Movie recommendation system using TF-IDF vectorization and Cosine Similarity

1PV. Snigdha, 2M. Naveen, 3S.Rahul
1student, 2student, 3student
1Sreenidhi institute of science and technology(SNIST),
2Sreenidhi institute of science and technology (SNIST),
3Sreenidhi institute of science and technology

Abstract - **The internet has widened the horizons of numerous areas to engage and share relevant information in recent years. As it is said, everything has its advantages and disadvantages, thus with the increase in the field comes data saturation and data extraction difficulties. The suggestion system is critical in overcoming this challenge. Its purpose is to improve the user's experience by providing quick and comprehensible suggestions. Because of its ability to provide improved amusement, a movie suggestion is vital in our personal interaction. Users might be recommended a collection of movies depending on their interests or the appeal of the films. A recommendation system is being used to make suggestions for things to buy or see. They comb through a big database of information to lead people to the things that can suit their demands. A recommender system, also known as a recommendation engine or platform, is a type of data filtering system that attempts to forecast a user's "rating" or "preference" for an item. They're mostly employed for business purposes. This project outlines a content-based movie recommendation model which uses TF-IDF vectorization and cosine similarity for providing users with generic options based on film popularity and/or theme.**

keywords - **Movie recommendation, Content-based recommendation system, TF-IDF vectorization, Cosine similarity, Data saturation, Data extraction**

I.   INTRODUCTION

It is quite tough for people to get content that they're truly fascinated by during this age of knowledge overload. it's also difficult for the content producer to form their material and stand out from the throng. to handle this inconsistency, numerous researchers and businesses have developed Recommender Systems. Recommender System's purpose is to link users and knowledge, to order to help users to locate information that's relevant to them, and to push information to particular users. All consumers and content providers get pleasure from this arrangement. People have relied on suggestions for each major and tiny choice since the dawn of civilization. The individual is going to be possible to adapt their viewpoint (recommendation) when it comes from a talented individual and also when over two or three persons advocate the identical thing. Recommendation systems emerged within the modern internet age, supporting the identical concept as before. Recommendation Systems are programs that make suggestions to end-users supported by their preferences or the preferences of comparable users. The above divides this same recommendation system into two major types: Content-Based Filtering Recommendation Systems and Collaborative Filtering Recommendation Systems. the subsequent sections will undergo each of those categories. These classifications are supported by similarity measures; however, we've progressed to more complex methodologies like Machine learning algorithms. With the promising performance of the recommender system in e-commerce, film, music, books, and news suggestions, it's now moved to other industries like tourism and banking. "A recommender system also referred to as a recommendation system, maybe a form of data filtering system that attempts to forecast a user's 'rating' or 'preference' for an item." After a forecast has been produced, the user is given recommendations or suggestions that are supported by the predictions' findings. There are many various styles of recommender systems, and not all of them are appropriate for each problem and circumstance.

II.   LITERATURE SURVEY

Abhishek Singh, Samyak Jain, j Shanmukh Rao, Uppalpati Yogendra Reddy, and Abhishek Rawat created this system by employing technologies such as matrix factorization and recollection algorithms, rather than the commonly utilized hybrid-based approach. They also used several packages which include TfidVectorizer, nltk, and others to train an emotional model that can transform a review which is in the form of text into vector file and determine if the feedback published was favorable or unfavorable. When a movie title is put into the finished product, related films are suggested. Javascript was used to achieve this. The cosine similarity measure is used to calculate document similarity by geometrically displaying the vectors on a multidimensional space [1]

N. Muthurasu, Kavitha coonjeevaram, and Nandhini Rengaraj used the Term-frequency Inverse document frequency approach are used in this study to vectorize a hybrid audiovisual recommendation engine. The similarity is measured using the cosine similarity approach. A web-based user interface is used to show the system to the user. Even with a limited data model, the system provides efficient predictions and correct suggestions. User characterization and recordkeeping, analytics

reports for creators and consumers, and data acquisition via web scraping are all planned for the future. It saves time for users, and future upgrades include a data analytics site that allows movie makers to study and track user performance and preferences for a certain genre/video. Faster and so more reliable recommendation engines expand market reach and provide a steady stream of repeat customers.[2]

