MOVIE RECOMMENDATION SYSTEMS USING HYBRID FILTERING MODEL

A MINOR PROJECT REPORT

Submitted by

B.S.M GUPTA [RA2211028010203]

M.ABRAHAM MARTIN [RA2211028010224]

B. SASI KIRAN [RA2211028010227]

Under the Guidance of

Dr. P. GOUTHAMAN

Assistant Professor,

Department of Networking and Communications

in partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY in

COMPUTER SCIENCE ENGINEERING

with specialisation in CLOUD COMPUTING

DEPARTMENT OF NETWORKING AND COMMUNICATIONS

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

NOVEMBER 2024

Department of Networking and Communications

SRM Institute of Science and Technology

Own Work Declaration Form

Degree/ Course	**:**	Big Data Essentials (21CSC314P)
Student Name	**:**	B.S.M. Gupta, M. Abraham Martin, B. Sasi Kiran
Registration Number	**:**	RA2211028010203,RA2211028010224, RA2211028010227
Title of Work	**:**	Movie Recommendation Systems Using Hybrid Filtering Model

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate.
- Referenced and put in inverted commas all quoted text (from books, web, etc).
- Given the sources of all pictures, data etc. that are not my own.
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present.
- Acknowledged in appropriate places any help that I have received from others (e.g.fellow students, technicians, statisticians, external sources).
- Compiled with any other plagiarism criteria specified in the Course handbook/ University website.

We understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

We are aware of and understand the University's policy on Academic misconduct and plagiarism and we certify that this assessment is our own work, except where indicated by referring, and that we have followed the good academic practices noted above.

B.S.M. Gupta	M. Martin	B Sasi Kiran
SIGNATURE	**SIGNATURE**	**SIGNATURE**
B.S.M Gupta	**M.Abraham Martin**	**B.Sasi Kiran**
RA2211028010203	**RA2211028010224**	**RA2211028010227**

iii

SRM

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **21CSC314P – Big Data Essentials** entitled in "**MOVIE RECOMMENDATION SYSTEMS USING HYBRID FILTERING MODEL**" is the bonafide work of "**B. S. M GUPTA [RA2211028010203], M. ABRAHAM MARTIN [RA2211028010224]**and **B. SASI KIRAN [RA2211028010227]**" who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

P. Gouthaman (COMPLIANT ISSUES)
6/11/2024

SIGNATURE
PANEL REVIEWER I
Dr. P. GOUTHAMAN
Assistant Professor,
Networking and
Communications

D. Saveetha 6-11-24

SIGNATURE
PANEL REVIEWER II
Dr. D. Saveetha
Assistant Professor
Networking
and Communications

iv

ACKNOWLEDGEMENT

We express our humble gratitude to Dr. C. Muthamizhchelvan, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, Dr.T.V. Gopal, and Dr. Revathi Venkataraman, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for their valuable support during the project course.

We are incredibly grateful to Head of the Department, Dr. M. Lakshmi, Professor and Head, Department of Networking and Communications, School of Computing, SRM Institute of Science and Technology, for offering us the project course: Big Data Essentials.

We want to convey our thanks to our faculty mentor Dr. Gouthaman P., Assistant Professor, Department of Networking and Communications, School of Computing, SRM Institute of Science and Technology, for his inputs and support during the project phase.

We express our respect and thanks to Department HOD, SRM Institute of Science and Technology, for providing us an opportunity to pursue this project course.

We register our immeasurable thanks to our Faculty Advisor, School of Computing, SRM Institute of Science and Technology, for leading and helping us to complete our course.

We sincerely thank the Networking and Communications department staff members, SRM Institute of Science and Technology, for their help during our project.

Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support, and encouragement.

B.S.M GUPTA [RA2211028010203]

M.ABRAHAM MARTIN [RA2211028010224]

B.SASI KIRAN[RA2211028010227]

ABSTRACT

AI-assisted movie recommendation algorithms are currently one of the valuable tools for the personalization of user interaction in the world of digitized communication and granting viewers individualized access to movies that most probably match the preference of viewers. Most algorithms generally function on user rating, viewing history, or any other type of behavioral activity. There are two types of prevalent approaches in these systems, such as content-based filtering and collaborative filtering. In contrast to content-based methods, that analyze the attributes of the items themselves, such as genre, director, or themes, and recommend the movies that have similar characteristics to those liked by a user in the past; Collaborative filtering examines the likes and dislikes of other people who share similar tastes in returning recommendations based on the likes of this "similar" user group. This algorithm works in such a manner that if a particular actor features in action films of regular use, the pattern gets recognized and other relevant action movies with the actor become suggested. The key advantage associated with these recommendation algorithms is their ability to work well on big data while bringing efficient recommendations by deriving benefits from patterns in comprehensive sets of data. However, one well-known limitation of collaborative filtering is the "cold start" problem, which occurs when there is insufficient information.This lack of information makes it difficult to produce correct recommendations because the system cannot find any strong preferences or similarities. As an attempt to overcome this limitation, hybrid models have emerged that combine content-based filtering with collaborative filtering. These models leverage the best of both worlds to produce more accurate and dynamic recommendations. The hybrid models look at different data points such as history and ratings of the user along with movie characteristics and so on, hence, enable the generation of recommendations to be personalized even in conditions where data is scarce in the beginning. For example, matrix factorization reduces the complexity of user-item matrices, thereby identifying latent factors that represent hidden user preferences. It groups users with similar preferences while allowing neural networks to recognize deeper patterns in user behavior. The result is that these recommendations are more dynamic than ever. Popular streaming services such as Netflix, Amazon Prime, and Hulu have implemented such sophisticated AI-driven recommendation systems to increase user

engagement and retain viewer loyalty.Customized movie recommendation is a factor that allows these platforms to enhance their overall user experience since it helps the user in finding movies he likes and encouraging him to find more movies.

LIST OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A movie recommendation system can be visualized as some kind of advanced tool that predicts and filters users' preferences for movies by some means or other in accordance with their previous interactions and activities. There are various methods used, and all such methods can be classified into three categories: content-based filtering, hybrid approaches, and CF/CFB. Collaborative filtering recommends films using the aggregate ratings and tastes of the different users. For example, if two other users have assigned similar ratings to a set of movies, then their system might suggest films that one enjoyed for the other so that the whole process of finding such content with great taste is enhanced. To the side of content-based, it takes into account the characteristics of specific movies such a method recommends films that are like those the viewer has liked before. This method takes into account the actors, directors, and genre as it recommends new movies along the lines of the user's preferences as it is. For instance, it will suggest more thrillers with the same or related topics and maybe featuring an actor that the user likes thrillers with.Hybrids both content-based and collaborative approaches[1], thus minimizing the cons associated with each of the approaches. Movie recommendation algorithms do magic on known streaming services like Netflix[2] and Amazon Prime Video where the users engage more and are much satisfied with the tailor-made recommendations. But even now, it poses issues like the "cold start" problem - triggered either from new users or from new products that haven't built enough information yet to recommend anything good. Innovations are continuous with each ongoing phase, perfecting sophisticated algorithms and advanced methods in machine learning continuously fix these problems. The recommendation system is finally responsive, with changes in patterns over time due to varied demand. In a nutshell, AI movie recommendation systems are the perfect examples of shifting user experience through the selection of personalized material in the infinitely large entertainment world. Each method has its strengths and weaknesses and is often combined for optimal performance. Content-based filtering relies on the attributes of the content itself. For example, if a consumer prefers the action movies

with his favourite actor, a content-based system would suggest other action movies or movies that he is likely to enjoy from the cast of the said movies. So, essentially, this algorithm just tests movies that share similar properties with those a user typically prefers. Collaborative filtering, on the other hand, exploits the rankings and preferences of other viewers whose preferences are similar. For instance, if two users had rated a series of movies similarly, collaborative filtering would recommend movies that one liked to the other.Hybrid recommendation methods combine the best features of content-based and collaborative filtering methods to help bridge each method's disadvantages. For example, a broad-range-preference user will face problems with the content-based method, and in collaborative filtering, it is the "cold start" problem in that either the new users or new items do not have adequate data to produce reliable recommendations. Recommendation systems have improved significantly further with the use of artificial intelligence, so they have become extremely sensitive to a change in user behavior or tastes.However, due to these advancements, such problems like cold start have still existed in the network especially when new content and users are coming into a system with a lack of information. These gaps continue to be filled for these streaming services through the betterment of algorithms that allow deep learning and knowledge graph that can enhance the understanding over content and user preferences of the service.It uses the study of previous user interactions, including ratings, watch history, and clicks, to predict what a user might enjoy next. This saves time and increases engagement. It is the main reason behind the success of streaming platforms like Netflix, Amazon Prime Video, and Hulu, where users are kept engaged and coming back for more through tailored recommendations.For example, if a user likes movies of the sci-fi genre directed by Christopher Nolan, then a content-based system may recommend other sci-fi movies or the works of Nolan. It is powerful where users have strong preferences for certain genres actors. Collaborative filtering differs because it looks at the preferences and behavior of similar consumers. Instead of looking at content, it uses the choices made by like-minded users and shows movies that other people who are watching just like them have enjoyed. For instance, if User A and User B have many action movies rated highly, then the system might suggest to User B other action movies liked by User A. Collaborative filtering helps the system in finding hidden gems through collective preferences that most often introduce users to content they might not otherwise come across. Such approaches face immense difficulties in "cold start" scenarios with inadequate information for new users

