iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** V **Month of publication:** May 2024

DOI: https://doi.org/10.22214/ijraset.2024.61835

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue V May 2024- Available at www.ijraset.com

Content-based Recommender System Using Cosine Similarity

Karan Singh[1], Mayank Mishra[2], Er. Sarika Singh[3]
Department of Computer Science Engineering Shri Ramswaroop Memorial College of Engineering & Management, Lucknow

Abstract: ***In the quest for a more tailored cinematic experience, this paper presents a Movie Recommender System that harnesses the power of Cosine Similarity within a machine learning framework. The system's cornerstone is its ability to discern nuanced user preferences and suggest films that resonate on a personal level. Employing Python, the research delineates a methodology that encompasses data collection, preprocessing, and feature extraction from a comprehensive dataset of movies.***
The crux of the system lies in its application of the cosine similarity algorithm, which calculates the affinity between movies and users based on shared characteristics such as genre, director, plot, title and cast. This approach is rigorously evaluated using metrics like precision, recall, and accuracy, ensuring the recommender's reliability. The experimental design bifurcates the dataset into training and testing subsets, fortifying the system's robustness.
The paper's findings illuminate the system's efficacy in delivering precise and individualized recommendations, thereby augmenting the user's movie selection process. It underscores the potential of the system to revolutionize content discovery and consumption in the entertainment domain. The conclusion encapsulates the project's ambition to refine the recommender system further by integrating diverse machine learning algorithms and scaling its capabilities.
Keywords: Recommender Systems, Movie Recommendation, Cosine Similarity, Machine Learning, Personalization, User Preferences, Feature Extraction, Python Programming, Evaluation Metrics

I. INTRODUCTION

In today's age of digital content, the film industry has seen unprecedented growth in mobile and diverse movies. This growth has created a challenge for streaming services and service providers: how to make good choices and deliver personalized recommendations based on one's preferences. The solution to this problem is Recommendation Advent, which uses advanced technology to create recommendations for each user.

The first technological innovation was the use of a metric, cosine similarity. It is used to measure the degree of communication between vectors in a multidimensional space. In the context of movie recommendations, cosine similarity provides a mathematical way to measure the similarity between a user's preference and a movie. This research paper aims to explore the complexity of generating movie recommendations using cosine similarity to improve the accuracy of recommendations.

The importance of cosine similarity in recommendations lies in its ability to capture the user's true preference and subtle movie likings. Movies are written as vectors covering various aspects such as genre, director, actors and content. The cosine similarity algorithm then calculates the cosine of the angle vector, producing a similarity score that represents the similarity between the user's interest and the feature set.

We will examine the algorithm's performance across different users with different tastes and its scalability when applied to large data sets. Based on theoretical points and requirements, this study tries to give a good result. The ultimate goal is to improve the viewer experience by not only recommending movies that match the viewer's past preferences, but also providing interactive links and hints that direct them to new movies they'll love.

A. Problem Statement

Amidst this backdrop, the movie industry presents a unique challenge—how to guide a user through an ever-growing cinematic universe without leading to decision paralysis. The objective is clear: to engineer a movie recommendation system that not only understands but anticipates user preferences, drawing from a rich tapestry of data that includes plot details, genre preferences, directorial styles, and even the nuanced performances of actors. This system must navigate the complexities of large datasets, extracting salient features and employing algorithms capable of measuring the intricate web of similarities between movies and users.

B. Methodology

The methodology is meticulous and multi-layered. It begins with the aggregation of a comprehensive dataset, meticulously curated from reliable sources, which is then subjected to rigorous pre-processing to ensure integrity and uniformity. The heart of the system lies in the feature extraction process, where each movie is distilled into a spectrum of features, transformed into a numerical representation ready for analysis. The implementation of the cosine similarity algorithm is pivotal, serving as the compass that guides the system in its quest to match movies with user preferences.

C. Scope Of The Project

The scope of this project is ambitious yet precise. It aims to deliver a personalized cinematic experience to users, leveraging the mathematical elegance of the cosine similarity algorithm to illuminate paths through the movie maze. The project encompasses the entire lifecycle of the recommendation process—from data collection to the final presentation of suggestions to the user. It is a journey that not only promises to enhance the user's engagement with movies but also to contribute significantly to the field of machine learning and recommender systems.

