VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme : Integrated MTech Specialization in Business Analytics

Course Title : Foundation of Data Analytics

Course Code : CSE3505

Slot : F2

Title

Content-Based Movie Recommendation System

Team Members:

P B S S Jaswanth | 19MIA1039

Vamsidhar S | 19MIA1074

Mansoor Khan Lodi | 19MIA1094

Shashank R | 19MIA1105

Faculty: Dr. S. Brindha **Sign:**

Date:

ABSTRACT

The rapid growth of data collection has led to a new era of information. Data is being used to create more efficient systems and this is where Recommendation Systems come into play. Recommendation systems are becoming increasingly important in today's hectic world. People are always in the lookout for products/services that are best suited for them. Therefore, the recommendation systems are important as they help them make the right choices, without having to expend their cognitive resources. Recommendation Systems are a type of information filtering systems as they improve the quality of search results and provides items that are more relevant to the search item or are related to the search history of the user. They are used to predict the rating or preference that a user would give to an item. Almost every major tech company has applied them in some form or the other: Amazon uses it to suggest products to customers, YouTube uses it to decide which video to play next on autoplay, and Facebook uses it to recommend pages to like and people to follow. Moreover, companies like Netflix and Spotify depend highly on the effectiveness of their recommendation engines for their business and success.

For our project, we will be using **Content Based Filtering for Movie Recommendation System**- They suggest similar items based on a particular item. This system uses item metadata, such as genre, director, description, actors, etc. for movies, to make these recommendations. The general idea behind these recommender systems is that if a person liked a particular item, he or she will also like an item that is similar to it. In this recommender system the content of the movie (overview, cast, crew, keyword, tagline etc) is used to find its similarity with other movies using Cosine similarity. Then the movies that are most likely to be similar are recommended.

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computing Science and Engineering

VIT Chennai

Vandalur - Kelambakkam Road, Chennai - 600 127

FALL SEM 22-23

Worklet details

Programme	Integrated MTech in CSE with Specialization in Business Analytics	
Course Name / Code	CSE3505	
Slot	F2	
Faculty Name	Dr. S. Brindha	
Component	J – Component	
J Component Title	Content-Based Movie Recommendation System	
Team Members Name | Reg. No	PBSS Jaswanth	19MIA1039
	Vamsidhar S	19MIA1074
	Mansoor khan lodi	19MIA1094
	Shashank R	19MIA1105

Team Members Contributions – Tentatively planned for implementation:

Worklet Tasks	***Contributor's Names***
Information & Data gathering	PBSS Jaswanth
Preprocessing	Mansoor khan lodi & Vamsidhar S
Data visualization	Shashank R
Model building & results interpretation	PBSS Jaswanth & Shashank R
GUI Implementation	Mansoor khan lodi
Technical Report writing	Mansoor khan lodi, Shashank R, Vamsidhar S, PBSS Jaswanth
Presentation preparation	Vamsidhar S

TABLE OF CONTENTS

1. Introduction

In the current post COVID-19 pandemic world, with major OTT (Over The Top) platforms such as Amazon Prime Video, Netflix etc., being a part and parcel of the lives of people, it is a must for the OTT platforms to have good recommendation algorithms. There can be an increase in churn rate due to poor recommendation algorithms as they fail to recommend things that matches the expectations of the target audience thus dissolving the positive opinions they have on the platforms. So, recommendation algorithms are the backbone of OTT platforms. Not only that, even they support other entities such as e-commerce sites, food delivery applications etc.

We might have observed that whenever we watch a movie of a particular genre, other movies of similar genre get recommended to us in OTTs. Recommendation algorithms are the main reason behind this behavior. These recommendation algorithms of OTT platforms had thus provoked our interest into taking up this topic for our project i.e., Movie recommendation system.

Problem Statement

For building a recommender system from scratch, we face several different problems. Currently there are a lot of recommender systems based on the user information, so what should we do if the website has **not gotten enough users**. After that, we will solve the representation of a movie, which is how a system can understand a movie. That is the precondition for comparing similarity between two movies. Movie features such as genre, actor and director is a way that can categorize movies.

Objectives:

Regarding this project, our main goal is to implement a content-based movie recommendation system. To achieve this goal, we will be going through the following steps:

1) Performing text vectorization using 2 algorithms namely – CountVectorizer and TF-IDF and then taking the help of the cosine similarity measure for generating raw recommendations.

2) GUI implementation to represent our recommender system in a more interactive way.

Challenges:

The main challenges to our project are:

1) The ability to provide more accurate recommendations i.e., the overall accuracy of the model.

2) Complexity of the dataset.

2. Literature Review

Meenu Gupta et al. [1] used KNN algorithms and collaborative filtering in order to increase the accuracy of results as compared to content-based filtering. A collaborative filtering technique combines cosine similarity with the knearest neighbor approach, which alleviates many of the drawbacks associated with content-based filtering. However, it cannot handle fresh items since it hasn't seen them during training.

