TITLE OF THE PROJECT

Recommender Systems

Kunal Kalra 20BCE2035

***Abstract*— In recent years, internet has become a huge pool for choices on everything, from streams music, movies, playing games to buying mobiles, laptops and books. Hence a filter to effectively sort more relevant options becomes a necessity. Recommender systems solves this problem by searching through large volume information to provide users with personalized content and services. This report contains more information about on different types recommender system, movie recommender systems with content-based approach, the project flow and the methodology.**

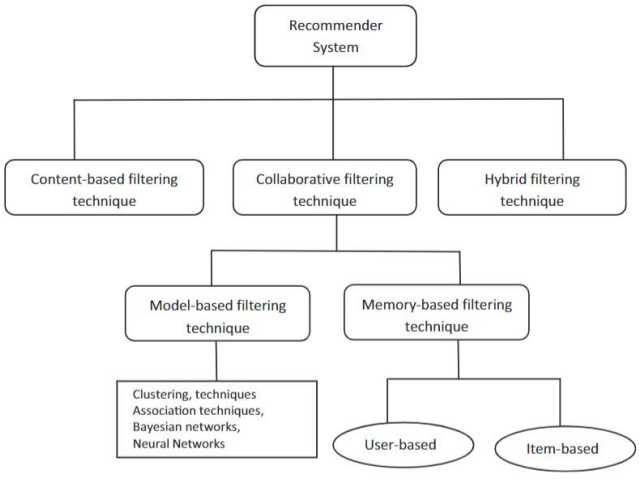
***Index Terms*—Enter key words or phrases in alphabetical order, separated by commas.**

I. INTRODUCTION

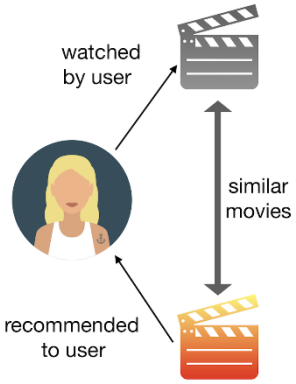
The enormous development in the volume of digital information available and the number of Internet users has created a possible challenge of information overload, which makes it difficult to have timely access to items of interest on the Internet. Information retrieval systems like Google, Devil Finder, and Altavista have partially solved this challenge, but prioritization and personalization of information (where a system maps accessible content to a user's interests and preferences) were missing. The need for recommender systems has risen to unprecedented levels as a result of this. Recommender systems are information filtering systems that address the problem of information overload by extracting critical information fragments from a huge amount of dynamically generated data based on the user's choices, interests, or observed behavior about the item. Based on the user's profile, a recommender system can anticipate whether or not a particular user will like an item. Both service providers and users benefit from recommender systems. They lower the expenses of searching for and selecting things in an online buying environment. Recommendation systems have also been shown to increase the quality and speed of decision-making. Recommender systems increase revenue in an e-commerce context since they are an efficient way of selling more things. Recommender systems in scientific libraries assist users by allowing them to go beyond catalogue searches. As a result, the importance of employing efficient and accurate recommendation algorithms inside a system that provides consumers with relevant and dependable recommendations cannot be overstated.

Different techniques of Recommender systems

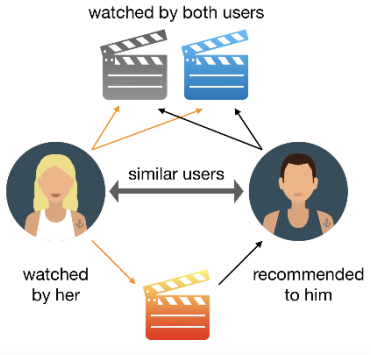
The use of efficient and accurate recommendation techniques is very important for a system that will provide good and useful recommendation to its individual users.



The content-based technique is a domain-specific algorithm that focuses on analysing item features in order to provide predictions. When it comes to recommending documents such as web pages, publications, and news, the content-based filtering strategy is the most effective. The content-based filtering technique makes recommendations based on user profiles and features collected from the content of items the user has already evaluated. The user is recommended things that are primarily connected to the positively scored items. These methods provide suggestions by learning the underlying model using statistical analysis or machine learning. Because other users' profiles do not influence recommendation, the content-based filtering technique does not require them. The main disadvantage of this method is that it necessitates a thorough understanding and explanation of the characteristics of the items in the profile.