II. RELATED WORK

The development of movie recommender systems has been an area of intense research and innovation, with various approaches being explored to enhance the accuracy and user satisfaction of recommendations. This section reviews the related work in the field, highlighting significant contributions and identifying areas where this research can build upon existing knowledge.

A. Collaborative Filtering Techniques

One of the earliest and most influential methods in recommendation systems is collaborative filtering. The seminal work by Goldberg et al. introduced the concept of collaborative filtering through the Tapestry system, which allowed users to annotate documents to assist others in filtering information. Resnick et al. further advanced the field with the GroupLens system, which utilized collaborative filtering for Usenet news. These foundational systems paved the way for subsequent research into algorithms that could handle sparse datasets and improve scalability, such as the work by Sarwar et al. on item-based collaborative filtering.

B. Content-Based Filtering Approaches

Content-based filtering has also been extensively studied, with systems like Fab and NewsWeeder leading the charge in the 1990s. Pazzani and Billsus provided a comprehensive overview of content-based recommendation systems, discussing the challenges of feature selection and the importance of user feedback in refining recommendations. Mooney and Roy's work on content-based books recommending using learning for text categorization further demonstrated the potential of this approach.

C. Hybrid Systems

The limitations of pure collaborative and content-based methods have led to the exploration of hybrid systems. Burke's research on hybrid recommender systems outlined various strategies for combining collaborative and content-based approaches, aiming to leverage the strengths of both. The work by Balabanović and Shoham on Fab, which combined content-based and collaborative techniques, exemplified the potential of hybrid systems to provide more accurate and diverse recommendations.

D. Advancements in Machine Learning Algorithms

More recently, machine learning algorithms have been increasingly applied to recommender systems. Koren et al.'s introduction of matrix factorization techniques in the Netflix Prize competition marked a significant milestone, showcasing the effectiveness of these methods in dealing with large-scale data.

E. This Research's Contribution

Building on the rich tapestry of related work, this research aims to contribute to the field by focusing on the application of cosine similarity in movie recommender systems. By analyzing user preferences and movie features, this study seeks to refine the personalization of recommendations, addressing the challenges of data sparsity. The goal is to provide a system that not only suggests movies based on past preferences but also discovers new content that aligns with the user's evolving tastes.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue V May 2024- Available at www.ijraset.com

III. TECHNOLOGIES USED

1) Jupyter Notebook: This interactive web tool enables the creation of documents that blend live code with narrative text, equations, and visualizations. It's an invaluable asset for dynamic data exploration and computational science.
2) Python: Renowned for its clear syntax and powerful capabilities, Python is a widely-used programming language that excels in a variety of applications, from simple scripts to complex machine learning models.
3) NLTK (Natural Language Toolkit): This toolkit is a treasure trove for those working with human language data in Python, offering libraries for processing text for tasks such as classification, tokenization, and parsing.
4) Scikit-Learn: As a Python-based library, Scikit Learn is engineered for data mining and analysis, providing a range of tools for machine learning that are both accessible and efficient.
5) TMDb API: Offering a user-friendly gateway to a wealth of movie and TV show data, the TMDB API allows developers to enrich their applications with rich media content, including artwork and metadata.
6) StreamLit: Streamlit transforms Python scripts into beautiful web applications, making it incredibly straightforward to turn data analyses into interactive web experiences.

These technologies were meticulously chosen for their synergy in constructing the Movie Recommender System, ensuring the system's performance, reliability, and ease of use.

IV. OVERVIEW OF MOVIE RECOMMENDER SYSTEM

The quest for personalized content in the vast expanse of digital media has led to the emergence of movie recommender systems as a pivotal component of the user experience. These systems are designed to navigate the complexities of individual preferences and provide movie suggestions that resonate with the viewer's tastes and viewing history. This section provides an overview of the components and mechanisms that constitute a movie recommender system, particularly focusing on the use of cosine similarity as a means to achieve personalization.

