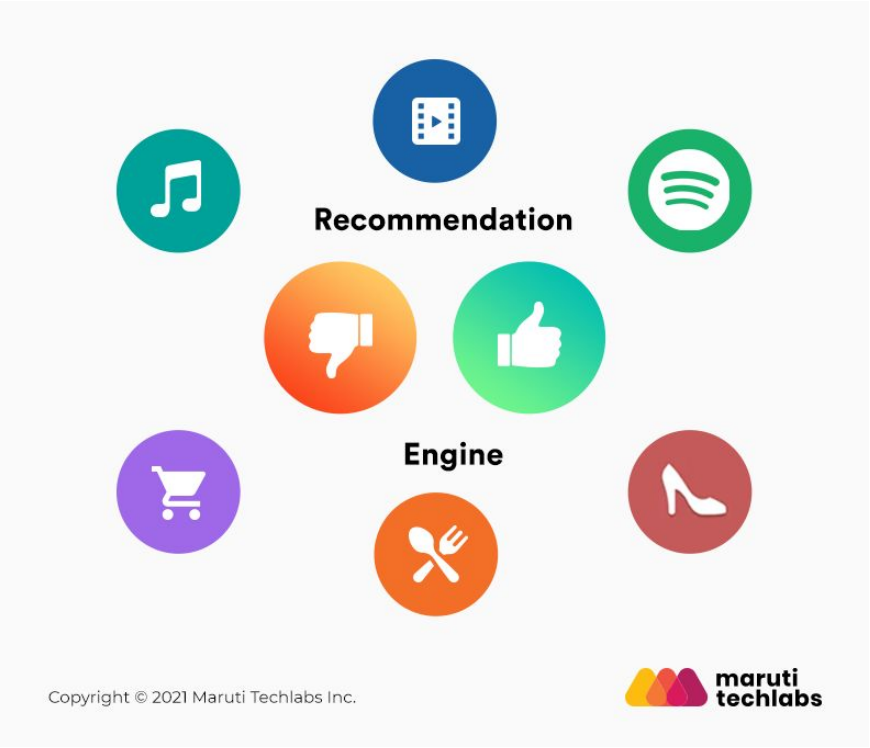
Introduction

Recommender System is a system that seeks to predict or filter preferences according to the user's choices. Recommender systems are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general.

Movies are one of the sources of entertainment, but the problem is in finding the desired content from the ever-increasing millions of content every year.However, recommendation systems come much handier in these situations.The aim of this project is to build a movie recommendation system using - content based and collaborative approach.



Examples



- If a user listens to rock music every day, his youtube recommendation feed will get full of rock music and music of related genres.
- All the websites such as Amazon,Flipkart
- Uber also uses it(Suppose frequently you are using mini cab then it will recommend you to use that by default)
- Google also feeds new as the notification(Based on what you have searched or read in past).
- Netflix,Amazon Prime.

Paper Implentation

Movie Recommender system using collaborative filtering

The purpose of this paper is to provide an approach that increases the precision and performance of a standard filtering approach. Although there are other approaches for implementing a recommendation system, content-based filtering is the most straightforward. Which takes the user's input, double-checks his/her history/past behavior, and suggests a list of related films. To demonstrate the effectiveness, K-NN algorithms and collaborative filtering are utilized in this research, with the primary goal of improving the accuracy of outcomes over content-based filtering.

Dataset:

We have used two MovieLens datasets.

The Full Dataset: Consists of 26,000,000 ratings and 750,000 tag applications applied to 45,000 movies by 270,000 users. Includes tag genome data with 12 million relevance scores across 1,100 tags.

The Small Dataset: Comprises of 100,000 ratings and 1,300 tag applications applied to 9,000 movies by 700 users.

Content Recommendation System

Content filtering as the name suggests is based on content similarities.

For eg let's say you bought something from Amazon and when the next time you login again to the site, you will be recommended similar products.

The same happens while surfing for the movie suppose you like horror genre so the you will be recommended horror movies.