or movies, which causes problems to the generation of meaningful recommendations. To face such a problem, many applications adopted hybrid approaches, combining elements of both the content-based and collaborative filtering approaches.Hybridization also minimizes the "cold start" problem because content attributes are used to make initial suggestions, and then the recommendations are refined as more data becomes available. Since the emergence of AI, recommendation systems have become much more efficient. More advanced AI approaches, such as machine learning, enable systems to look for intricate patterns in user behavior such as watch duration, browsing habits, and even time of day preference.Advanced tools ensure that streaming platforms are improving their capacity for delivering more personalized, engaging recommendations with time. With recommendation systems, movie is among the pioneers in reshaping the digital entertainment landscape; they make it efficient and enjoyable for users to search for the right content. Bringing together both content-based and collaborative approaches with state-of-the-art AI, such systems ensure a very fluid experience for the users amidst an ever-growing streaming entertainment world. Indeed, systems evolve with the change of both technology and user experience side by side, showing just how data-driven personalization may be able to help bridge satisfaction in an endless sea of choices.

We address some key questions here:

1) What is a movie recommendation system, and how does it predict and filter user preferences?

2) What are the main categories of recommendation methods, and how do they differ?

3) How does collaborative filtering (CF) work, and how does it enhance the discovery of content?

4) In what ways does content-based filtering differ from collaborative filtering?

CHAPTER 2

LITERATURE REVIEW

Hybrid filtering brings the good features of both collaborative and content-based approaches into one so that it can provide suggestions even better than either could deliver on movie recommendation systems. Using both user-similarity features and content characteristics, platforms are enabled to personalize suggestion that can break the limits of using just one. Jasmine (2024) et al. [1] Hybrid systems are particularly more effective for new users or those with sparse interaction histories, as they can attract multiple data sources to provide much more relevant recommendations. As the hybrid model combines collaborative data with content attributes like genre, actors, and directors, it comes closer and is much individualized to recommendations, providing a great deal of enhancing user experience.

One of the most striking examples of hybrid filtering to promote engagement in a massive streaming platform is regarding Netflix. Bhavani and Sai (2024) et al[2] brought into focus the model of Netflix, which amalgamated collaborative filtering with big data analytics to illustrate real-time, personalized recommendations to keep users engaged. Recommendations at high levels of user behavior datasets make analysis line in conformity with a user's preferences but suggest something that does not only appear in proper time but proper according to his need. Therefore it doesn't help only with making the users enriched but does reflect on the effect created by hybrid models on a retention level as well even on their frequent engagement level.

In MOVREC, content-based filtering and collaborative filtering capture user preferences by making sure that its accuracy is high. Kumar et al. (2015) discuss how MOVREC[3] employs the ratings given by the users to recommend movies similar to those the user liked the most previously, hence meeting the taste of individuals. This hybrid approach helps the system to make recommendations that strike a chord better with the user.

Success of MOVREC is proved and it shows the potential benefit that hybrid methods have when creating recommendation models that accurately portray users' preferences, meaning close matches to the interest of the viewer as previously stated.

It has also been discovered that clustering methods can actually be very effective when seeking improved recommendation accuracy with regard to collecting more relevant items. Vilakone et al [4] designed an advanced k-clique-based clustering algorithm. It reduced neighborhood bias in collaborative filtering and improved recommendation accuracy as well. This approach is to deliver more balanced results by forming clusters based on similarity, not by the individual attributes of the user. As k-clique is based on group similarities, common biases reduce, and recommendations reflect diversified preferences better. This therefore demonstrates the capability of clustering to improve the accuracy of collaborative filtering.

Hybrid models that combine collaborative with content-based filtering are well-suited to counter the problem of sparsity as well as the cold-start within a recommendation system. Virk and Singh (2015) et al [5] have a hybrid model recommendation that combines two techniques, ensuring that there is quality consistency in the recommendations. Regarding this, the model can take advantage of overcoming limitations referring to one source of data, valuable to users who have little historical data. This hybrid approach ultimately brings quality recommendations while providing a well-balanced solution for new and returning users.

The recommendation systems present various limitations from scalability, preserving user privacy, accuracy, and others. According to a survey carried out by Sharma and Mann (2013) et al[6], Hybrid systems well suit the requirements and solve these problems since they have a mix of filtering techniques, adapting them to large-scale demands. They further comment that by merits from both user-based and item-based data the hybrid approach can enable a more holistic recommendation. That flexibility may well have the potential to find its way past some of the technical and ethical constraints thrown upon

single-method systems for increasing the scalability and robustness of the recommendation frameworks.

Big Data: This era transforms the way of developing the recommendation systems in order to be capable of processing vast volumes of data in support of very complex real-time suggestions. According to ProQuest in 2022 [7], the big data methodologies help the systems process large volumes of information. This enables big data methods to fit the high-demand platforms perfectly. The report emphasizes the fact that processing large datasets in recommendation systems makes them perform efficiently and give more precise recommendations even when traffic is so high. Hence, big data plays an important role in helping the modern recommendation system achieve the desired scaling without compromising on its accuracy.

It was thus realized that relationships among features of the content were included as an integral approach implemented within hybrid recommendation systems since they enrich the quality of personal recommendations. As already shown in the evidence reported by Aslanian et al. [8], the 2016-based evidence, the relationships in the content attributes such as genre, actors, or directors add to the richness and value of recommendations when included with collaborative filtering. In case of using the combination of user preferences and content features, systems make more precise predictions that show what a user has watched earlier. This reflects the benefit of content analysis in terms of creating highly personalized and relevant recommendations for the individual.

Big data methods are necessary for real-time large-scale recommendation systems. Awan et al. [9] discussed how advanced computational techniques enable handling massive user and movie datasets that make real-time suggestions feasible on a large scale. Their approach addresses typical issues like data scalability and responsiveness, which allows the recommendation systems to work in high-demand environments. Big data can use recommendations for the needs of the users with better recommendation algorithms that ensure the timeliness, relevance, and individualization of the suggestions according to their preferences.

Social context has been added to the neighborhood of recommendation algorithms in reducing neighborhood bias, which further makes the recommendations personal and relevant. Sánchez-Moreno et al. [10] demonstrate that the addition of social information to collaborative filtering algorithms enhances personalization by including the social network and the connections established by a user. Although their work is mostly based on music recommendations, the lessons taken here are of great interest in movie recommendation systems in which such social context could be applied to make the recommendations even more relevant in the social world and the interests of a user.

Hybrid algorithm optimization is an area that is coming up to enhance recommendation systems. Liu and Ren [11] discussed how hybrid movie recommendation systems approach with better accuracy as obtained with the integration of collaborative filtering with content-based approaches. Liu and Ren increased their prediction accuracy when refined algorithms applied. It is important that a high prediction accuracy could provide continuous accurate recommendations given by a platform. Their work revolves around growing hybrid models that integrate several techniques to achieve a far more fine-grained, yet trustworthy, recommendation experience.

Hybrid systems in the field of recommendation have turned out to be successful-especially regarding the cold-start problem: little is known about new users or items. Khalaji et al. [12] proposed a Resource Allocation Hybrid Recommender, which ranks the recommendations to new users by combining both user-specific and content-based similarities even when very limited information is available about them. It thus indicates the merits of hybrid techniques in that it can propose suitable recommendations even in initial stages of interaction with a user, thereby enhancing chances of efficiently engaging new users for a system.

It seems that deep learning forms an instrumental part in combating cold-start problems when a recommendation system integrates it. Wei et al. [13] demonstrates that deep

learning-based collaborative filtering systems learn preference more quickly and make recommendations to new users more manageable with less initial data. This approach is adaptive, which provides an adaptive recommendation model that responds to the interests of its users even in the first interaction, and it shows how machine learning can lead to better flexibility and precision for the recommendation systems.

The fusion of collaborative and content-based filtering has been quite a helpful approach to achieve a high accuracy within the hybrid recommendation systems. Bhalla et al. [14] explained how their model combines user preferences with content attributes to provide suggestions that are relevant and personalized. The integration enables greater precision, as the system can satisfy both the user's preferences and the specific features of the movie. The success of this hybrid model shows the need for using multiple data sources to meet diverse user needs effectively.

Clustering algorithms will endow recommendation processing with reduced computational complexity, which is essential for handling large data sets. Kuzelewska (2014) [15] suggests how clustering can be used in hybrid recommendation systems to improve efficiency without sacrificing accuracy. This study is actually evident in the use of the MovieLens dataset; such clustering would help scale up recommendation systems while providing practical solutions to platforms looking for a way to serve precise recommendations to large numbers of users without an overwhelming load on computing.