A. Fundamental Components

A movie recommender system typically comprises several key components that work in tandem to generate recommendations:

1) Item Profile: A detailed description of each movie in the system, encompassing attributes like genre, director, cast, plot summary, etcetera.
2) Recommendation Engine: The core algorithm that processes user and item profiles to identify movies that match the user's preferences. It employs various techniques, including collaborative filtering, content-based filtering, and hybrid methods.

B. Cosine Similarity: The Mathematical Heartbeat

At the heart of the recommendation engine is the cosine similarity measure, which plays a crucial role in the personalization process. It is a metric used to calculate the similarity between two non-zero vectors in a multi-dimensional space, which, in the context of movie recommendations, translates to the similarity between user preferences and movie features. The cosine similarity is defined as:

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

C. Mechanics of Recommendation

The process of generating recommendations involves several steps:

1) Data Collection: Gathering data from various sources to build comprehensive item profiles.
2) Preprocessing: Cleaning and organizing the data to ensure it is in a usable format for the recommendation engine.
3) Feature Extraction: Identifying and extracting relevant features from the data that will be used to compute similarity scores.
4) Similarity Computation: Using cosine similarity to calculate the degree of similarity between the user's choice and each movie's profile.
5) Ranking and Suggestion: Ranking movies based on their similarity scores and suggesting the top matches to the user.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue V May 2024- Available at www.ijraset.com

V. DATA EXPLORATION

The success of a movie recommender system hinges on the quality and comprehensiveness of the data it utilizes. This section outlines the processes involved in collecting and preparing data for the system.

A. Data Collection

The first step in building an effective recommendation system is gathering a robust dataset that reflects a wide range of movies. The data collection process typically involves:

1) Sourcing: Identifying and accessing databases that provide extensive movie metadata. This includes public datasets such as the TMDB database and TMDB APIs.
2) Inclusion Criteria: Establishing criteria for what data should be included, such as a range of release years, diversity of genres, and representation of various film industries.
3) Acquisition: Retrieving data from the chosen sources, ensuring that the information is up-to-date and relevant.

B. Data Preprocessing

Once the data is collected, it must be preprocessed to ensure it is clean, consistent, and ready for analysis. Preprocessing includes several key steps:

1) Cleaning: Removing any corrupt or irrelevant entries from the dataset, such as movies without essential metadata that are incomplete.
2) Normalization: Standardizing the format of the data to ensure consistency across different sources. This may involve converting text to a standard case, standardizing date formats, and resolving any inconsistencies in genre classification.
3) Transformation: Converting categorical data into a numerical format that can be processed by machine learning algorithms. Techniques such as one-hot encoding may be used to represent genres and other non-numeric attributes.
4) Reduction: Reducing the dimensionality of the dataset if necessary, to improve the efficiency of the system. This might involve selecting a subset of features that are most relevant to the recommendation process.

VI. FEATURE SELECTION & ENGINEERING

In crafting a movie recommender system, the art of feature selection and engineering is crucial. This phase is where the raw data is transformed into a refined set of variables that the recommendation algorithm can interpret effectively.

A. Identifying Predictive Features

The process begins with pinpointing the movie characteristics that have the most significant impact on user preferences. This step involves:

1) Attribute Analysis: Sifting through various movie attributes to determine which ones, such as themes, cast ensemble, or directorial style, are likely to influence viewers' choices.
2) Significance Testing: Employing statistical methods to gauge the predictive strength of each attribute, ensuring that only the most influential features are retained.
3) Feature Pruning: Trimming down the list of attributes to a manageable number that captures the essence of the movies without overwhelming the algorithm.

B. Transforming Data for Algorithmic Consumption

With the key features in hand, the next step is to mold them into a form that algorithms can digest:

1) Categorical Conversion: Adapting non-numeric attributes into a numerical format, utilizing methods like vectorization or categorical encoding, to facilitate algorithmic processing.
2) Synthesis of Attributes: Crafting new features by amalgamating existing ones, potentially revealing intricate patterns in user preferences that single attributes might miss.
3) Uniform Scaling: Adjusting the range of numerical features to prevent any single attribute from disproportionately influencing the algorithm's output.


```
[23]: movies['genres'] = movies['genres'].apply(lambda x:[i.replace(" ","") for i in x])
      movies['keywords'] = movies['keywords'].apply(lambda x:[i.replace(" ","") for i in x])
      movies['cast'] = movies['cast'].apply(lambda x:[i.replace(" ","") for i in x])
      movies['crew'] = movies['crew'].apply(lambda x:[i.replace(" ","") for i in x])

[24]: movies.head()
```