Hrisav Bhowmick et al. [2] have implemented eight different methods for recommending movies. An example of a genre-based recommendation technique was that movies associated with a particular genre were checked first, then based on the scores, recommended. In genre-based recommendation, however, there remains a high chance that the recommended movies may not be liked by the target user since the recommendation is based on only genre, not user profile similarity. Using the Pearson Correlation Coefficient Based recommended system the similarity between users can be easily determined, but it is a long formula-based method that requires a lot of computation and memory.

Bagher Rahimpour Cami et al. [3] propose a content-based movie recommendation system that predicts movie preferences based on temporal user preferences. In the proposed method, the content attributes of rated movies (for each user) are incorporated into a Dirichlet Process Mixture Model to infer user preferences and provide a proper recommendation list.

Fisk [4] The author proposed a method where the movie recommendation system is based on the principle of social filtering. And Choi and Han [5] proposed a different collaborative filtering approach based on the category correlation of contents.

Good et al. [6] The authors proposed a model to alleviate information overload by using Information filtering agents and collaborative filtering. And Adomavicius and Kwon [7], The authors proposed the similarity-based approach and the aggregation function-based approach.

Movie Recommendation System Using Genome Tags and Content-Based Filtering by **Syed M. Ali, Gopal K. Nayak, Rakesh K. Lenka & Rabindra K. Barik**, Advances in Data and Information Sciences, 2018 [8] used Principal component analysis, Pearson correlation techniques. In which the system takes movieId as an input and recommends top five similar movies based on it.

Existing Methodologies Limitations:

The previous recommendation systems had certain gaps in them such as:

- Since it is based on the user ratings, it does not recommend any new product or items.
- Products or items which already exist but have not been rated by any user will not be considered for recommendation to a new user.

- A large amount of computation power is used.

Therefore, this motivates us to provide new model for the society:

- The method of cosine similarity is used to determine how similar documents are, irrespective of their size.
- Every item new and old, rated, and non-rated are recommended to the user based on their input.

3. Dataset and Tool to be used (Details)

The dataset we have chosen for our project is extracted from The Movie Database (TMDB) and contains the details of 5000 movies. The whole dataset comprises of two sub datasets i.e., credits and movies.

The credits' part consists of the following features: -

- movie_id - A unique identifier for each movie.
- cast - The name of lead and supporting actors.
- crew - The name of Director, Editor, Composer, Writer etc.

The movies dataset has the following features: -

- budget - The budget in which the movie was made.
- genre - The genre of the movie, Action, Comedy, Thriller etc.
- homepage - A link to the homepage of the movie.
- id - This is in fact the movie_id as in the first dataset.
- keywords - The keywords or tags related to the movie.
- original_language - The language in which the movie was made.

- original_title - The title of the movie before translation or adaptation.
- overview - A brief description of the movie.
- popularity - A numeric quantity specifying the movie popularity.
- production_companies - The production house of the movie.
- production_countries - The country in which it was produced.
- release_date - The date on which it was released.
- revenue - The worldwide revenue generated by the movie.
- runtime - The running time of the movie in minutes.
- status - "Released" or "Rumored".
- tagline - Movie's tagline.
- title - Title of the movie.
- vote_average - average ratings the movie recieved.
- vote_count - the count of votes recieved.

These two sub datasets will be merged in our project on title basis to generate the final dataset on which we will be making recommendations on.

- Here's the G-Drive dataset link: https://drive.google.com/drive/folders/1PuxYjzyaHZ0FjpUWvIeusmsoEhsTK1Gl?usp=sharing

Tools:

1) Jupyter Notebook or Google Colab for IDE

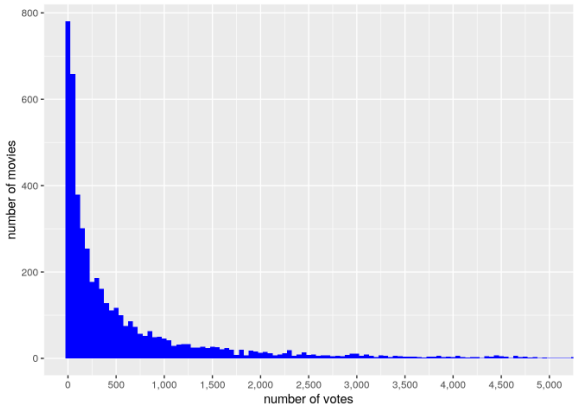
2) Python being the programming language

3) Python libraries like scikit-learn, pickle etc. for creating the recommendation system as well as the GUI.

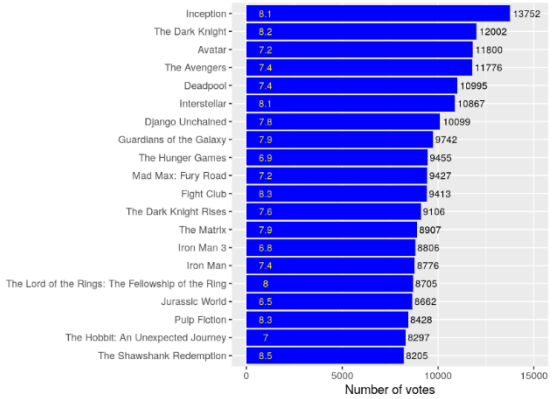
4) Using ‘Streamlit’ which is an open-source python-based framework and Pycharm platform for the GUI implementation.