Collaborative filtering is a domain-independent prediction tool for content such as movies and music that cannot be effectively represented by metadata. The collaborative filtering technique works by storing user preferences for items in a database (user-item matrix). It then makes recommendations by matching users with related interests and preferences based on profile similarities. Users who fall under this category form a neighbourhood. A user receives suggestions for things that he has not yet evaluated but that have already received positive feedback from other users in his area. Recommendations that are produced by CF can be of either prediction or recommendation. Prediction is a numerical value, Rij, expressing the predicted score of item j for the user i, while Recommendation is a list of top N items that the user will like the most. The technique of collaborative filtering can be divided into two categories: memory-based and model-based.



II. Literature Survey

https://reader.elsevier.com/reader/sd/pii/S1110866515000341?token=470204C8AF6340BAAD65144DC6D5BCE22F22C0CEC1381053EF2C589BBFF983921CE0786FA08A2F21572F2A3381889C97&originRegion=eu-west-1&originCreation=20211210112528
This paper explores the different characteristics and potentials of different prediction techniques in recommendation systems in order to serve as a compass for research and practice in the field of recommendation systems.

https://www.cs.ubc.ca/~tmm/courses/infovis/projects/deepansha-lucie-niloofar/update.pdf
In this article we explored and extract knowledge from the publicly available movie datasets. Results of this work provided an insight which can be used for designing movie recommender systems.

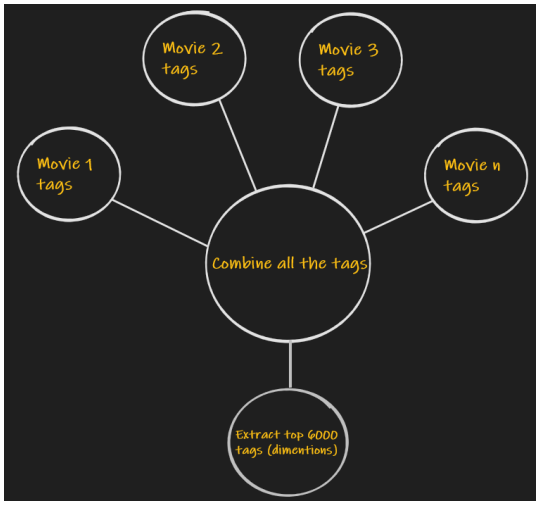
III. Methodology

1. First extraction and pre-processing the data is done, all the movies are left with an attribute called tags. Tag is a list of all the important strings of a movie, example actors, casts, summary, genre etc.

	budget	genres	homepage	id	keywords	original_language	original_title	overview
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond's past sends him o...

	title	tags
0	Avatar	[Avatar, SamWorthington, ZoeSaldana, Sigourney...
1	Pirates of the Caribbean: At World's End	[Pirates, of, the, Caribbean:, At, World's, En...
2	Spectre	[Spectre, DanielCraig, ChristophWaltz, LéaSeyd...
3	The Dark Knight Rises	[The, Dark, Knight, Rises, ChristianBale, Mich...
4	John Carter	[John, Carter, TaylorKitsch, LynnCollins, Sama...

2. Now we extract top 6000 most recurring strings from all the tags. Make a matrix of (size number of movies X 6000). Each row of this matrix is a movie and each column is a tag. Each element of the matrix tells how many times a particular tag is reoccurring the tag of that movie.



	D1 (spaceship)	D2 (love)	D2 (death)
movie 1	3	0	0
movie 2	0	2	1
movie 3	0	1	0
movie n	1	0	0

For this we used Count veectorizer from the sklearn library

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 6000, stop_words='english')
```

```
#CountVectorizer converts a collection of text documents to a matrix of token counts.
```

```
matrix = cv.fit_transform(movies['tags']).toarray()
```

```
matrix
```

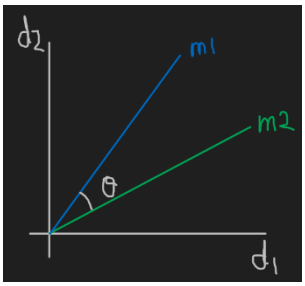
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

Some of the dimentions are:

```
cv.get_feature_names()
```

```
 'aristocrat',
 'arizona',
 'arm',
 'armageddon',
 'armi',
 'armiehamm',
 'armor',
 'armsdeal',
 'army',
 'arnold',
 'arnoldschwarzenegg',
 'arrang',
 'arrangedmarriag',
 'array',
 'arrest',
 'arriv',
 'arrives',
 'arrog',
```

3. Based on this matrix we calculate distance between every possible pair of movies. Distance is calculated in terms of cosine distance. Hence distance is directly proportional to similarity.



This uses cosine similarity function and is used as follows

```
from sklearn.metrics.pairwise import cosine_similarity
#returns dot product
```

```
similarity = cosine_similarity(matrix)
```

```
similarity
```

```
array([[1.        , 0.23310861, 0.18852561, ..., 0.04997225, 0.        ,
        0.        ],
       [0.23310861, 1.        , 0.17253244, ..., 0.03429972, 0.01889822,
        0.01889822],
       [0.18852561, 0.17253244, 1.        , ..., 0.01849317, 0.        ,
        0.        ],
       ...,
       [0.04997225, 0.03429972, 0.01849317, ..., 1.        , 0.03241019,
        0.04861528],
       [0.        , 0.01889822, 0.        , ..., 0.03241019, 1.        ,
        0.07142857],
       [0.        , 0.01889822, 0.        , ..., 0.04861528, 0.07142857,
        1.        ]])
```

4. Then we develop the recommender function that sorts all the movies on the basis of similarity matrix and give the top recommendations.

```
def recommend(movie):
    movie_index = movies[movies['title'] == movie].index[0]
    distances = similarity[movie_index]
    movie_list = sorted(list(enumerate(distances)),reverse=True,key=lambda x:x[1])[1:21]

    for i in movie_list:
        print(movies.iloc[i[0]].title)
```

```
def recommend2():
    #num_of_movies = input('Number of movies: ')
    a = [movie for movie in input("Enter movies: ").split(", ")]
    num_of_movies = len(a)
    #movie_index_list = [movies[movies['title'] == movie].index[0]]
    movie_index_list = [movies[movies['title'] == movie].index[0] for movie in a]
    x = np.linspace(0, 0, 6000)
    for i in movie_index_list:
        x = x + matrix[i]
    #cummulative vector of all movies
    List=[]
    for i in range(4806):
        t = np.dot(x, matrix[i])
        List.append(t)
    movie_list = sorted(list(enumerate(List)),reverse=True,key=lambda x:x[1])[num_of_movies:10+num_of_movies]

    for i in movie_list:
        print(movies.iloc[i[0]].title)
```

IV. Experimentations and Results

```
recommend("Iron Man")
```

```
Iron Man 3
Iron Man 2
Avengers: Age of Ultron
Guardians of the Galaxy
The Avengers
Captain America: Civil War
Ant-Man
X-Men
X-Men Origins: Wolverine
X-Men: The Last Stand
X2
The Helix... Loaded
The Wolverine
The Incredible Hulk
Fantastic Four
X-Men: Days of Future Past
Star Wars: Episode I - The Phantom Menace
Man of Steel
Captain America: The First Avenger
The Time Machine
```

```
recommend2()
```

```
Enter movies: Titanic, Captain America: The First Avenger, The Dark Knight Rises
The Dark Knight
Captain America: The Winter Soldier
In the Heart of the Sea
Batman Begins
Kiss of Death
The Net
Batman Forever
Run All Night
The Good German
Frequency
```

V. Conclusions

Successfully implemented movie recommender system using Jupyter Notebook. Data set plays a huge role in the recommendation process. The types of attributes and their number impacts the recommendations by a lot.

VI. References

- https://www.cs.ubc.ca/~tmm/courses/infovis/projects/deepansha-lucie-niloofar/update.pdf
- https://reader.elsevier.com/reader/sd/pii/S1110866515000341?token=470204C8AF6340BAAD65144DC6D5BCE22F22C0CEC1381053EF2C589BBFF983921CE0786FA08A2F21572F2A3381889C97&originRegion=eu-west-1&originCreation=20211210112528
- https://en.wikipedia.org/wiki/Recommender_system#:~:text=A%20recommender%20system%2C%20or%20a,would%20give%20to%20an%20item.
- https://www.analyticsvidhya.com/blog/2021/07/recommendation-system-understanding-the-basic-concepts/
- https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109
- https://data-flair.training/blogs/data-science-r-movie-recommendation/
- https://www.kaggle.com/rounakbanik/the-movies-dataset
- https://scikit-learn.org/stable/