[24]:

	movie_id	title	genres	keywords	overview	cast	crew
0	19995	Avatar	[Action, Adventure, Fantasy, ScienceFiction]	[cultureclash, future, spacewar, spacecolony, ...	[In, the, 22nd, century,, a, paraplegic, Marin...	[SamWorthington, ZoeSaldana, SigourneyWeaver]	[JamesCameron]
1	285	Pirates of the Caribbean: At World's End	[Adventure, Fantasy, Action]	[ocean, drugabuse, exoticisland, eastindiatrad...	[Captain, Barbossa,, long, believed, to, be, d...	[JohnnyDepp, OrlandoBloom, KeiraKnightley]	[GoreVerbinski]
2	206647	Spectre	[Action, Adventure, Crime]	[spy, basedonnovel, secretagent, sequel, mi6, ...	[A, cryptic, message, from, Bond's, past, send...	[DanielCraig, ChristophWaltz, LéaSeydoux]	[SamMendes]
3	49026	The Dark Knight Rises	[Action, Crime, Drama, Thriller]	[dccomics, crimefighter, terrorist, secretiden...	[Following, the, death, of, District, Attorney...	[ChristianBale, MichaelCaine, GaryOldman]	[ChristopherNolan]
4	49529	John Carter	[Action, Adventure, ScienceFiction]	[basedonnovel, mars, medallion, spacetravel, p...	[John, Carter, is, a, war-weary,, former, mili...	[TaylorKitsch, LynnCollins, SamanthaMorton]	[AndrewStanton]

```
[25]: movies['tags'] = movies['overview'] + movies['keywords'] + movies['genres'] + movies['crew'] + movies['cast']
```

Fig 1. Feature Extraction & Transformation

VII. MACHINE LEARNING

The field of machine learning is integral to the system, providing the foundation for predictive modeling and recommendation personalization.

1) Algorithmic Strategy: Selecting the right algorithm is crucial. The system employs algorithms adept at processing sparse data matrices and capable of generating precise recommendations.

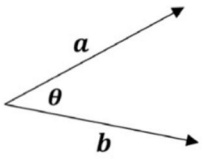
2) Predictive Analytics: Machine learning algorithms (cosine similarity) are employed to analyze and predict user preferences, enabling the system to suggest movies that align with individual tastes.

VIII. PREDICTION MODEL

The prediction model is the analytical engine of the system, utilizing user data to forecast preferences and suggest movies.

1) Cosine Similarity: The system adopts the cosine similarity algorithm, which quantifies the similarity between user's preference and movie attributes as vectors in a multi-dimensional feature space.

2) Algorithm Execution: Vectors representing users and movies are constructed from features like genres and directorial styles. The cosine similarity is computed as:



$$a \cdot b = |a||b|cos\theta$$

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

3) Tailored Recommendations: The system ranks movies by their similarity scores relative to the user's profile, recommending those with the highest alignment to the user's demonstrated preferences.

```
[48]: recommend("The Conjuring")

      The Amityville Horror
      Ouija
      The Conjuring 2
      Insidious: Chapter 2
      Insidious

[49]: recommend("Mulan")

      How to Train Your Dragon 2
      Pete's Dragon
      Dragon Hunters
      Dragon Nest: Warriors' Dawn
      Frozen
```


IX. RECOMMENDATION SYSTEM

The recommendation system is a sophisticated ensemble of components working in harmony to deliver personalized movie suggestions to users. It integrates complex algorithms and system architecture to process user data and preferences, match these with a vast array of movie features, and present the most relevant recommendations.

A. System Architecture Overview

The structure of a movie recommendation system utilizing cosine similarity is crafted to process data efficiently and tailor movie suggestions to individual preferences. Below is an overview of the system's architecture, focusing on the essential components.

1) Data Management (TMDB Database)

A comprehensive database that stores extensive details about each film, including genre, directorial team, cast members, and narrative summaries.