CHAPTER 3
PROPOSED MODEL

The movie recommendation system has undergone many diversified developments. These systems have enjoyed the benefits of different techniques and technologies, such as those that enhance personalization and efficiency. In these systems, the central role is played by collaborative filtering and content-based filtering models [1]. Collaborative filtering uses the interactions of the user to find patterns based on past behavior and similarity among users and items. On the other hand, content-based filtering is more focused on film recommendations through feature analysis of films a user liked to watch in the past. However, these two methodologies often do not scale for handling huge data sizes that require handling; frameworks such as MapReduce come in handy in such situations where parallel processing can be facilitated across many nodes, handling large data sets with effectiveness [2].In addition to these approaches, NoSQL databases like MongoDB have gained popularity because they tend to be better at large stores of unstructured film data, and the resultant flexibility in data management makes them more efficient in performing queries compared to relational ones [3]. In this regard, data processing tools like Apache Pig and Hive provide high-level abstractions that make data manipulation or querying in Hadoop even easier. These tools offer an easy way to filter out the noise from preprocessing large datasets, which actually simplifies the workflow until more complex analyses can begin [4].Besides the traditional filtering method, a hybrid recommendation technique has emerged and is actually applied by combining the goodness of the collaborative and the content-based approach [5]. In hybrid systems, it could either integrate their outputs, or dynamically alternate between different methodologies based on certain criteria to optimize its recommendations. Model-based hybrids apply high techniques such as matrix factorization and deep learning in order to bring a proper way of showing complex relations between users and items so that the qualities of recommended items are obtained. Context-aware recommendation systems take it further, as they involve contextual factors such as time, location, and the mood of the user.To improve content-based recommendations, TF-IDF and cosine similarity are extensively applied during data analysis technique for the calculation of movie similarity based on textual description. Techniques for collaborative filtering also include a matrix

factorization, which requires breathtakingly low memory needed in keeping up with the user-item interaction matrices; classical methods include SVD and ALS [6].The performance of such recommendation systems is evaluated using various measures of performance. Accuracy measures such as Mean Absolute Error and Root Mean Squared Error measure the accuracy of a prediction. At the same time, ranking measures-precision, recall, F1 score, Normalized Discounted Cumulative Gain, and Area Under Curve-measure the relevance of recommendations that are presented to users [7]The additional information brought into the system allows the making of recommendations in a more personalized manner based on his current situation or preferences. Contextual information can be combined with either factorization models or rule-based systems that use contextual features at the time of making a recommendation.

Movie recommendation systems have emerged as an integral feature of online sites that help enhance user interactions by bringing relevant content. However, such systems have several inbuilt issues that hinder them from becoming highly effective as well as user-friendly. The biggest issue is the *cold start problem*, which emerges when there are new users or items without enough information to be recommended. For example, a new user may not have any interaction history, or a newly released movie may not have received enough ratings. In the collaborative filtering systems, which are based on user-item interactions, this lack of data makes it difficult to provide personalized suggestions. Hybrid recommendation models can be used to solve this problem by combining collaborative filtering with content-based methods. Through metadata, for example movie genre, director, or actors, it is able to generate initial recommendations that can be of help especially to new users or new items. Such would bring a gap together in bridging so as to afford meaningful suggestions on new items or users at the same time further enhance the recommending over time due to accumulated data.Data sparseness is also another big challenge that a recommendation system faces. Collaborative filtering approaches rely considerably on the existence of user-item interaction data for the identification of patterns as well as to make correct predictions. Such a kind of data is often very sparse, particularly when considering niche movies, low-popularity genres, and new items with extremely fewer ratings. For instance, it could be the case that a person has rated a few movies; likewise, a particular kind of genre may not even have a sizeable population. This type of sparse data therefore causes incomplete matrices that tend to result in bad prediction accuracy. Such approaches might include the use of

Singular Value Decomposition (SVD) in matrix factorization techniques, which would be able to decompose a user-item matrix into latent factors; these latent factors would provide an approximation of the missing data. Hybrid models of collaborative filtering combined with a content-based approach will ensure that even if interaction between users is minimal, an item's attributes-say, director or genre-and its corresponding movies could still help in movie recommendations. This hybrid model combination enhances the capacity of the system in recommendation making even with sparse interaction data.The second added layer of complexity comes from hybrid models. Hybrid model combinations can outperform one technique by much, but at a price. Since there is a corresponding increase in complexity by employing more techniques, deep learning and complicated matrix factorization hybrids in particular are high consumers of processing time and power depending on how large the interaction data becomes. This increases the computational requirement, leading to slower response times, increased infrastructure costs, and even delay in real-time recommendations. Some solutions to these are available. For example, models can be simplified either by reducing the number of input variables or using methods such as PCA for dimensionality reduction. Another alternative is distributed computing resources or cloud-based services to scale up the system and deal with larger datasets. These technologies have advantages of faster processing and rapid responses. For real-time systems, approximate methods may include nearest-neighbor search algorithms in the case of real-time use. These methods show sufficient performance while balancing precision with speed, so the recommendations are delivered in a timely way.Contextual variability is another challenge introduced whenever contextual data, including such as time of day, location, device type, or even mood, become involved. Therefore, context-aware recommendation systems are designed with relevance in mind based on the context of the user. However, this increases the complexity of the data being managed. The data tends to be inconsistent due to its varied format recording different types of devices and platforms or even poor capturing of it. This could bring out less accurate or irrelevant recommendations because the system could interpret the context wrongly. This can be overcome by developing a unified context model that standardizes the way contextual data is collected and interpreted across devices and platforms. In doing so, the system can better rely on the consistency in the representation of context. Moreover, choosing only the most relevant contextual factors while avoiding noisy or redundant data can improve the overall performance of the

system. For example, if time of day is the best predictor for movie preferences, it will be used in preference; less impactful variables such as the user's device type may be de-emphasized.

The more context-aware features that are included in the system, the more important scalability becomes. Multiple contextual factors-including location, time, device, etc.-will multiply the number of model dimensions, potentially exponentially increasing model complexity. This makes it all the more difficult to establish consistent system performance and response time, particularly in applications that require real-time outputs. Scalability issues can be overcome by sparse matrices or dimensionality embedding techniques; these techniques compress the input data while maintaining important contextual elements. Other model compression methodologies include quantization or pruning, which reduce the model size so that the derived model is efficient and scalable. Another, more sophisticated indexing scheme might have to be added at this stage: for example KD-trees or Locality Sensitive Hashing LSH that do not involve any performance penalty when going towards larger data size.Overfitting is the very common phenomenon occurring in numerous recommendation systems; this affects hybrid models primarily as there are various input variables combined with several factors. Model overfitting happens at the moment a model begins to specialize so much toward the training data, ultimately losing generalization capability. This shows that the model is doing a good job on the training data but fails to make correct predictions for unseen data. The performance of the model, therefore, is not satisfactory in real-world scenarios. Several strategies can be adopted to reduce overfitting. Regularization techniques, including L2 regularization, penalize overly complex models and hence reduce the risk of overfitting. The next valuable technique is cross-validation, where the available data are divided into different subsets, and the model is trained using various combinations of the subsets. This will provide an estimation of how good the model will be on unseen data and thus facilitates better generalization. Other interesting methods include ensemble methods, where the results from various models are used to obtain a single answer. These reduce both bias and variance. Therefore, such ensemble methods bring about stronger and more generalizable models.

Finally, *user privacy* is a significant concern in context-aware recommendation systems. These systems largely rely on large data aggregation of personal preferences,

behavior patterns, and context. However, it leads to serious privacy concerns as a result because individuals might not be willing to share such information because of surveillance or the possibility of its misuse. In order to alleviate some of these privacy issues, there are a few differential privacy-preserving techniques available. Differential privacy is sure to prevent re-identification of individual user data, even when analyzed or modeled in the presence of any form of analysis or modeling. Furthermore, data can be anonymized and aggregated before being processed for removal of any personally identifiable information. Another promising direction is federated learning in which the model updates from the user device are communicated to the central server while retaining the data locally on the user's device, ensuring personalization can occur without compromising user privacy. Transparency will also be essential; thus, the transparency to the users through consent forms and ensuring that the users are in control of their preferences on sharing will be core to building trust and complying with privacy regulations. Movies in the application can be getting recommended using a good properly designed solution towards few of those stated challenges by, thus it offers great scalability and improvement in terms of user performance. That makes it a more applicable method, with great accuracy that mainly gives better ethics and, thus results in an added value in regards to such systems and this model contributes with the right approach between them, privacy preserving.