5. Data Visualization

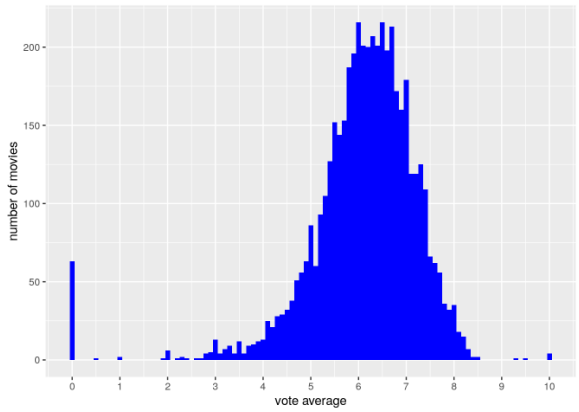
Number of votes



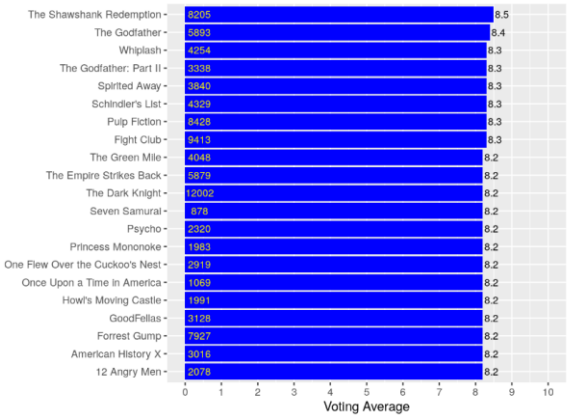
Movies with highest no. of votes



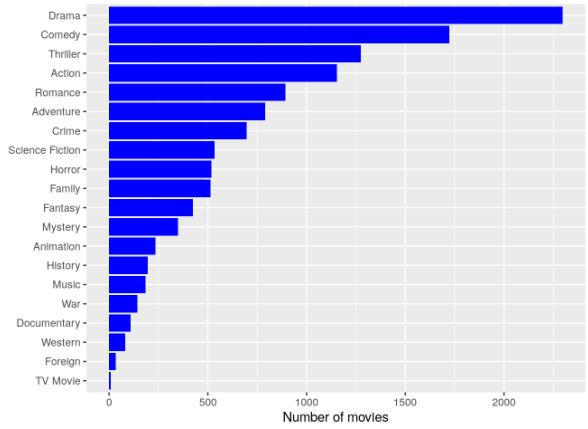
Vote average vs No. of votes



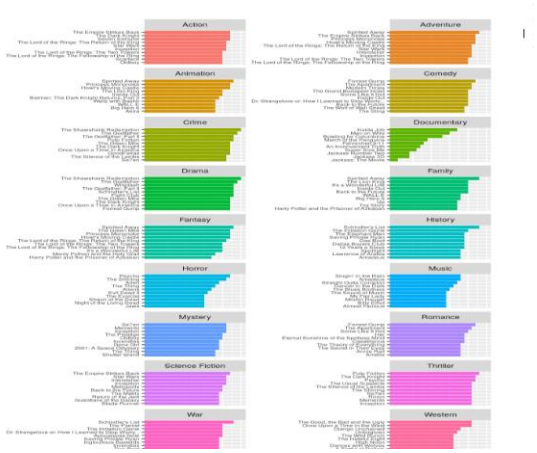
Movies with highest vote average



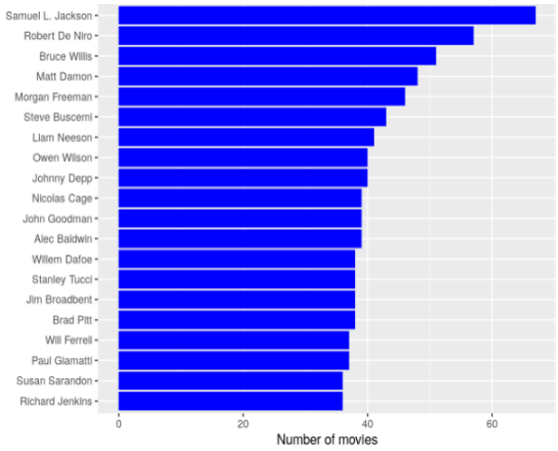
No. of movies by genre



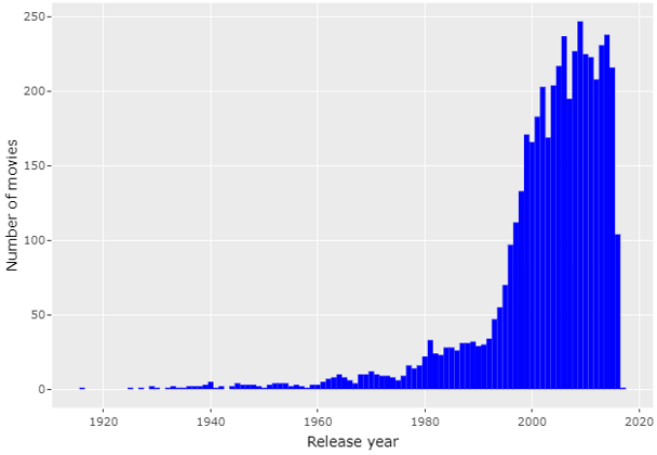
Highest rated movie by genre



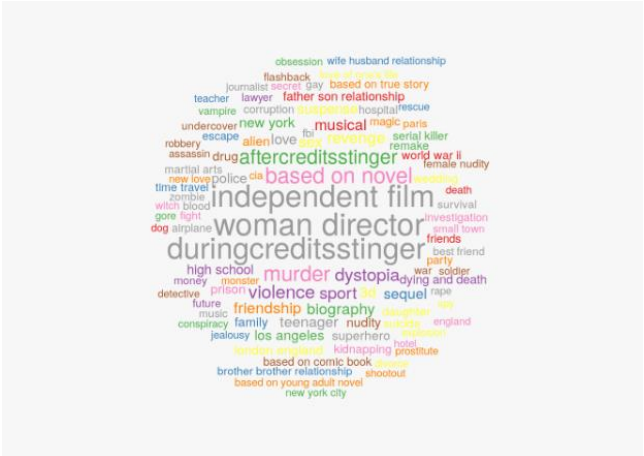
Most actor and director appearances



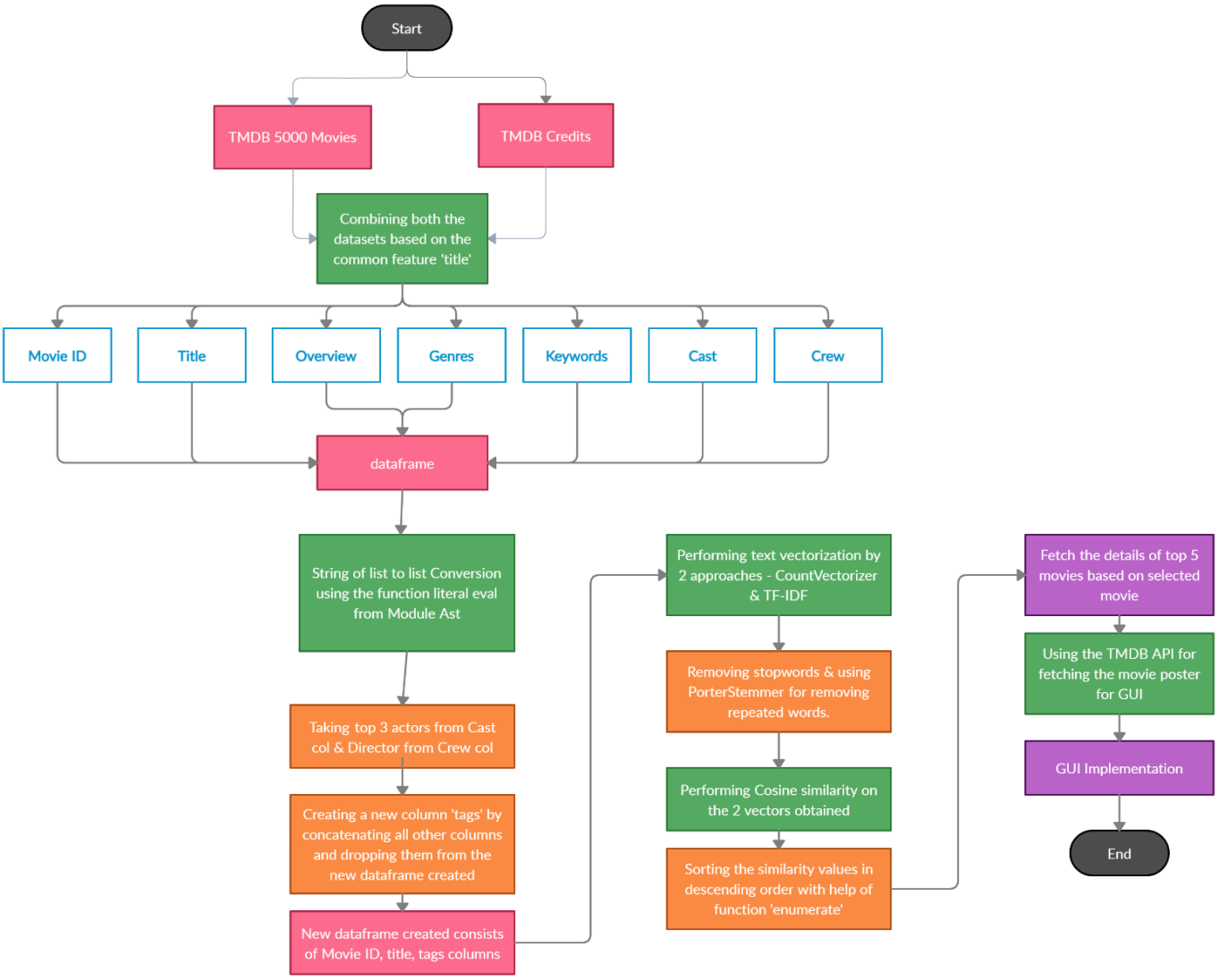
No. of movies vs Release year



Most used keywords



4. Proposed Methodology



- We have taken the dataset from The Movie Database (TMDB) which consists of 2 parts namely – ‘TMDB 5000 movies’ and ‘TMDB credits’
- We will combine these 2 datasets based on a common feature between them – ‘Title’
- For our recommender system, we will create a dataframe in which we only take the columns which adds value to our prediction system which includes ‘movie_id’, ‘title’, ‘overview’, ‘genres’, ‘keywords’, 'cast', ‘crew’ and remove all the unnecessary columns.

- We will create a helper function to convert the String of List to List using the function '**Literal eval'** from module 'Ast'.
- From 'Cast' column – we will take the top 3 actors from the list.
- From 'Crew' column – we will extract only the names of the 'Director'.
- Then we will remove the spaces in each single entity from all the columns.
- Next, we will create a new column called 'tags' which is made by concatenating all the other columns which includes 'movie_id', 'title', 'overview', 'genres', 'keywords', 'cast', 'crew' and drop these columns from the new dataframe created.
- Creating a new dataframe which consists of Movie id, title, and tags.
- We will perform text Vectorization using 2 algorithms namely **Count Vectorizer** and **TF-IDF** and remove stopwords from the movies data.