J. Aswin and P. Sabari Ramkumar suggested a system to overcome the cold start problem and suggest movies to its consumers. A hybrid method is presented that combines content and collaborative-based approaches, including a similarity-based approach for content-based collaborative filtering, a model-based approach for user-based collaborative filtering, and a neighbor-based approach for item-based collaborative filtering. The total performance of the network is increased by combining different filtering approaches. To increase the accuracy of both the recommendation system, they applied two collaborative approaches: user-based and item-based. The object Collaborative filter is based on Bayesian customized ranking, whereas the consumer Collaborative filter is built upon the Pearson product technique correlation coefficient algorithm.[3]

Yu Zhu, Shibi He, Ziyu Guan, Jinhao Lin, Beidou Wang, Haifeng Liu, and Deng Cai concentrated mostly on the item cold-start problem in this study. Capturingcapturing users' opinions on a new item, including information (e.g. item characteristics) and first user ratings are useful. The suggested system in this research is a revolutionary item cold-start recommendation approach that takes advantage of both enhanced learning and item attribute information. They created consumer selection specific to item qualities and user rating history and then combined the data in an optimization method for user selection. We then construct reliable rating predictions for the remaining unselected users using the feedback ratings, users' past ratings, and item characteristics. The superiority of our suggested strategy over previous methods is demonstrated by experimental findings on two real-world datasets.[4]

III.   METHODOLOGY

In this section, we are going to give a brief walkthrough of the project from data collection to model suggestion.
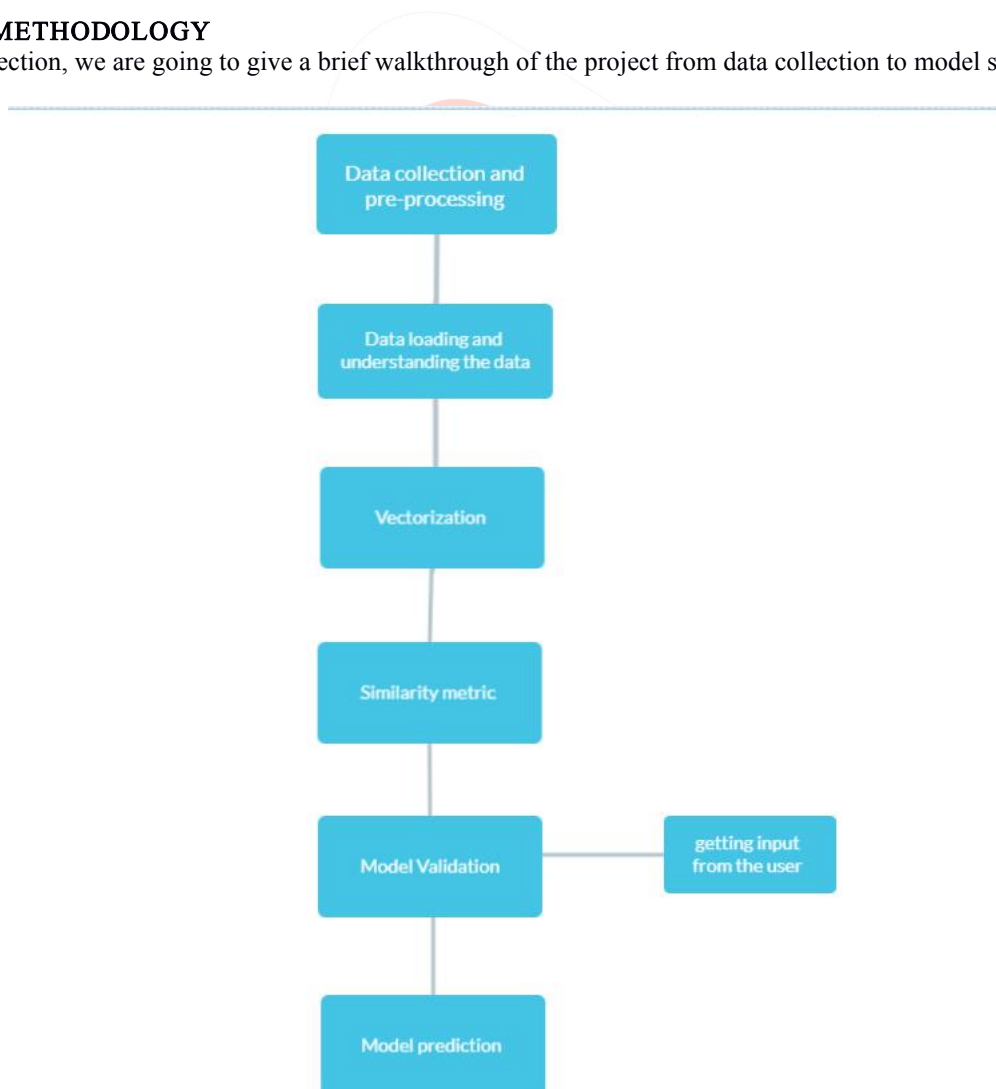
Fig 3.1 Workflow of the project