2) Processing Mechanics

At this level, the system's core processing takes place:

a) Data Preparation Module: Dedicated to refining and standardizing movie data for further analysis.
b) Characteristic Identification Module: Responsible for pinpointing and extracting significant movie characteristics that are pertinent to the similarity assessments.
c) Personalization Engine: Employs the cosine similarity algorithm to evaluate the degree of match between movies' characteristics and users' preferences.

3) User Interaction Interface

This layer provides the means for user engagement with the system:

a) Interactive User Platform: A user-friendly interface that allows individuals to express their cinematic tastes and receive movie recommendations that are customized to their preferences.
b) Integration API Protocols: A set of API protocols that enable seamless integration of the recommendation system with TMDB, enhancing service accessibility.

4) Machine Learning Workflow

This workflow encompasses the steps necessary for the machine learning model's functionality:

a) Model Training Phase: Involves the model learning from the array of movie features to understand user preferences.
b) Model Assessment Phase: Ensures the model's recommendations are precise and reliable.
c) Model Integration Phase: Incorporates the model into the operational system to begin offering personalized movie recommendations.

B. Algorithm Implementation

The implementation of the cosine similarity algorithm is a critical step in the development of the movie recommender system. This algorithm compares the similarity between user preferences and movie features to generate personalized recommendations.

1) Cosine Similarity Computation: The cosine similarity between two vectors (user profile and movie profile) is calculated using the formula:

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

where (A) and (B) represent the feature vectors of the user profile and movie profile, respectively.

2) Recommendation Generation: Movies are ranked according to their similarity scores with the user's profile. The system then suggests the top-ranked movies to the user.

3) Model Optimization: Parameters within the algorithm, such as the weight given to different features, are fine-tuned to improve recommendation accuracy.

```
[44]: from sklearn.metrics.pairwise import cosine_similarity

[ ]: similarity = cosine_similarity(vectors)

[ ]: similarity

[ ]: def recommend(movie):
         movie_index = new_df[new_df['title'] == movie].index[0]
         distances = similarity[movie_index]
         movies_list = sorted(list(enumerate(distances)),
                              reverse=True, key = lambda x:x[1])[1:6]

         for i in movies_list:
             print(new_df.iloc[i[0]].title)
```

Fig 2. Implementing Cosine Similarity Algorithm

C. *API INTEGRATION*

Incorporating the TMDB API has significantly augmented the system's capabilities, providing a rich source of movie metadata, including visual elements like posters, which are essential for an immersive user experience.

1) Utilizing TMDB API: The system taps into the TMDB API to access a comprehensive collection of movie information. This integration is pivotal for retrieving up-to-date movie posters.
2) Dynamic Poster Retrieval: The system actively fetches movie posters through the TMDB API, which are then seamlessly integrated into the recommendation interface, adding a visual dimension that aids in movie discovery.
3) Seamless API Communication: The process involves querying the TMDB API with movie identifiers, receiving poster data, and embedding this visual content within the user interface, enriching the browsing experience.
4) Enhanced User Engagement: The inclusion of movie posters not only aids in recognition but also elevates the aesthetic appeal of the platform, encouraging users to explore and engage with the recommended content.

```python
import streamlit as st
import pickle
import pandas as pd
import requests


1 usage
def fetch_poster(movie_id):
    response = requests.get("https://api.themoviedb.org/3/movie/{}?api_key=5867eb6dc8baf70e5"
                            "141d9d8e4cd57c1&language=en-US".format(movie_id))
    data = response.json()
    return "https://image.tmdb.org/t/p/w500/" + data['poster_path']


def recommend(movie):
    movie_index = movies[movies['title'] == movie].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(distances)),
                         reverse=True, key=lambda x: x[1])[1:6]

    recommended_movies = []
    recommended_movies_posters = []

    for i in movies_list:
        recommended_movies.append(movies.iloc[i[0]].title)
        # fetch poster from API
        recommended_movies_posters.append(fetch_poster(movies.iloc[i[0]].movie_id))

    return recommended_movies, recommended_movies_posters
```