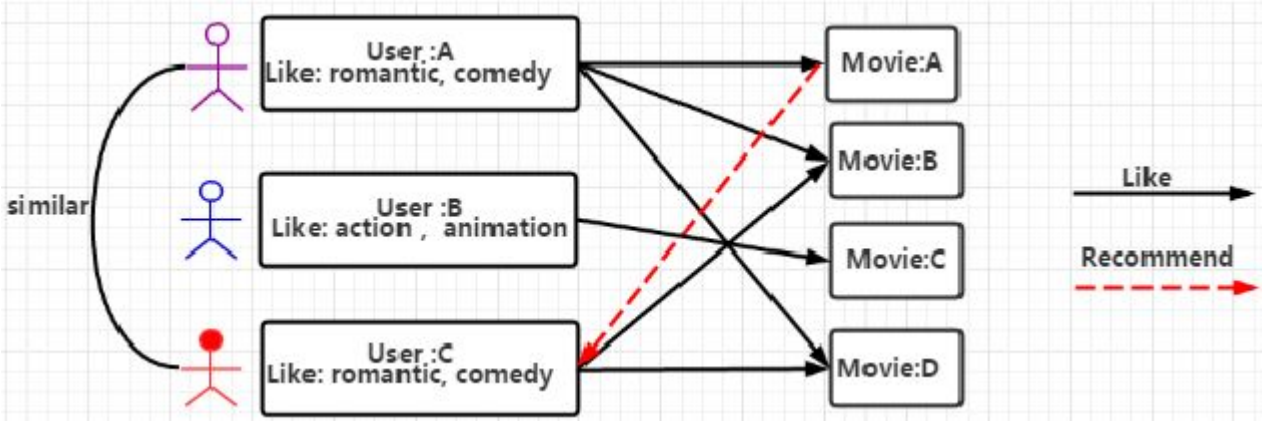
Also movies can be recommended on the basis of cast and crew, time of release,director etc. So basically the user will follow it's historic footprints and thus this is the basis of content filter.

Collaborative Recommendation System

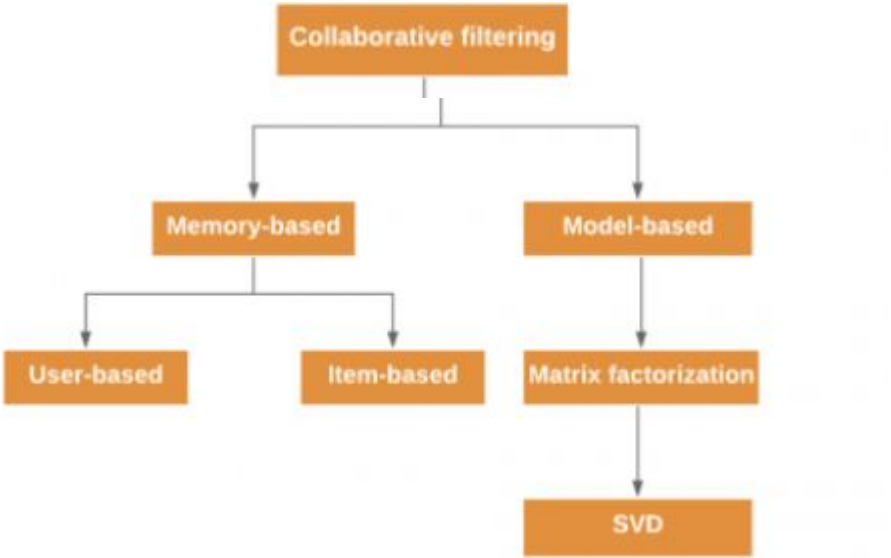
Our content-based engine has a number of serious flaws. It can only recommend films that are similar to a specific film. That is, it is incapable of collecting tastes and making genre-specific recommendations. Also, the engine we created isn't truly personal because it doesn't account for a user's particular preferences and biases. Regardless of who is asking our engine for recommendations based on a movie, everyone will receive the same recommendations for that movie.

Collaborative Based Recommendation System:

It matches users with same interests and gives recommendations based on their likes.



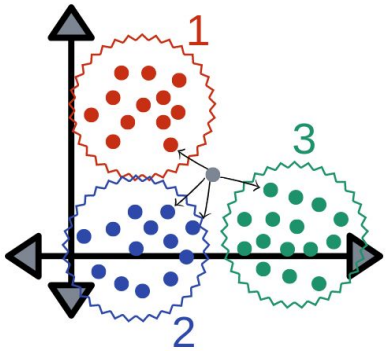
Further, there are several types of collaborative filtering algorithms —



Collaborative Filtering using k-Nearest Neighbors (kNN):