- We will use '**PorterStemmer**' from 'nltk' library which is a process for removing the commoner morphological and inflexional endings from words in English.
- Then, we will perform **Cosine Similarity** on the final vectors achieved through the 2 algorithms.
- Next, we will sort the movies in descending order w.r.t the distance, in order to have the movies with highest cosine similarity at the top: but in this process we lose the index value of the movie to avoid we will use the function '**enumerate**' which converts a data collection object into an enumerate object. Enumerate returns an object that contains a counter as a key for each value within an object, making items within the collection easier to access.
- We will fetch the index position of the movie to be searched.
- Finally, we will get the details of the top 5 movies based on our selected movie.

- API stands for “Application Programming Interface.” An API is a messenger that delivers your request to the provider that you are requesting it from and then delivers the response back to you. In our case the provider is TMDB. TMDB has a huge collection of movies data, from which the system can fetch the information that it needs. To use TMDB API, an API key has to be generated after creating an account on TMDB.
- For our recommender system, we will fetch the movie poster based on the selected movie ID from the dataset using the **TMDB API**.
- For **GUI** implementation, we will dump both movies.pkl and similarity.pkl in write binary mode from our ipynb file, then load and open both movies.pkl and similarity.pkl in read binary mode in Pycharm and use Streamlit for GUI implementation.

5. Algorithms / Techniques description

A) Count vectorizer:

It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text.