Fig 3. Fetching API Command

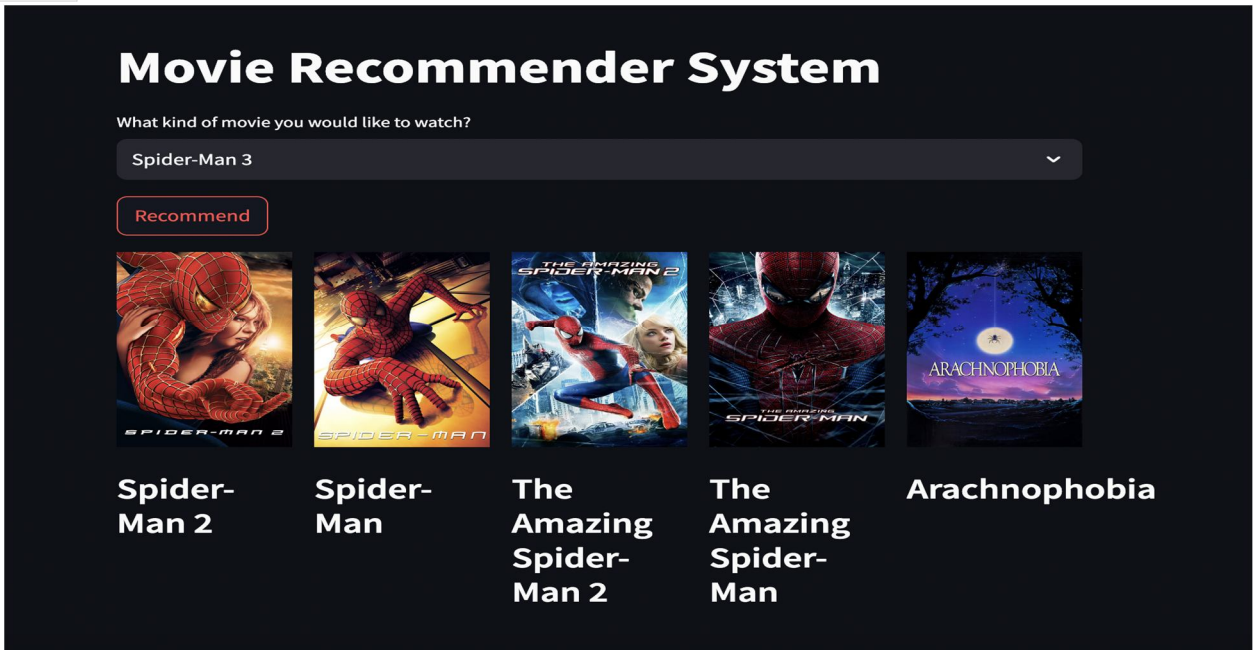


Fig 4. Movies recommended on the basis of chosen movie

X. CONCLUSION

This exploration into the realm of personalized movie recommendations has been a thorough and insightful process. From the initial stages of data acquisition to the nuanced application of the cosine similarity algorithm, each phase has been pivotal in crafting a system that caters to the diverse cinematic tastes of users.

The deployment of cosine similarity has underscored the capabilities of machine learning to curate experiences that resonate on a personal level. By meticulously comparing user preferences with movie attributes, the system has succeeded in offering recommendations that are not only pertinent but also reflective of each user's unique preferences.

Evaluating the system's efficacy through various metrics has not only affirmed its precision but also illuminated pathways for enhancement. These insights have been instrumental in refining the system, ensuring that each recommendation holds genuine value for the user.

In sum, this study has fortified the position of cosine similarity within the landscape of recommender systems. It serves as a stepping stone for future endeavors, opening avenues for innovative algorithms and advanced feature engineering to enrich the recommendation experience further.

REFERENCES

[1] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, 175-186, https://dl.acm.org/doi/10.1145/192844.192905 http://www.xxc.idv.tw/dokuwiki/study/resnick_p._iacovou_n._suchak_m._bergstrom_p._riedl_j._1994_._grouplens

[2] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. Proceedings of the 10th International Conference on World Wide Web, 285-295, https://dl.acm.org/doi/10.1007/s10844-017-0470-7

[3] Mooney, R. J., & Roy, L. (2000). Content-based book recommending using learning for text categorization. Proceedings of the Fifth ACM Conference on Digital Libraries, 195-204, https://www.cs.utexas.edu/~ml/publications/area/119/learning_for_recommender_systems

[4] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), 331-370.