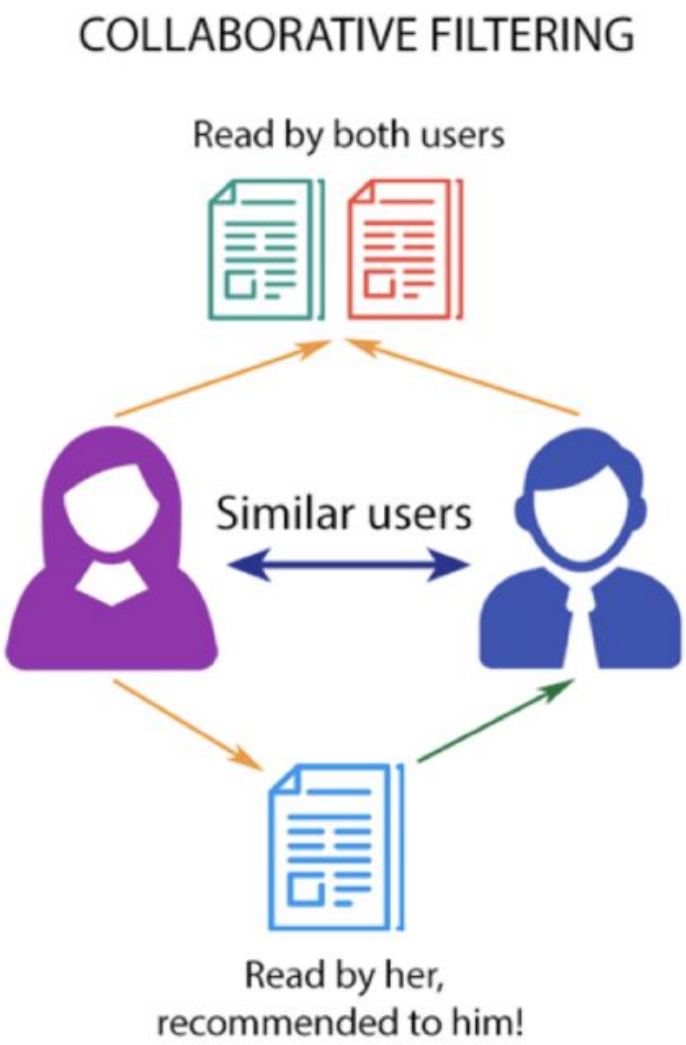
kNN uses the average rating of top-k nearest neighbours to locate clusters of similar users based on common movie ratings and generate predictions.

The KNN algorithm is well-known for its quick prediction and minimal calculation time. By predicting a similarity measure, KNN classifies each unlabeled class to their corresponding classes.



SVD(Singular Value Decomposition)

Two popular approaches of CF are latent factor models, which extract features from user and item matrices and neighbourhood models, which finds similarities between products or users. Singular Value Decomposition (SVD) extract features and correlation from the user-item matrix. SVD would generate factors when looking into the dimension space like action vs comedy, Hollywood vs Bollywood, or Marvel vs Disney. It shrinks the space dimension from N-dimension to K-dimension (where K<N) and reduces the number of features.



SVD(Singular Value Decomposition)

SVD is a widely used technique to decompose a matrix into several component matrices, exposing many of the useful and interesting properties of the original matrix.SVD constructs a matrix with the row of users and columns of items and the elements are given by the users' ratings. Singular value decomposition decomposes a matrix into three other matrices and extracts the factors from the factorization of a high-level (user-item-rating) matrix.

$$A=USV^T$$

Matrix U: singular matrix of (user*latent factors)

Matrix S: diagonal matrix (shows the strength of each latent factor)

Matrix V: singular matrix of (item*latent factors)

SVD CODE

```
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
data.split(n_folds=5)
```

```
svd = SVD()
evaluate(svd, data, measures=['RMSE', 'MAE'])
```

```
trainset = data.build_full_trainset()
svd.train(trainset)
```

```
ratings[ratings['userId'] == 1]
```

```
svd.predict(1, 302, 3)
```

```
Prediction(uid=1, iid=302, r_ui=3, est=2.8779447226327712, details={'was_impossible': False})
```

For movie with ID 302, we get an estimated prediction of 2.686. One startling feature of this recommender system is that it doesn't care what the movie is (or what it contains). It works purely on the basis of an assigned movie ID and tries to predict ratings based on how the other users have predicted the movie.

Disadvantages of Previous Techniques

- **Content Based**
  - Lack of novelty and diversity.
  - Scalability is a challenge.