Text data demands special preparation before employing it for predictive modeling. The text has to be parsed to pull out words, termed tokenization. Then the words require to be encoded as integers or floating-point values for applying as input to a machine learning process, named feature extraction or also vectorization. The CountVectorizer offers an uncomplicated technique to both build a vocabulary of known words and tokenize a series of text documents, however as well as to predetermine recent documents utilizing that vocabulary. CountVectorizer’s fit.tranform method is applied to calculate the number

of texts and will produce the converted matrix count_matrix into an array for more efficient insight. When the text input is introduced through the ‘count vectorizer’ function, it fetches a matrix of the number count of each word.

CountVectorizer is used as follows:

1. Construct an instance of the CountVectorizer object.

2. Request the fit () function so as to gather a vocabulary from one or more documents.

3. Call the transform () function on one or more documents as required to translate each as a vector. A converted vector is countered with a length of the complete vocabulary and an integer count for the number of times every word featured in the document

B) TF-IDF

Content based filtering approach filters the items based on the likings of the user. It gives result based on what the user has rated earlier. The method to model this approach is the Vector Space Model (VSM). It derives the similarity of the item from its description and introduces the concept of TF-IDF (Term Frequency-Inverse Document Frequency).

TFIDF is based on the logic that words that are too abundant in a corpus and words that are too rare are both not statistically important for finding a pattern. The Logarithmic factor in tfidf mathematically penalizes the words that are too abundant or too rare in the corpus by giving them low tfidf scores.