The model explained in this paper is implemented in three phases, The first phase consists of the Collection and analysis of data where we used a dataset containing roughly 5000 movies and cleaned the data for further analysis. In the second phase of the project, we primarily focused on vectorization and the calculation of the similarity score of the dataset used. For that, we used the functions TfidfVectorizer and cosine_similarity available on the scikit learn python library. The Sci-Kit Learn library

is the best place to go for machine learning algorithms because it contains nearly all types of ML (Machine Learning) algorithms for Python, making evaluations faster and easier. The final phase consists of validating the model and looking at the suggestion made by the model.

3.1 DATASET USED

The dataset was procured from Kaggle. Kaggle is a hub for datasets where the datasets are published by a community of data scientists and machine learning practitioners. It contains roughly around 4808 movies and 24 attributes. The attributes of the dataset are Index, Budget of the movie, Key Genres of the movie, Homepage of the movie's IMDB, the original language the movie was released in, Title of the movie, Overview/ Synopsis of the movie, Popularity, Production company, The country of production, Date of release, Revenue generated, Runtime of the movie, Status of the movie, Tagline, cast, crew and the director of the movie. The dataset was procured from Kaggle and can be found here

Fig 3.2 Dataset 1

Fig 3.3 Dataset 2

IV. IMPLEMENTATION

In this section, we are going to give a brief walkthrough of the project from data collection to model suggestion.

4.1 DEFINITIONS, CONCEPTS, AND SUPPLEMENTAL DATA:

4.1.1 PRE-PROCESSING THE DATA

The vast bulk of the material on the internet is guaranteed to have mistakes and blank spaces. The need to develop techniques for leveraging resources to make educated judgments has become critical in the drive for greater performance and dependability. In order to gain better insights, it is necessary to clean data before using it for predictive modeling. This necessitated some simple pre-processing of the Movie dataset we were working with.

1) **Converting the dataset from CSV format to a Pandas data frame:**

CSV files, also known as Comma-separated values are files where Commas are used to separate data. As the name indicates. It's a mechanism for applications that can't speak to one other directly to share structured data, such as the data of a spreadsheet. Here we convert the data from CSV to a pandas data frame to perform various arithmetic operations on the

database. Pandas is a python package that offers a variety of data structures and operations which can be used for manipulating numerical data. It is primarily used for importing and analyzing the data.

2) Replacing the Null values with Null string

When we concatenate strings together, we usually replace Null values in the dataset with Null strings. When a Null string is concatenated with a null value, the outcome is another null value, implying that the data we had before the concatenation is lost. We do so by using the for-in function available in python.

4.1.2 TF-IDF VECTORIZATION

Text vectorization is the process of converting text into a quantitative feature. It compares a phrase's "relative frequency" in a document to the consistency of that term across all papers. The TF-IDF (Term Frequency – Inverse Document Frequency) weight shows a phrase's relative importance in the document and throughout the corpus. Phrase Frequency (TF) is a measure that displays how frequently a phrase appears in a document. Due to document size disparities, a term may appear more frequently in a large document than in a short one. As a result, the document's length is usually separated by term frequency. TF-IDF is among the most extensively used text vectorizers, and the computation is straightforward. It distinguishes between the uncommon word heavier weight and the more frequent term reduced weight.

4.1.3 COSINE SIMILARITY

Here we calculate the cosine similarity using the Cosine_similarity function. Cosine similarity is a statistic for determining how similar papers are regardless of size. It estimates the cosine of the angle made of two vectors cast in a cross-dimensional space mathematically. Because of the cosine similarity, even if two comparable documents are separated by the Distance measure (considering the size of such documents), they are likely to be orientated closer together. The higher the cosine similarity, the smaller the angle. The measure is utilized in data mining, information retrieval, and text matching applications. In information retrieval, utilizing weighted TF-IDF and cosine similarity to swiftly find documents that are comparable to a search query is a typical strategy.