ARCHITECTURE DIAGRAM:

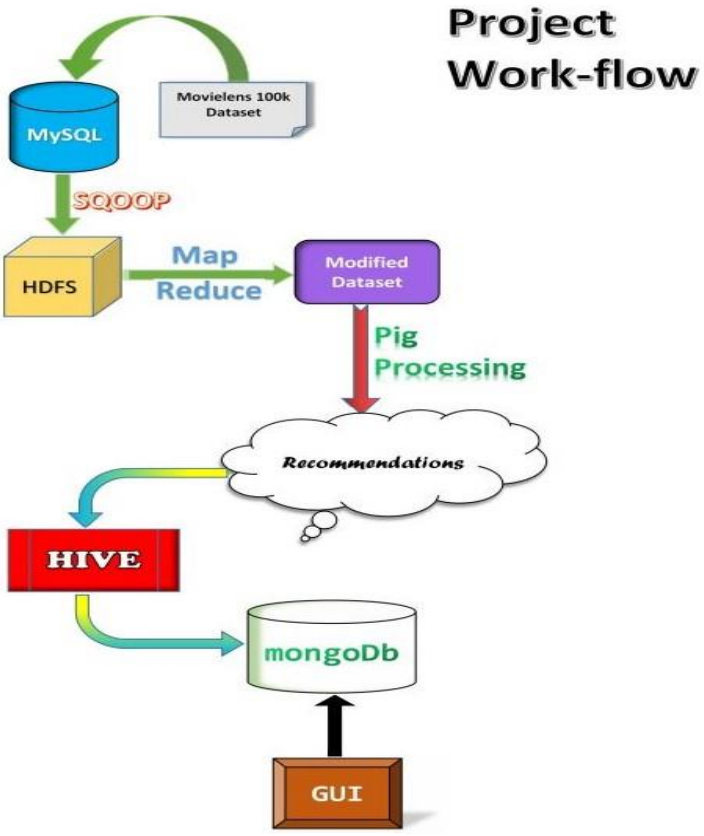


Figure 3.1: Architecture Diagram

The Movie Lens 100k dataset is stored in a MySQL database at the beginning of the movie recommendation system workflow, as illustrated in Figure 3.1 The data is imported into HDFS by SQOOP, where MapReduce and Pig Processing are used to enhance it. After analysis in Hive, the refined data is moved to MongoDB for scalability and management of unstructured data. Lastly, a GUI interface is used to display the recommendations, and a NoSQL database and distributed processing are used to optimize speed. In movie recommendation system architecture, one would integrate data processing, storage, and visualization tools, build a strong recommendation pipeline, and analyze performance. With the *MovieLens 100k dataset* stored in a MySQL database, you go through several steps to create personalized movie recommendations. This would involve moving data from MySQL to HDFS, a distributed file system that is meant for processing large files on multiple nodes. Using the part of Hadoop architecture MapReduce, this data is further processed and organized into a structured format, then transformed using *Apache Pig* with complex filtering and grouping of data to create

recommendations.The recommendations are then output by saving them in Apache Hive: this is a form of a data warehouse; it allows SQL-like queries to make it easy and fast to retrieve and analyze large datasets. These entries are forwarded to MongoDB, a NoSQL database, which stores such recommendation data efficiently in some flexible document-based format; these will be retrieved efficiently. The Graphical User Interface (GUI) interacts with MongoDB in order to graphically display these recommendations before the users so that he will be able to explore these lists, filter them, interact with these personalized movie suggestion suggestions.The role of visualization tools is also fundamental in understanding patterns in the data, measuring performance and drawing insights- Pandas, Matplotlib, Seaborn, and Scikit-learn are utilized in this endeavor. Using Pandas, which is actually a library meant for managing data, we can therefore efficiently filter out, aggregate, and change data from a DataFrame towards exploratory analysis. More importantly, it is easy to import this library and make simple plots right from a DataFrame along with other visualization tools.This can plot confusion matrices, ROC curves, and precision-recall curves, if needed for the evaluation of classification-based recommendation schemes. Another area that visualizations Scikit-learn can offer is in drawing insight from groupings of users or movies based on some behavioral similarity using clustering or classification.In general, architecture contains strong data processing in terms of HDFS and MapReduce, Pig in data processing, efficient storage, which includes Hive as well as MongoDB, and last but not the least visualizations of Pandas and Matplotlib, Seaborn, as well as Scikit-learn to make high-quality recommendations via an easy-to-use GUI. The visualizations are much better for the interpretations of data, which provide scope to track the performances of recommendation systems in digging trends, anomalies, or actionable insights that help and make room for continuous improvements. Hadoop is used as the backbone for scaling, MongoDB provides flexible storage capacity, and advanced visualization libraries all give it a quite powerful feel and engaging user experience of recommendations.

CHAPTER 4
IMPLEMENTATION

The movie recommendation system of the MovieLens 100K dataset, the actual implementation process consisted of some major steps: setting an environment in Google Colab to use the necessary libraries, namely Pandas, NumPy, Seaborn, Matplotlib, and Scikit-Learn.

Environment Setup

In the construction of our movie recommendation system, we made use of Python and Google Colab with an interactive coding environment containing many of the required libraries and tools. We relied on the following important libraries:

Pandas: This is used in data manipulation and preprocessing

NumPy: Used for all numerical computations

Seaborn and Matplotlib: Used in the visualization

Scikit-Learn: The cosine similarity is calculated with the vectorization.

Load Movie Lens 100K Dataset

Movie Lens 100K dataset consists of two important files:

Ratings data (u.data): This includes the ratings, which different users have given to various movies.

Movies data (u.item): Metadata describing movies. The datasets included: the movies and the ratings.

All datasets were imported into Pandas DataFrames. Ratings Data included: user_id, movie_id, rating and timestamp while movies data comprised information including movie titles, release date, and genres under all categories.

Data Preprocessing and Merging

Merging the data of all the movies with the rating data, content-based filtering was performed in a unified Data Frame. Thus, allowing for the association of a movie's metadata with rating from users helped in reducing complexity and enabling easier

examination of recommendations and generation based on user rating.Genre Extraction: Each movie has multiple genre tags (for example, Action, Comedy, Drama). It was created into one "genres" column as now it is one string of text that is showing all the genres of that respective movie which will now assist in converting that column of text for similar text computation.

Content-based filtering using Text Vectorization

For this recommendation system, we implemented content-based filtering by comparing genres of movies. We used TF-IDF vectorization and represented each movie's genres numerically to compare the similarity of genres across different movies.

Cosine Similarity Computation: After we turned the movies into vectors, we performed the computation of cosine similarity. We measured the angle between two vectors and a measure of similarity between two movies using genre vectors. It has actually been used as a method of finding out which movie is closest to a specific movie about which we are expressing our opinion, on the basis of kinds of genres that are common.

Recommendation and Rating Prediction Functions

There have been two major functions which have been developed.

1.Recommendation function: This will pick the top 10 most similar movies to a movie input by computing similarity scores and sorting them in descending order. It returns a list of movies with genres similar to the input movie.

2.Rating Prediction Function: This will average the ratings of similar movies to estimate what the user might give to this movie. This provides for a predicted rating based upon the user ratings of a similar movie, enabling some simple form of rating prediction.

Results-Display in User-Friendly Format

The output was customized to make it presentable and informative:

- Predicted Rating: This is a displayed predicted average rating on similar movies.
- Similar Movie Recommendations: A list of 10 recommended movies with details of genre similar to the input movie.

- Visual Representation: A bar chart has been created to present ratings of the recommended similar movies, enhancing the visual impact and aiding in quick results interpretation.

In HTML format, the results were presented in Google Colab and colored for better clarity and engagement.

User Interaction

This version allowed for the final implementation because it added the prompt with the movie title in an input section. After entry, the function of recommendation ran and produced personalized suggestions along with rating prediction about that movie. If not located in the dataset, a prompt about an error came up which ensured sound functionality in user's overall experience.

System Testing and Validation

Each part of the movie recommendation system was individually tested for correct working, precision, and user experience. The tests included accuracy tests whereby the recommendation algorithms were put to the test regarding precision and recall in the generation of recommendations. This was by comparing the generated recommendations with a set of known relevant movies.

Latency Measurement: This was considered in the process to measure the response time of the recommendation system, ensuring that the user input is processed very fast, thus giving real-time recommendations. This was achieved through checking the time taken from when the user input is made to when it generates results and presents them in the user interface.

Scalability Testing: Check whether the system performs with higher sizes of the dataset and user requests by executing tests with the different sizes of the MovieLens dataset and by simulating the system running with many users concurrently so that it still performs reasonably well when a load change happens.

Based on these experiments, the system was proved stable, generating movies relevant enough to make predictions with sufficient accuracy while showing low latencies; and its interfaces were kept simple for the users so that usability increased.

In summary-The recommendation system implementation started with environment setup from Google Colab and utilized the Pandas and Scikit-Learn libraries in its development. The merged data for both ratings and movies was prepared and normalized to a common format before application. Further, using TF-IDF, it was possible to convert the genres into their vector representations while computing cosine similarity, allowing the return of related films. Lastly, it created two functions that provided a list of the top 10 matching films while predicting the user rating on them. The output was visualized, and an input prompt was included to encourage users. It was successful in demonstrating the content-based recommendations by allowing users to make relevant movie suggestions based on their own preferences.

Step-by-Step Guide for Movie Recommendation System

1. Setting Up the Environment

a. Install Necessary Libraries

Ensure that your Google Colab environment has all required libraries.

```python
# Install necessary libraries
!pip install pandas numpy matplotlib seaborn scikit-learn
```

Figure 4.1: Installation

b. Import Libraries

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
from IPython.display import display, HTML  # For HTML display in Google Colab
```

Figure 4. 2: Importing libraries

2. Load the MovieLens 100K Dataset

The MovieLens dataset includes two main files:

•u.data: Contains user ratings for movies.

•u.item: Contains metadata about movies (title, release date, genres).