- **Collaborative Based**
  - Cold Start problem
  - Grey Sheep Problem

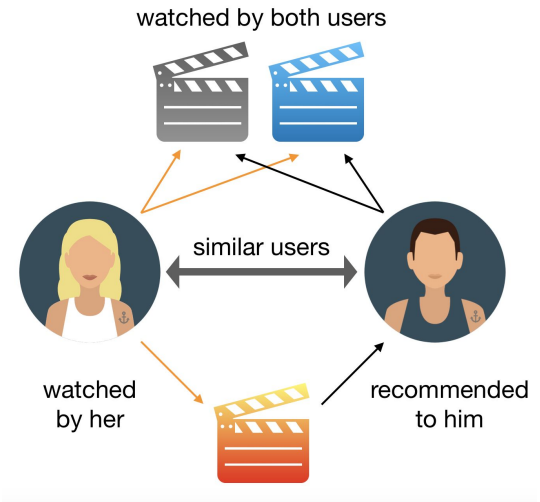
Innovation-Hybrid Recommender

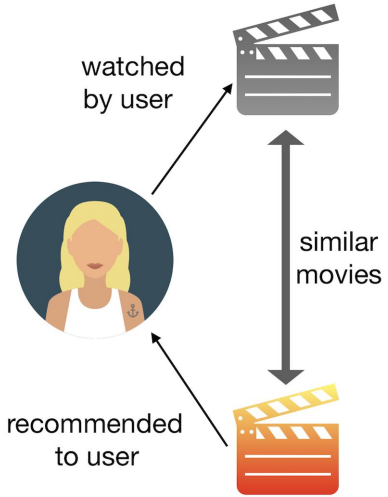
```
In [109]: get_recommendations('The Dark Knight').head(10)

Out[109]: 7931                        The Dark Knight Rises
          132                               Batman Forever
          1113                              Batman Returns
          8227    Batman: The Dark Knight Returns, Part 2
          7565                Batman: Under the Red Hood
          524                                      Batman
          7901                        Batman: Year One
          2579              Batman: Mask of the Phantasm
          2696                                       JFK
          8165    Batman: The Dark Knight Returns, Part 1
          Name: title, dtype: object
```

Hybrid System

Our Innovation is to build a simple hybrid recommender that brings together two techniques content based and collaborative filtering .





Code

```
def hybrid(userId, title):
    idx = indices[title]
    tmdbId = id_map.loc[title]['id']
    #print(idx)
    movie_id = id_map.loc[title]['movieId']

    sim_scores = list(enumerate(cosine_sim[int(idx)]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:26]
    movie_indices = [i[0] for i in sim_scores]

    movies = smd.iloc[movie_indices][['title', 'vote_count', 'vote_average', 'year', 'id']]
    movies['est'] = movies['id'].apply(lambda x: svd.predict(userId, indices_map.loc[x]['movieI
d']).est)
    movies = movies.sort_values('est', ascending=False)
    return movies.head(10)
```

Firstly we are sorting the list by content based approach using the cosine formula(used to check similarity) and storing the top 26 results in the movie_indices.

After that we are applying the SVD collaborative filtering algorithm to the movie_indices and thereafter printing the results.

Results

In [67]:
```
hybrid(500, 'Avatar')
```

Out[67]:

	title	vote_count	vote_average	year	id	est
8401	Star Trek Into Darkness	4479.0	7.4	2013	54138	3.238226
974	Aliens	3282.0	7.7	1986	679	3.203066
7265	Dragonball Evolution	475.0	2.9	2009	14164	3.195070
831	Escape to Witch Mountain	60.0	6.5	1975	14821	3.149360
1668	Return from Witch Mountain	38.0	5.6	1978	14822	3.138147
1376	Titanic	7770.0	7.5	1997	597	3.110945
522	Terminator 2: Judgment Day	4274.0	7.7	1991	280	3.067221
8658	X-Men: Days of Future Past	6155.0	7.5	2014	127585	3.043710
1011	The Terminator	4208.0	7.4	1984	218	3.040908
2014	Fantastic Planet	140.0	7.6	1973	16306	3.018178

In [66]:
```
hybrid(1, 'Avatar')
```

Out[66]:

	title	vote_count	vote_average	year	id	est
1011	The Terminator	4208.0	7.4	1984	218	3.083605
522	Terminator 2: Judgment Day	4274.0	7.7	1991	280	2.947712
8658	X-Men: Days of Future Past	6155.0	7.5	2014	127585	2.935140
1621	Darby O'Gill and the Little People	35.0	6.7	1959	18887	2.899612
974	Aliens	3282.0	7.7	1986	679	2.869033
8401	Star Trek Into Darkness	4479.0	7.4	2013	54138	2.806536
2014	Fantastic Planet	140.0	7.6	1973	16306	2.789457
922	The Abyss	822.0	7.1	1989	2756	2.774770
4966	Hercules in New York	63.0	3.7	1969	5227	2.703766
4017	Hawk the Slayer	13.0	4.5	1980	25628	2.680591

Conclusion

In this project we have seen 3 different recommendation system -

1. **Content Based Recommender:** the one that took movie overview,taglines, cast, crew, genre and keywords as input and come up with predictions. We also devised a simple filter to give greater preference to movies with more votes and higher ratings.
2. **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.
3. **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.

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 personalized and tailored towards particular users