4.2 STAGES OF IMPLEMENTATION

Stage 1: Data preparation

After loading the data, here we print the first five rows from the downloaded data frame to observe the attributes of the data. Then we selected the relevant features required for an accurate recommendation. The key features are genres, keywords, tagline, cast, and director. Coming to the pre-processing part, we replaced the null values in the data with null strings. Finally, we combined the selected five key features.

```
['genres', 'keywords', 'tagline', 'cast', 'director']
```

Fig 4.1 Key features

```
0       Action Adventure Fantasy Science Fiction cultu...
1       Adventure Fantasy Action ocean drug abuse exot...
2       Action Adventure Crime spy based on novel secr...
3       Action Crime Drama Thriller dc comics crime fi...
4       Action Adventure Science Fiction based on nove...
                             ...
4798    Action Crime Thriller united states\u2013mexic...
4799    Comedy Romance  A newlywed couple's honeymoon ...
4800    Comedy Drama Romance TV Movie date love at fir...
4801       A New Yorker in Shanghai Daniel Henney Eliza...
4802    Documentary obsession camcorder crush dream gi...
Length: 4803, dtype: object
```

Fig 4.2 Combined features

Stage 2: VECTORIZATION OF THE DATA

In this step, we converted the text data into feature vectors using the function TfidVectorizer. Tfidvectorizer is a function found in the sklearn library. Are the dataset after it has been vectorized.

Stage 3: CALCULATING COSINE SIMILARITY

Here we calculate the cosine similarity using the Cosine_similarity function found in the sklearn library. Below seen is the similarity score matrix of the dataset.

Stage 4: MODEL VALIDATION AND SUGGESTION

When an input is given by the user a list is created with all the movies in the dataset after which the algorithm tries to find the closest match to the input given by the user. After finding the closest match, using the similarity score creates a list of similar movies. The movies are sorted based on their similarity score. Then a list of similar movies to the given input is printed.

4.3 PROCESS OF IMPLEMENTATION:

STEP 1: In the first step we take the input from the user by using the prompt, "Enter your favorite movie".

STEP 2: Here we create a list with all the movie names given in the dataset.

STEP 3: Then we find the close match to the movie name given by the user.

STEP 4: We find the closest match to the input given by the user following that.

STEP 5: Then we find the index of the movie with the title.

STEP 6: Then using we apply the similarity function on the index of the movie to calculate the cosine similarity of all movies in the dataset and create a list of similarity scores.

STEP 7: Then we sort the similarity scores using the lambda function available in python.

STEP 8: Finally, we print the name of similar movies based on the sorted similarity indices.

```
[(68, 1.0000000000000002), (79, 0.40890433998005965), (31, 0.31467052449477506), (7, 0.23944423963486405),
```

Fig 4.3 Sorted similarity scores

```
Enter your favourite movie name : iron man
```

Fig 4.4 Input given by the user

V.   RESULTS:

```
(0, 2432)      0.17272411194153
(0, 7755)      0.1128035714854756
(0, 13024)     0.1942362060108871
(0, 10229)     0.16058685400095302
(0, 8756)      0.22709015857011816
(0, 14608)     0.15150672398763912
(0, 16668)     0.19843263965100372
(0, 14064)     0.20596090415084142
(0, 13319)     0.2177470539412484
(0, 17290)     0.20197912553916567
(0, 17007)     0.23643326319898797
(0, 13349)     0.15021264094167086
(0, 11503)     0.27211310056983656
(0, 11192)     0.09049319826481456
(0, 16998)     0.1282126322850579
(0, 15261)     0.07095833561276566
(0, 4945)      0.24025852494110758
(0, 14271)     0.21392179219912877
(0, 3225)      0.24960162956997736
(0, 16587)     0.12549432354918996
(0, 14378)     0.33962752210959823
(0, 5836)      0.1646750903586285
(0, 3065)      0.22208377802661425
(0, 3678)      0.21392179219912877
(0, 5437)      0.1036413987316636
```