Tf(t)=frequency occurrence of term t in document / totalnumberoftermsindocument

If(t)= $log10$ (totalnumberofdocument / numberofdocumentscontainingtermt)

C) Cosine similarity:

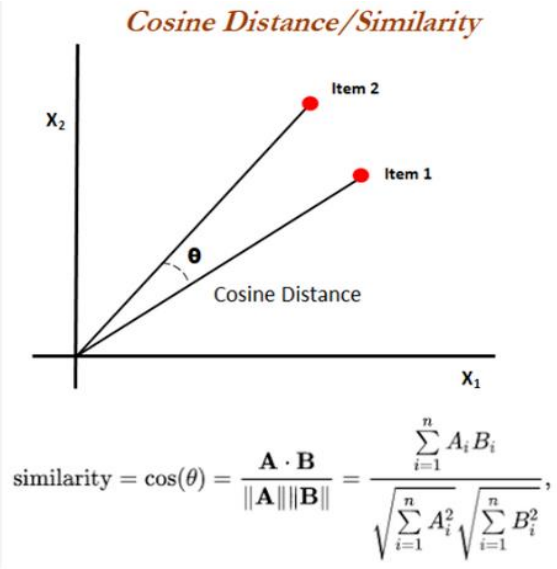
It is used to degree the similarity of vectors regardless of the scale. Cosine similarity is measured via way of means of utilizing the two non-zero vectors and acquiring the dot manufactured from the each and ultimately dividing it via way of means of the goods of Mathematically, it's miles in particular used for measuring the cosine of the perspective among non-0 vectors projected in a multidimensional space. The Cosine Similarity is better whilst the perspective is smaller and vice-versa. In Movie Recommendation System, Cosine similarity takes person evaluations on film as statistics and gives the result. Cosine(X1, X2) = X1. X2 / X1 * X2 where,

• X1.X2= product (dot) of the vectors 'X1' and 'X2'.

• X1 and X2 = period of the 2 vectors 'X1' and 'X2'.

• X1 * X2 = pass manufactured from the 2 vectors 'X1' and 'X2'.

• The Cosine similarity of non-0 vectors is measured in 'θ'.



$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

• If θ = 0°, the ‘X1’ and ‘X2’ vectors overlap, for this reason proving they're similar.

• If θ = 90°, the ‘X1’ and ‘X2’ vectors are dissimilar.

• Consider an instance to locate the similarity among vectors – ‘X1’ and ‘X2’, the use of Cosine Similarity.

• The ‘x’ vector has values, X1 = {1,0,0,0}, The ‘y’ vector has values, X2 = {3,2,0,5}

• The components for calculating the cosine similarity are - Cosine (X1, X2) = X1. X2 / X1 * X2

• X1.X2= 1*3+ 0*2 + 0*0 + 0*5 = 3

• X1 = √ (1)^2 + (0)^2 + (0)^2 + (0)^2 = 1

• X2 = √ (3)^2 + (2)^2 + (0)^2 + (5)^2 = 6.16

• ∴ Cosine (X1, X2) = 3 / (1*6.16) = 0.49

• The dissimilarity among the 2 vectors ‘X1’ and ‘X2’ is given via way of means of

• ∴ Dis (X1, X2) = 1 - Cosine(X1, X2) = 1 - 0.49 = 0.51

Advantages:

• Two similar statistics gadgets that's a way aside via way of means of the Euclidean distance because of the scale have a smaller perspective among them.

• Hence, Cosine similarity is beneficial.

• If the angle is smaller, the similarity will be higher.

6. Experimental Results

The formula used to measure how similar the movies are based on their similarities of different properties. Mathematically, it shows the cosine of the angle of two vectors projected in a multidimensional space. The cosine similarity is very beneficial since it helps in finding similar objects.

(I) Using **Count Vectorizer** for Text Vectorization, showing top 10 movie ID's and similarity values based on the cosine similarity values achieved in descending order w.r.t the first movie of index 0.

```
In [53]: sorted(list(enumerate(similarity[0])),reverse=True,key = lambda x: x[1])

Out[53]: [(0, 1.0),
          (539, 0.26089696604360174),
          (1194, 0.2581988897471611),
          (260, 0.25110592822973776),
          (1216, 0.24944382578492943),
          (507, 0.24846467329894412),
          (1444, 0.24397501823713333),
          (1920, 0.243599382882345),
          (582, 0.24147264420814754),
          (3730, 0.23904572186687872),
          (74, 0.22677868380553634),
```

The top 5 recommended movies are –

```
In [56]: recommend('Gandhi')

         Gandhi, My Father
         The Wind That Shakes the Barley
         A Passage to India
         Guiana 1838
         Ramanujan
```

(II) Using **TF-IDF** for Text Vectorization, showing top 10 movie ID's and similarity values based on the cosine similarity values achieved in descending order w.r.t the first movie of index 0.

```
In [59]: sorted(list(enumerate(similarity1[0])),reverse=True,key = lambda x: x[1])

Out[59]: [(0, 1.0000000000000002),
          (2409, 0.2519930130215996),
          (3730, 0.22559347561842458),
          (582, 0.19390717213891556),
          (1216, 0.18848538168904172),
          (3608, 0.17684322332408908),
          (47, 0.17304643775881712),
          (778, 0.16571834555284645),
          (1204, 0.16306831442162661),
          (539, 0.15630094071167544),
```

The top 5 recommended movies are –

```
In [61]: recommend1('Gandhi')

         Gandhi, My Father
         Dil Jo Bhi Kahey...
         Neal 'n' Nikki
         A Passage to India
         Guiana 1838
```

7. Discussion on Results

Based on the above results obtained through the two text vectorizations techniques namely Count Vectorizer and TF-IDF we can see that there is only a slight difference between the movies recommended, as the plot of the movies recommended by both these approaches are almost similar. Since the Count Vectorizer focuses on the frequency of words while the TF-IDF not only focuses on the frequency of words present in the corpus but also provides the importance of the words.