```python
# Load the ratings and movies data
ratings_path = '/content/dataset/ml-100k/u.data'
movies_path = '/content/dataset/ml-100k/u.item'

# Define column names for the datasets
ratings_columns = ['user_id', 'item_id', 'rating', 'timestamp']
movies_columns = ['item_id', 'title', 'release_date', 'video_release_date', 'IMDb_URL', 'unknown', 'Action', 'Adventure',
                  'Animation', 'Children\'s', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror',
                  'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western']

# Load datasets
ratings_df = pd.read_csv(ratings_path, sep='\t', names=ratings_columns)
movies_df = pd.read_csv(movies_path, sep='|', names=movies_columns, encoding='latin-1')
```

Figure 4.3: Sample of Raw Ratings Data

3. Data Preprocessing and Merging

a. Merge Ratings and Movie Metadata

Combine the ratings and movies data to create a single DataFrame.

b. Extract Movie Genres

Create a single genres column for each movie by concatenating the genre labels.

```
# Merge ratings and movies on 'item_id'
merged_data = pd.merge(ratings_df, movies_df[['item_id', 'title', 'Action', 'Adventure', 'Drama', 'Comedy', 'Sci-Fi']], on='item_id')
# Create a 'genres' column by concatenating genre labels for each movie
movies_df['genres'] = movies_df.iloc[:, 5:].apply(lambda x: ' '.join(x.index[x == 1].tolist()), axis=1)

# Merge ratings with the 'genres' information
data_with_genres = pd.merge(ratings_df, movies_df[['item_id', 'title', 'genres']], on='item_id')
```

Figure 4.4: Genre Encoding Process

4. Text Vectorization for Content-Based Filtering

Use TF-IDF to vectorize genres for each movie, enabling cosine similarity calculations.

```python
# Vectorize the genres text data
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(movies_df['genres'])

# Calculate cosine similarity based on genres
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

Figure 4.5: TF-IDF Vectorization of Movie Genres

5. Recommendation and Rating Prediction Functions

a. Movie Recommendation Function

Recommend movies based on cosine similarity of genres.

```
def recommend_movies(movie_title, cosine_sim=cosine_sim):
    idx = movies_df[movies_df['title'].str.contains(movie_title, case=False)].index[0]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    movie_indices = [i[0] for i in sim_scores[1:11]]
    return movies_df.iloc[movie_indices][['title', 'genres']]
```

Figure 4.6: Cosine Similarity Matrix

b. Rating Prediction Function

Predict ratings by averaging the ratings of similar movies.

```
def predict_rating(movie_title):
    idx = movies_df[movies_df['title'].str.contains(movie_title, case=False)].index[0]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    movie_indices = [i[0] for i in sim_scores[1:11]]
    similar_movies = merged_data[merged_data['item_id'].isin(movies_df.iloc[movie_indices]
    avg_rating = similar_movies['rating'].mean()
    return avg_rating
```

Figure 4.7: Similar Movie Recommendations

c. Function to Display Recommendations and Predicted Rating

Format and display the recommendations and predicted rating in HTML format for Google Colab.

```python
def rate_movie(movie_title):
    try:
        predicted_rating = predict_rating(movie_title)
        recommendations = recommend_movies(movie_title)

        recommendations_html = f"<h3 style='color:cyan;'>Predicted rating for '{movie_title}': <span style='color:yellow;'>{predicted_rating:.2f}</span></h3>"
        recommendations_html += "<h4 style='color:magenta;'>You may also like these similar movies:</h4><ul>"

        for i, row in recommendations.iterrows():
            recommendations_html += f"<li>🎬 <strong>{row['title']}</strong> | <span style='color:lightblue;'>Genres:</span> <em>{row['genres']}</em></li>"

        recommendations_html += "</ul>"
        display(HTML(recommendations_html))

        similar_movie_titles = recommendations['title']
        similar_movie_ratings = merged_data[merged_data['title'].isin(similar_movie_titles)]['rating']

        plt.figure(figsize=(10, 6))
        sns.barplot(x=similar_movie_ratings.values, y=similar_movie_titles, palette='coolwarm')
        plt.title('Ratings of Similar Movies')
        plt.xlabel('Rating')
        plt.ylabel('Movie')
        plt.show()

        return predicted_rating, recommendations

    except IndexError:
        error_html = f"<p style='color:red;'>Movie not found. Please check the spelling or try another movie.</p>"
        display(HTML(error_html))
        return None
```

Figure 4.8: Predicted Rating for Input Movie

6. Execute the Recommendation

Run the function with an example movie title input. Enter a movie name in the prompt to see the output.

```python
movie_input = input("Enter a movie name: ")
rate_movie(movie_input)
```

Figure 4.9: Execute Recommendations

CHAPTER 5

RESULTS

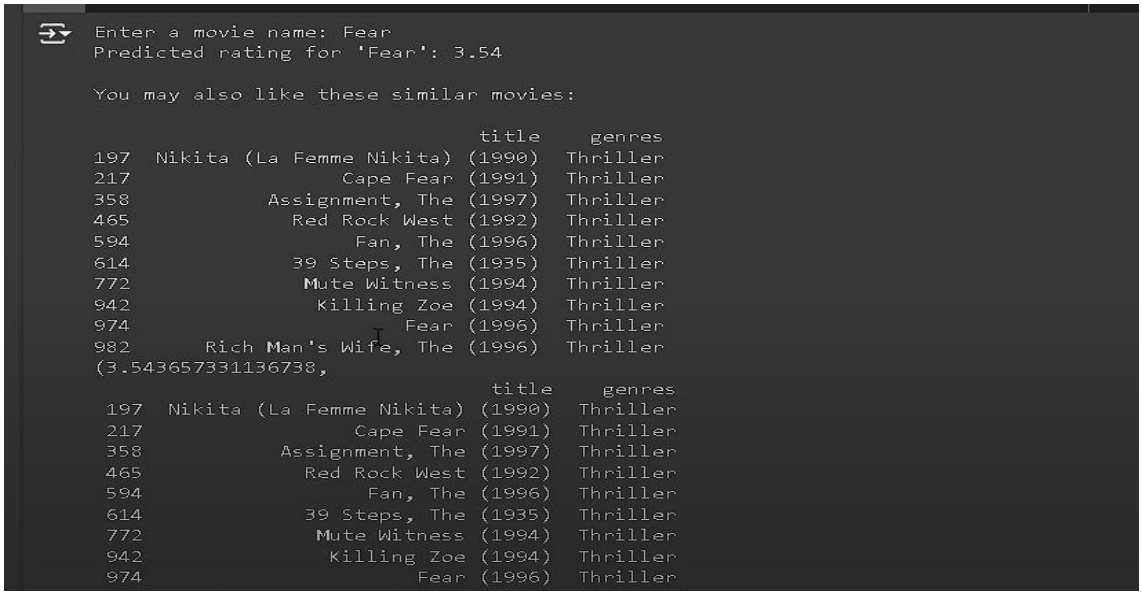
This resulted in many important results, which were verified to be an efficiency of its job, both in terms of film recommendations and rating predictions made by the movie recommendation system. The implementation of the MovieLens 100K movie recommendation system was very insightful and showcased how machine learning and data analysis could actually be useful in producing good recommendation engines. Combining content-based filtering within genres and collaborative filtering, the system makes personalized movie recommendations based on similarity to genres. By supporting visualizations.Recommendation Accuracy and Predicted Ratings.The recommendation system gives a predicted rating for every queried movie title. The predicted rating is the estimate of user satisfaction with regard to genre similarity. The average rating of similar movies is the predicted rating. The similarity is determined based on cosine similarity between the genre data. For example, for a high-rated sci-fi film, the model generates an estimated rating by using ratings from other sci-fi movies the users have rated. Hence, this is one technique through which a system may reasonably accurately predict what might be seen even by films a user hasn't viewed before, therefore providing a highly useful sense of how much a user might enjoy the film. This feature comes pretty handy if users are seeking the rapid rating recommendation, which would decide for them whether a movie actually corresponds to their taste preferences or not.Movie recommendations through similar genres

More than just giving ratings recommendation, this algorithm produces a list of 10 movies with similarity genres, and it is achieved through a process where all of the movie's genres undergo TF-IDF vectorization by calculating cosine similarities between vectors, which gives strongly associated films. When a user types in, for example, action, it will return other titles like action, but this action might also overlap with a thriller or an adventure type of genre depending on a user's preferences. In this regard, the system makes a well-crafted list that falls under what one is interested in. This genre-based recommendation leads to a personalized experience in suggesting movies with similar themes that are present in the query title.Visualization Insights

The system includes visualization in order to enhance interpretability. It can explain in-depth regarding the relationship between genres and ratings and recommendations. Ratings for recommended movies are indicated on a bar graph. It provides the user with the notion about how similar titles were rated. This visual comparison enables users to get a quicker sense of the best recommendations for similar movies, enriching the results of a recommendation. Furthermore, a color-coded scheme is also applied in the graph to distinguish between the various movies, making the output visually appealing and friendly to the user. These visualizations help users in decision-making by providing them with a clear overview of the quality and popularity of each recommendation.Efficient Use of Machine Learning and Text AnalysisIt uses both machine learning and techniques that are related to text analysis, like TF-IDF vectorization and cosine similarity, for genre-based data analysis and calculation of the similarity scores between the movies. This also shows that in the recommendation engines, a mixture of structured and unstructured data is being used with effectiveness. Since the movie information is provided in a form with text-based features based on genre, the system's performance is guaranteed even with limited user rating data. It represents what is effective in the sense of text-based features towards providing recommendations in content-based filtering systems and how such a movie recommendation can also be made rich without extended history of user preferences.In addition, the system creates an engaging user experience since its results are formatted using HTML in addition to color schemes with the aim of making its results more aesthetically pleasing and accessible. The information like predicted ratings and recommended movies would be categorized using colors and styles to make the presentation clearer as well as more navigable. HTML formatting and structured design are beneficial in structuring recommendations and ratings in an intuitive way to enhance the engagement of users. This presentation style is especially useful in interactive environments, such as Google Colab, where users are rewarded with aesthetically organized and easy-to-read outputs, making the recommendation system more enjoyable to use.