Figure 5.1 TF-IDF vectorized data

```
     :        :
(4801, 17266) 0.2886098184932947
(4801, 4835)  0.24713765026963996
(4801, 403)   0.17727585190343226
(4801, 6935)  0.2886098184932947
(4801, 11663) 0.21557500762727902
(4801, 1672)  0.1564793427630879
(4801, 10929) 0.13504166990041588
(4801, 7474)  0.11307961713172225
(4801, 3796)  0.3342808988877418
(4802, 6996)  0.5700048226105303
(4802, 5367)  0.22969114490410403
(4802, 3654)  0.262512960498006
(4802, 2425)  0.24002350969074696
(4802, 4608)  0.24002350969074696
(4802, 6417)  0.21753405888348784
(4802, 4371)  0.1538239182675544
(4802, 12989) 0.1696476532191718
(4802, 1316)  0.1960747079005741
(4802, 4528)  0.19504460807622875
(4802, 3436)  0.21753405888348784
(4802, 6155)  0.18056463596934083
(4802, 4980)  0.16078053641367315
(4802, 2129)  0.3099656128577656
(4802, 4518)  0.16784466610624255
(4802, 11161) 0.17867407682173203
```

Figure 5.2 Remaining Vectorized data

```
[[1.         0.07219487 0.037733   ... 0.         0.         0.        ]
 [0.07219487 1.         0.03281499 ... 0.03575545 0.         0.        ]
 [0.037733   0.03281499 1.         ... 0.         0.05389661 0.        ]
 ...
 [0.         0.03575545 0.         ... 1.         0.         0.02651502]
 [0.         0.         0.05389661 ... 0.         1.         0.        ]
 [0.         0.         0.         ... 0.02651502 0.         1.        ]]
```

Figure 5.3 Cosine similarity score matrix

Figure 5.4 Recommended list of movies

From the images, The algorithm has successfully vectorized the data using TF-IDF vectorization and also calculated the similarity scores using cosine similarity. we can also observe that when batman is given as an input to the algorithm, a list of movies that have a high similarity score gets printed as the output. Here the suggestions given by the algorithm are batman (the closest match to the input from the dataset), Batman Returns, Batman and Robin, the dark knight rises, Batman begins, etc. Which as we can tell can be deemed accurate.

VI. CONCLUSION

Our objective has been to develop a unique method for enhancing movie categorization, The fact that recommender systems require a lot of data to provide excellent recommendations is perhaps the largest difficulty they face. To provide reliable suggestions, large volumes of data are needed, which opens the door to further applications including the use of big data technologies and effective data processing procedures. Another issue is that data is always changing. Data or information is never static, and it fluctuates constantly as a result of changing user behaviors and preferences. This project could be used as a prerequisite for developing more robust content-based recommender systems. In this project, we have successfully implemented a movie recommendation system using TF-IDF vectorization and Cosine similarity. And we further plan to develop a hybrid movie recommendation system with better accuracy and efficiency.

REFERENCES
[1] Abhishek Singh, Abhishek Rawat, j Shanmukh Rao, Samyak Jain, Uppalpati Yogendra Reddy" A research paper on machine learning-based movie recommendation system", International Research Journal of Engineering and Technology (IRJET)
[2] N. Muthurasu, Nandhini Rengaraj, Kavitha Conjeevaram Mohan, "Movie Recommendation System using Term Frequency-Inverse Document Frequency and Cosine Similarity Method", International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-7 Issue-6S3 April 2019
[3] P. Sabari Ramkumar, J. Aswin, "A Hybrid Movie Recommender system based on Content and Collaborative Filtering methods", 2019 JETIR March 2019, Volume 6, Issue 3

[4] Yu Zhu, Jinhao Lin, Shibi He, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai, Member, IEEE"Addressing the Item Cold-start Problem by Attribute-driven Active Learning" JOURNAL OF LATEX CLASS FILES

[5] Ranjan Kumar, S A Edalatpanah, Sripati Jha, Ramayan Singh, "A Novel Approach to Solve Gaussian Valued Neutrosophic Shortest Path Problems", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8 Issue-3, February 2019

[6] Choi, Sang-Min, Sang-Ki Ko, and Yo-Sub Han. "A movie recommendation algorithm based on genre correlations." Expert Systems with Applications 39.9 (2012): 8079-8085.

[7] Lekakos, George, and Petros Caravelas. "A hybrid approach for movie recommendation." Multimedia tools and applications

[8] Zhang, Jiang, et al. "Personalized real-time movie recommendation system: Practical prototype and evaluation." Tsinghua Science and Technology

[9] Rajarajeswari, S., et al. "Movie Recommendation System." Emerging Research in Computing, Information, Communication, and Applications. Springer, Singapore, 2019.

[10] Ahmed, Muyeed, Mir Tahsin Imtiaz, and Raiyan Khan. "Movie recommendation system using clustering and pattern recognition network." 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2018.