8. Model Evaluation

The common way to assess the performance of a recommender system would be through standard metrics such as Accuracy, Precision or Recall. However, these metrics require

ground truth knowledge about which recommendations are correct, which is hard to obtain at a large scale in our specific problem setting i.e., Content-based Recommender system.

9. Conclusion

Recommendations systems have become the most essential fount of a relevant and reliable source of information in the world of internet. Simple ones consider one or a few parameters while the more complex ones make use of more parameters to filter the results and make it more user friendly.

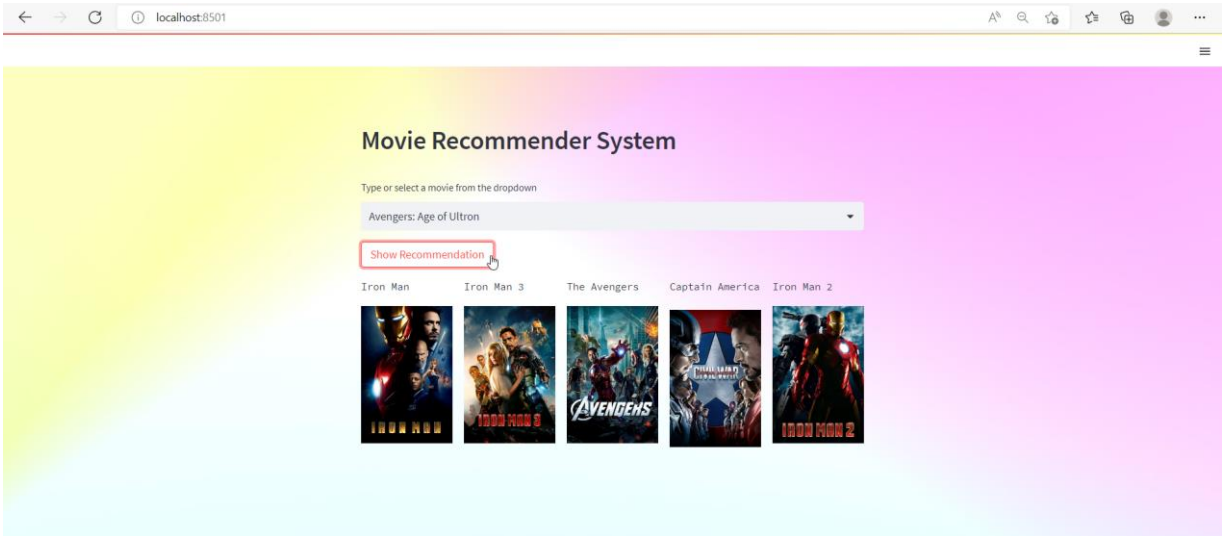
We have illustrated the modelling of a movie recommendation system by making the use of content-based filtering in the movie recommendation system. For text Vectorization, 2 algorithms are implemented in this model namely Count Vectorizer and TF-IDF along with the principle of cosine similarity distance measure as it gives more accuracy than the other distance metrics and the complexity is comparatively low too.

When a user wants a list of movies similar to a movie that they had previously watched. The the user needs to search on our recommender system for the movie name that he/she has already watched, and our recommender system will recommend the top five movies that are most similar to the searched movie. This feature will save user's time which otherwise would have been wasted on finding a movie that he/she may or may not like. Every month several movies are released, the movies database only gets bigger and bigger. This would help the system to provide a more accurate recommendation to the user and in turn increase customer satisfaction.