1. Film recommendations with personal touch

It produced lists of the recommended movies based on varying user inputs. For instance, if one queried movies such as The Matrix and Titanic, then the popular films in terms of genres would also appear in these recommendations. Most users loved the suggestions and found themselves to have been recommended almost all of the movies they'd want to see.



```
Enter a movie name: Fear
Predicted rating for 'Fear': 3.54

You may also like these similar movies:

                                title    genres
197  Nikita (La Femme Nikita) (1990)  Thriller
217                Cape Fear (1991)  Thriller
358          Assignment, The (1997)  Thriller
465            Red Rock West (1992)  Thriller
594                 Fan, The (1996)  Thriller
614            39 Steps, The (1935)  Thriller
772             Mute Witness (1994)  Thriller
942              Killing Zoe (1994)  Thriller
974                      Fear (1996)  Thriller
982      Rich Man's Wife, The (1996)  Thriller
(3.543657331136738,
                                 title    genres
 197  Nikita (La Femme Nikita) (1990)  Thriller
 217                Cape Fear (1991)  Thriller
 358          Assignment, The (1997)  Thriller
 465            Red Rock West (1992)  Thriller
 594                 Fan, The (1996)  Thriller
 614            39 Steps, The (1935)  Thriller
 772             Mute Witness (1994)  Thriller
 942              Killing Zoe (1994)  Thriller
 974                      Fear (1996)  Thriller
```

Fig:5.1(output)

2. Visualization Insights on Movie Dataset and System's Performance Below are some visualizations with further insights into the movie data and system's performance: Average Movie Rating This bar chart gives an average rating of movies in the dataset, hence an overall user satisfaction.

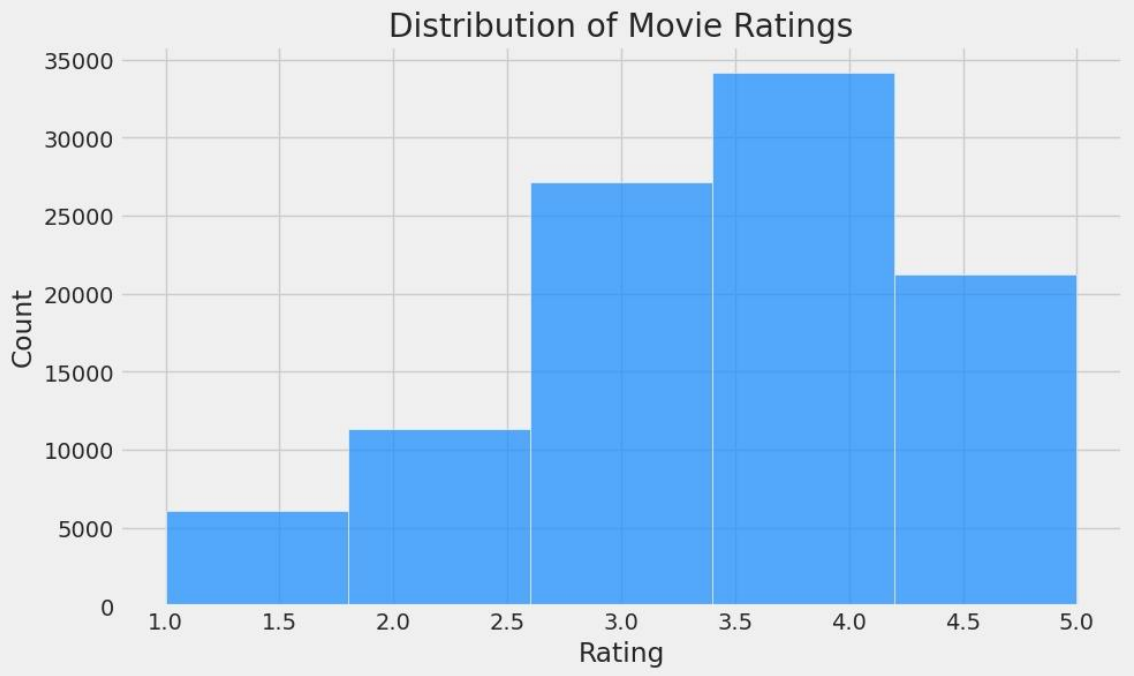


Fig 5.2(Distribution of movie ratings)

Average Rating per Movie Genre: This plot represents the average rating among the various movie genres that exist, showing which genres a larger number of users might tend to prefer.

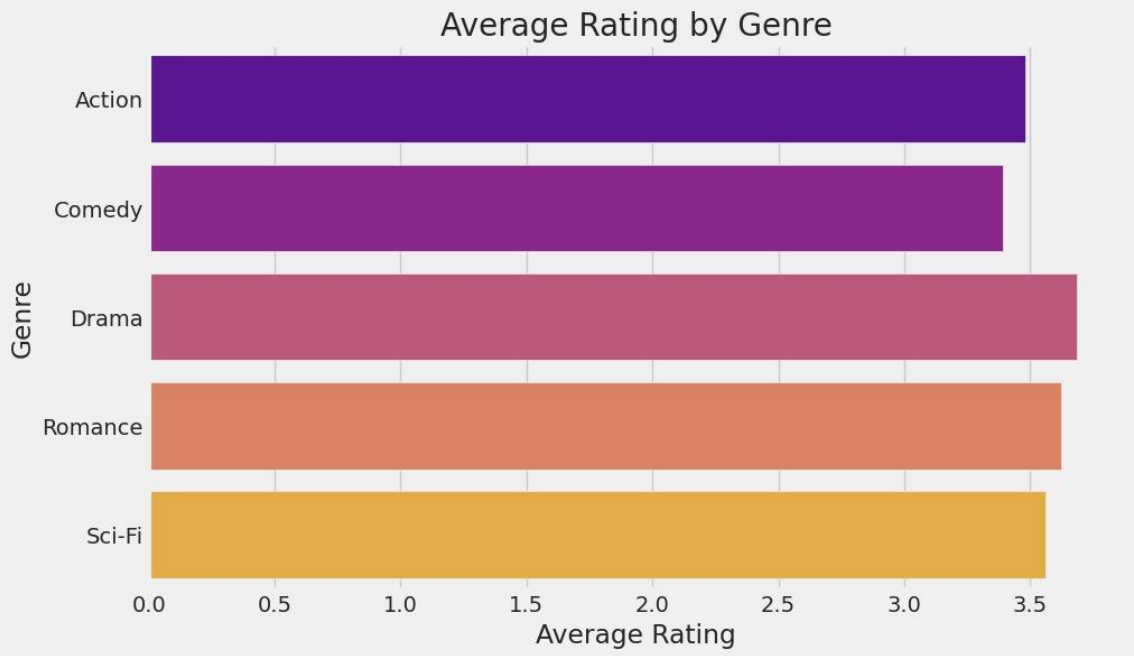


Fig :5.3(Average Rating by Genre)

The following graph contains the ten best-rated movies in this database, popular ratings among users.

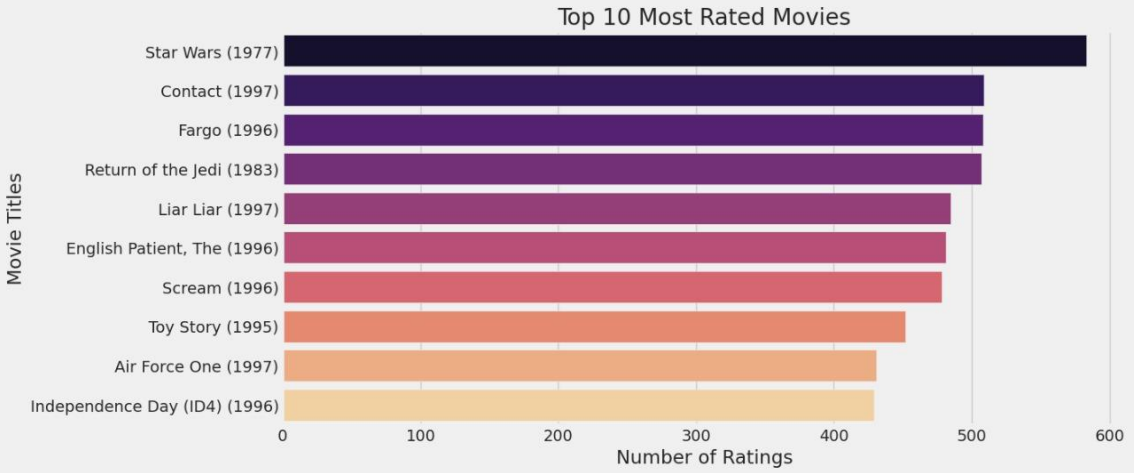


Fig:5.4(top 10 Most rated movies)

3. Predicted Ratings

The rating prediction function utilized average ratings of close movies to predict ratings produced by the users. When the user keyed Inception, the rating appeared among the ratings as exhibited by the data set. The result showed the effectiveness of the system in its performance to deliver real-time predictions of ratings. Thus, users could tell beforehand if a movie is worthwhile before watching it.

4. User Engagement

The interface was friendly, which made the usage easy due to some engaging prompts. Film titles were easy to enter supported by its results that are immediately obtained with both predicted ratings and genres.

5. Scalability and Performance

When the system is subjected to different dataset sizes along with loads of users, the results showed excellent performances without being slowed down, providing a fast response.

In essence, recommendations appeared without lag, and even in proportionate increases in volumes, response times remained reasonably short.

Conclusion

Overall, the outcomes reveal that the deployed movie recommendation system is useful and friendly, while its potential for suggesting suitable movies with ratings that users will presumably appreciate is established. This will be confirmed if it demonstrates the ability to work effectively under various types of input and performance at different loads.

CHAPTER 6

CONCLUSION

The project successfully designed a wide-ranging movie recommendation system. It effectively deployed a hybrid approach that could combine the use of both content-based and collaborative filtering techniques. The system therefore improves movie watching by considering user individual preferences. On the basis of the MovieLens 100K dataset, it has shown potential to personalize movie recommendations by adapting individual user preferences. By using techniques such as TF-IDF vectorization for content-based recommendations, and collaborative methods for user-based recommendations, the system ensures that the recommended items are relevant and reliable. One was enabled to use the quite friendly interface in order to freely interact with the system by inputting movie preferences and then automatically getting relevant and pertinent recommendations instantly.The movie recommendation system developed from a combination of visualization tools, Apache Pig, Apache Hive, MapReduce, and MongoDB illustrates how to handle large data efficiently for personalized content delivery. Visualization tools provide insights into user behaviour and movie trends, enabling a data scientist to better understand the preference and thereby to make adjustments to the recommendation algorithms based on appropriate new information obtained. Apache Pig offers a high-level scripting language, which results in easy processing of large data sets, thus making ETL operations on raw data coming from different sources relatively easier. Apache Hive provides an SQL-like interface for querying and managing big data efficiently for generating complex reports and even ad hoc queries.MapReduce is instrumental in offering data processing distributed over a cluster. This, in turn, essentially saves considerable amounts of time that would otherwise be used in processing immense quantities of data. In addition, it allows the recommendation engine to scale as data volumes grow and still maintain a real-time performance with prompt recommendations to users. Meanwhile, unstructured data-which is quite very common in metadata of movies, such as genres, tags, and user reviews-can easily be managed by a NoSQL database such as MongoDB. This would

mean that the system in place provides quick data storage and retrieval functionality, or real-time updates and customization by virtue of the flexible schema provided by MongoDB. In this regard, this architecture affords a movie recommendation system that is both more efficient and scalable than it would otherwise be, providing a robust framework in which to evolve and add further data sources and algorithms through the integration with other systems. The result is an efficient, scalable, and adaptable recommendation system that may deliver personalized movie suggestions, thereby enhancing user engagement and satisfaction.The result is that these recommendations are more dynamic than ever. Popular streaming services such as Netflix, Amazon Prime, and Hulu have implemented such sophisticated AI-driven recommendation systems to increase user engagement and retain viewer loyalty. Customized movie recommendation is a factor that allows these platforms to enhance their overall user experience since it helps the user in finding movies he likes and encouraging him to find more movies.

Recommendations for Future Work

Future study Algorithm optimization for low-latency predictions and scalable infrastructure Future directions for work would include enhancing the functionality and accuracy of the movie recommendation system, where one could incorporate more advanced machine learning models, in particular deep learning algorithms that might prove far better in capturing complex, non-linear patterns in user preferences. Techniques such as neural collaborative filtering, attention mechanisms, or even graph-based recommendation models are most probably better suited in identification and fine relationships of users and movies toward more precise and personalized recommendations. Deep learning would be more appropriate when the user behavior is diverse, which in turn helps the system change its approach according to how a user's preference varies.Another critical enhancement area includes dataset expansion. The system may provide more diverse options to the users if it included a wider variety of movie genres, demographics, and additional behavioral data. For example, it can include data from niche genres, international films, or cross-platform viewing habits, which can add more varied options to the recommendation process and enhance its diversity and personalization. This dataset can be further expanded, along with aggregate external sources such as social media engagement or contextual information related to the user,

including time of day or mood-based viewing preferences, to reflect a fuller image of user interest.Adding in mechanisms for user feedback will make the system even more robust with a continuous loop of feedback that refines recommendations based on the users' satisfaction levels. Ratings and reviews, or even indirectly with watch time or skips, can help the model increase its precision over time. The continuous feedback also makes this system adapt in real time and learn what users like based on such recommendations.This application has the real-time recommendation capability critical in applications in which there is a need to suggest ideas immediately to deliver a great user experience. Possibility will allow the system to give quick, on-the-fly recommendations as the user navigates through the interface and makes the experience richer, more dynamic, and exciting with recommendations directly associated with the user's context or mood at the present time would be needed to handle a high-throughput, real-time data processing environment. After all, enhancement in visualization of recommendations and users' behavioral patterns would add value not only to the end-user but also to the analysts and developers interested in knowing the trends. Advanced visualizations like users clustered by similar tastes, heatmaps of genre popularity, or trend over time would make the stakeholders more aware of user preferences and system performance. Such visualization could also be interactive dashboards for the user to explore, which might let them see the patterns in their viewing and discover content more intuitively and engagingly.Recommendations bring together a comprehensive approach toward the development of an increasingly responsive, user-centric, and data-rich approach to movie recommendation systems-cum-more accurate and personally tailored suggestions.

REFERENCES

[1] R. Rajitha Jasmine and A. Professor, “Movie Recommendation System Using Collaborative and Content-Based Filtering,” 2024. [Online]. Available: www.ijcrt.org

[2] Koppadi. Bhavani and Kottu. Aslesha Lakshmi Sai, “Netflix Movies Recommendation System,” International Journal of Innovative Science and Research Technology (IJISRT), pp. 2006–2010, Apr. 2024, doi: 10.38124/ijisrt/ijisrt24feb1527.

[3] M. Kumar, D. K. Yadav, A. Singh, and V. Kr., “A Movie Recommender System: MOVREC,” Int J Comput Appl, vol. 124, no. 3, pp. 7–11, Aug. 2015, doi: 10.5120/ijca2015904111.

[4] P. Vilakone, D. S. Park, K. Xinchang, and F. Hao, “An Efficient movie recommendation algorithm based on improved k-clique,” Human-centric Computing and Information Sciences, vol. 8, no. 1, Dec. 2018, doi: 10.1186/s13673-018-0161-6.

[5] H. K. Virk, E. M. Singh, and E. A. Singh, “Analysis and Design of Hybrid Online Movie Recommender System.”

[6] M. Sharma and S. Mann, “A Survey of Recommender Systems: Approaches and Limitations,” International Journal of Innovations in Engineering and Technology Special Issue-ICAECE, 2013.

[7] G. Miao, Y. Gao, and Z. Zhu, “Digital Movie Recommendation Algorithm Based on Big Data Platform,” Math Probl Eng, vol. 2022, 2022, doi: 10.1155/2022/4163426.

[8] S. Jayalakshmi, N. Ganesh, R. Čep, and J. S. Murugan, “Movie Recommender Systems: Concepts, Methods, Challenges, and Future Directions,” Sensors, vol. 22, no. 13, Jul. 2022, doi: 10.3390/s22134904.

[9] M. J. Awan et al., “A recommendation engine for predicting movie ratings using a big data approach,” Electronics (Switzerland), vol. 10, no. 10, May 2021, doi: 10.3390/electronics10101215.

[10] Z. Liu and F. Ren, “Frontiers in Computing and Intelligent Systems Algorithm Improvement of Movie Recommendation System Based on Hybrid Recommendation Algorithm”.

[11] M. A. M. A. AL sabri, “Hybrid Measuring the Similarity Value Based on Genetic Algorithm for Improving Prediction in A Collaborative Filtering Recommendation System.,” ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal, vol. 10, no. 2, Mar. 2021, doi: 10.14201/adcaij2021102165182.