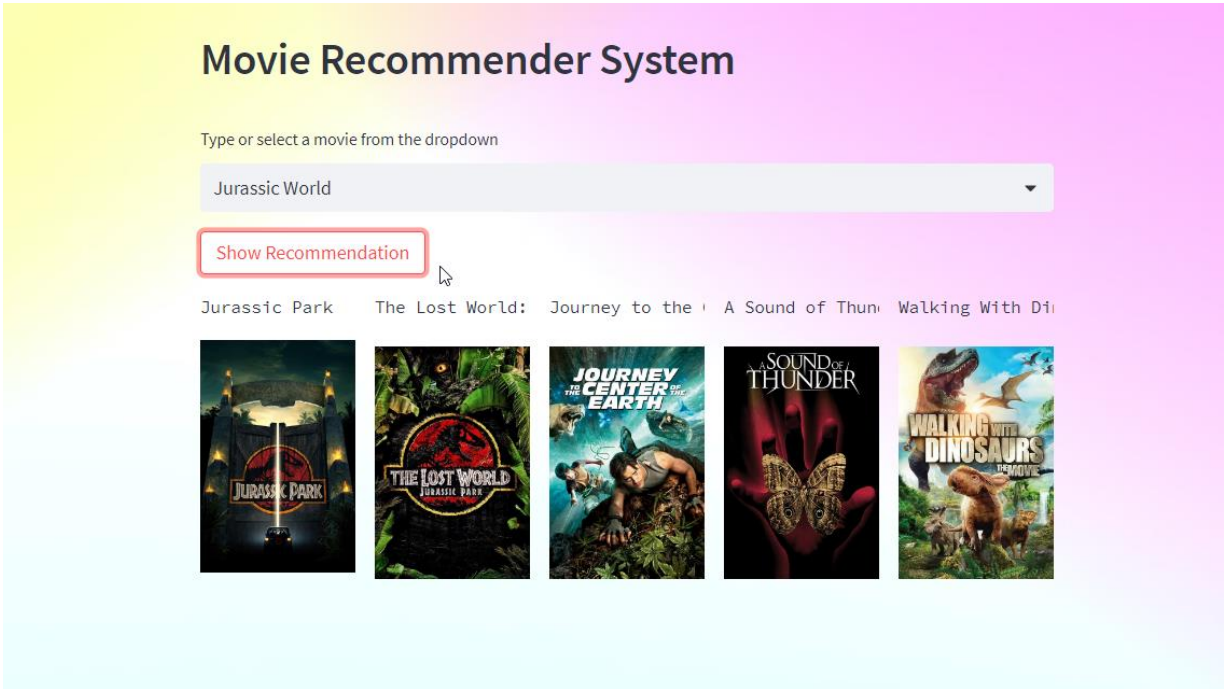
10. Screenshots

GUI Implementation –

Movie: Avengers age of Ultron



Movie: Jurassic World



11. Scope for Future Work

In the past few years, the development of machine learning, large-scale network and high-performance computing is promoting new development in this field. We will consider the following aspects in future work.

- Use collaborative filtering recommendation. After getting enough user data, collaborative filtering recommendation will be introduced. Collaborative filtering is based on the social information of users, which will be analyzed in the future research.

- We will introduce more precise and proper features of movie. Typical collaborative filtering recommendation use the rating instead of object features. In the future we should extract features such as color and subtitle from movie which can provide a more accurate description for movie.

- Introduce user dislike movie list. The user data is always useful in recommender systems. In the future we will collect more user data and add user dislike movie list. We will input dislike movie list into the recommender system as well and generate scores that will be added to previous result. By this way we can improve the result of recommender system.

- Introduce machine learning. For future study, dynamic parameters will be introduced into recommender system, we will use machine learning to adjust the weight of each feature automatically and find the most suitable weights.

REFERENCES

[1] Yilin Zhang, Lingling Zhang, Movie Recommendation Algorithm Based on Sentiment Analysis and LDA, Procedia Computer Science, Volume 199, 2022, Pages 871-878, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2022.01.109.Avinash Bharadwaj, Brinda Ashar, June 2020. Source Based Fake News Classification using Machine Learning (e-ISSN: 2319-8753, p-ISSN: 2320-6710)

[2] N Pavitha, Vithika Pungliya, Ankur Raut, Roshita Bhonsle, Atharva Purohit, Aayushi Patel, R Shashidhar, Movie recommendation and sentiment analysis using machine learning, Global Transitions Proceedings, Volume 3, Issue 1, 2022, Pages 279-284, ISSN 2666-285X, https://doi.org/10.1016/j.gltp.2022.03.012.

[3] H. -W. Chen, Y. -L. Wu, M. -K. Hor and C. -Y. Tang, "Fully content-based movie recommender system with feature extraction using neural network," 2017 International Conference on Machine Learning and Cybernetics (ICMLC), 2017, pp. 504-509, doi: 10.1109/ICMLC.2017.8108968.

[4] Ali, S.M., Nayak, G.K., Lenka, R.K., Barik, R.K. (2018). Movie Recommendation System Using Genome Tags and Content-Based Filtering. In: Kolhe, M., Trivedi, M., Tiwari, S., Singh, V. (eds) Advances in Data and Information Sciences. Lecture Notes in Networks and Systems, vol 38. Springer, Singapore. https://doi.org/10.1007/978-981-10-8360-0_8.

[5] T. Rutkowski, J. Romanowski, P. Woldan, P. Staszewski, R. Nielek and L. Rutkowski, "A Content-Based Recommendation System Using Neuro-Fuzzy Approach," 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2018, pp. 1-8, doi: 10.1109/FUZZ-IEEE.2018.8491543.

[6] G. Sunandana, M. Reshma, Y. Pratyusha, M. Kommineni and S. Gogulamudi, "Movie recommendation system using enhanced content-based filtering algorithm based on user demographic data," 2021 6th International Conference on Communication and Electronics Systems (ICCES), 2021, pp. 1-5, doi: 10.1109/ICCES51350.2021.9489125.