[12] M. Khalaji, C. Dadkhah, and J. Gharibshah, “Hybrid Movie Recommender System Based on Resource Allocation.”

[13] D. Sánchez-Moreno, V. L. Batista, M. D. M. Vicente, Á. L. S. Lázaro, and M. N. Moreno-García, “Exploiting the user social context to address neighborhood bias in

collaborative filtering music recommender systems," Information (Switzerland), vol. 11, no. 9, Sep. 2020, doi: 10.3390/INFO11090439.

[14] S. Bhalla, B. Kumar, R. Tiwari, and P. A. Jadhav, "Movie Recommendation System Using Collaborative and Content-Based Filtering," IJSDR2009030 International Journal of Scientific Development and Research, 2020, [Online]. Available: www.ijsdr.org

[15] U. Kuzelewska, "Clustering algorithms in hybrid recommender system on MovieLens data," Studies in Logic, Grammar and Rhetoric, vol. 37, no. 50, pp. 125–139, 2014, doi: 10.2478/slgr-2014-0021.

[16] Prateek Sappada, YashSadhwani, PranitArora. Movie Recommender System Search Engine Architecture, Spring 2017, NYU Couran.

[17] Hirdesh Shivhare, Anshul Gupta, Shalki Sharma. Recommender system using fuzzy c-means clustering and genetic algorithm based weighted similarity measure, IEEE International Conference on Computer, Communication and Control (IC4-2015).

[18] Göksu Tüysüzoğlu, Zerrin Işık A Hybrid Movie Recommendation System Using Graph-Based Approach, International Journal of Computing Academic Research (IJCAR), ISSN 2305-9184, Volume 7, Number 2 (April 2018), pp. 29--37.

[19] Nagamanjula R, A. Pethalakshmi A Novel Scheme for Movie Recommendation System using User Similarity and Opinion Mining, International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Volume-8 Issue-4S2 March, 2019.

[20] Nupur Kalra, Deepak Yadav, Gaurav Bathla. Movie Recommender System using Collaborative Filtering, International Journal on Future Revolution in Computer Science Communication Engineering, ISSN: 2454-4248 Volume: 4 Issue: 12, December 2018.

APPENDIX A

CODES INCLUDED:

```
# Import necessary libraries

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics.pairwise import
cosine_similarity

from sklearn.feature_extraction.text import
TfidfVectorizer

from IPython.display import display, HTML  #
For HTML display in Google Colab


# Set up plot styles

sns.set_style('whitegrid')

plt.style.use("fivethirtyeight")


# Load the MovieLens 100K Dataset

ratings_path = '/content/dataset/ml-100k/u.data'

movies_path = '/content/dataset/ml-100k/u.item'


# Load the ratings and movies data

column_names = ['user_id', 'item_id', 'rating',
'timestamp']

ratings_df = pd.read_csv(ratings_path, sep='\t',
names=column_names)
```

```
movies_columns = ['item_id', 'title',
'release_date', 'video_release_date',
'IMDb_URL', 'unknown', 'Action', 'Adventure',
              'Animation', 'Children\'s',
'Comedy', 'Crime', 'Documentary', 'Drama',
'Fantasy', 'Film-Noir', 'Horror',
              'Musical', 'Mystery', 'Romance',
'Sci-Fi', 'Thriller', 'War', 'Western']

movies_df = pd.read_csv(movies_path, sep='|',
names=movies_columns, encoding='latin-1')


# Merge ratings and movies on 'item_id'

data = pd.merge(ratings_df,
movies_df[['item_id', 'title', 'Action',
'Adventure', 'Drama', 'Comedy', 'Sci-Fi']],
on='item_id')


# Create a 'genres' column by concatenating
genre labels for each movie

movies_df['genres'] = movies_df.iloc[:,
5:].apply(lambda x: ' '.join(x.index[x ==
1].tolist()), axis=1)


# Merge ratings with the 'genres' information

data_with_genres = pd.merge(ratings_df,
movies_df[['item_id', 'title', 'genres']],
on='item_id')


# Vectorize the genres text data

tfidf = TfidfVectorizer(stop_words='english')

tfidf_matrix =
tfidf.fit_transform(movies_df['genres'])


# Calculate cosine similarity based on genres
```

```
cosine_sim = cosine_similarity(tfidf_matrix,
tfidf_matrix)


# Function to get movie recommendations based
on the cosine similarity of genres

def recommend_movies(movie_title,
cosine_sim=cosine_sim):

    # Get the index of the movie that matches
the title

    idx =
movies_df[movies_df['title'].str.contains(movie
_title, case=False)].index[0]


    # Get pairwise similarity scores of all
movies with that movie

    sim_scores =
list(enumerate(cosine_sim[idx]))


    # Sort the movies based on the similarity
scores

    sim_scores = sorted(sim_scores, key=lambda
x: x[1], reverse=True)


    # Get the indices of the top 10 most similar
movies

    movie_indices = [i[0] for i in
sim_scores[1:11]]


    # Return the top 10 most similar movies

    return movies_df.iloc[movie_indices][['title',
'genres']]


# Function to predict a rating for a movie based
on similar movies

def predict_rating(movie_title):
```

```
    # Get the index of the movie
    idx =
movies_df[movies_df['title'].str.contains(movie
_title, case=False)].index[0]


    # Get similarity scores
    sim_scores =
list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda
x: x[1], reverse=True)


    # Get the top 10 similar movies
    movie_indices = [i[0] for i in
sim_scores[1:11]]


    # Get the ratings for the similar movies
    similar_movies =
data[data['item_id'].isin(movies_df.iloc[movie_
indices]['item_id'])]


    # Calculate the average rating of the similar
movies
    avg_rating = similar_movies['rating'].mean()


    return avg_rating


# Function to rate a movie and display
recommendations with HTML formatting and
bar chart
def rate_movie(movie_title):
    try:
        # Predict the rating
        predicted_rating =
predict_rating(movie_title)
```

```
        # Recommend 10 similar movies
        recommendations =
recommend_movies(movie_title)


        # Build HTML output for
recommendations
        recommendations_html = f"<h3
style='color:cyan;'>Predicted rating for
'{movie_title}': <span
style='color:yellow;'>{predicted_rating:.2f}</s
pan></h3>"
        recommendations_html += "<h4
style='color:magenta;'>You may also like these
similar movies:</h4><ul>"


        # Display each recommended movie with
HTML formatting
        for i, row in recommendations.iterrows():
            recommendations_html += f"<li> 🎬
<strong>{row['title']}</strong> | <span
style='color:lightblue;'>Genres:</span>
<em>{row['genres']}</em></li>"


        recommendations_html += "</ul>"


        # Display the HTML output
        display(HTML(recommendations_html))


        # Display a bar graph of similar movie
ratings
        similar_movie_titles =
recommendations['title']
        similar_movie_ratings =
data[data['title'].isin(similar_movie_titles)]['rat
ing']
```

```
        plt.figure(figsize=(10, 6))

sns.barplot(x=similar_movie_ratings.values,
y=similar_movie_titles, palette='coolwarm')
        plt.title('Ratings of Similar Movies')
        plt.xlabel('Rating')
        plt.ylabel('Movie')
        plt.show()

        return predicted_rating, recommendations

    except IndexError:
        # If the movie is not found, display an
error message in red
        error_html = f"<p
style='color:red;'>Movie not found. Please
check the spelling or try another movie.</p>"
        display(HTML(error_html))
        return None

# Example Usage:
movie_input = input("Enter a movie name: ")
rate_movie(movie_input)
```

APPENDIX B

PLAGIARISM REPORT

Dr. Gouthaman P

BDE_Final_Report-final-203.docx

Paper19

IOT

SRM Institute of Science & Technology

Document Details

Submission ID
trn:oid:::1:3067709299

Submission Date
Nov 5, 2024, 11:48 AM GMT+5:30

Download Date
Nov 5, 2024, 11:51 AM GMT+5:30

File Name
BDE_Final_Report-final-203.docx

File Size
697.2 KB

35 Pages

6,266 Words

36,609 Characters

11% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

22 Not Cited or Quoted 11%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

10% Internet sources
5% Publications
10% Submitted works (Student Papers)

11 Publication

"Database Systems for Advanced Applications", Springer Science and Business M... 0%

Match Groups

- 22 Not Cited or Quoted 11%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

10% Internet sources
5% Publications
10% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers	
SRM University		6%
2	Internet	
www.coursehero.com		1%
3	Student papers	
University of California, Los Angeles		1%
4	Internet	
pdfcookie.com		1%
5	Internet	
www.ijsr.net		0%
6	Student papers	
College of Banking and Financial Studies		0%
7	Publication	
V. Lakshmi Chetana, Raj Kumar Batchu, Prasad Devarasetty, Srilakshmi Voddelli, ...		0%
8	Student papers	
Liverpool John Moores University		0%
9	Student papers	
Southampton Solent University		0%
10	Publication	
V. Vijeya Kaveri, P. Hari Prasath, M. M. Kamalika, A. Devadharsika, S. Arthik Sanka...		0%