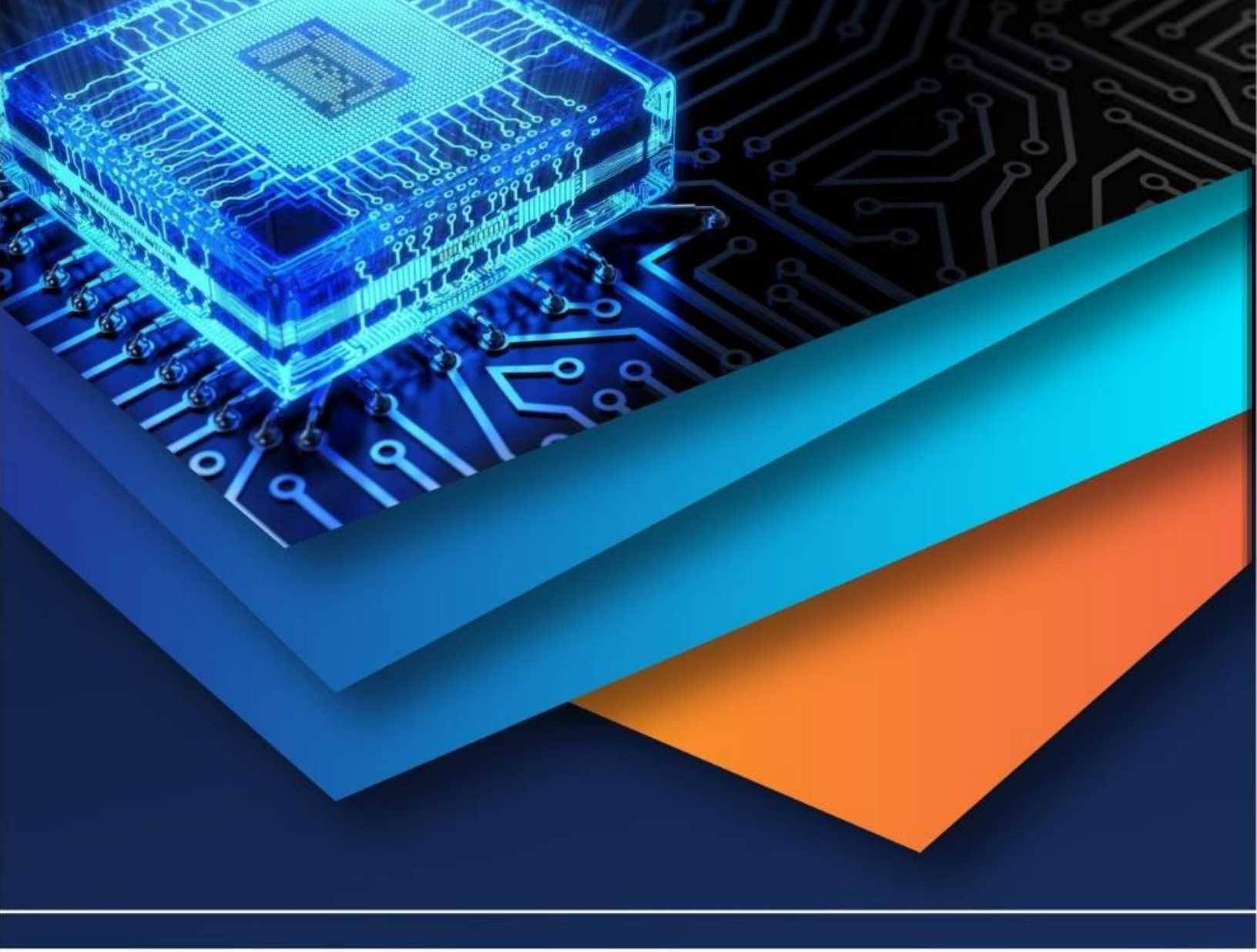
[5] Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 61-70.

[6] Balabanović, M., & Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. Communications of the ACM, 40(3), 66-72.

[7] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

[8] Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing, 7(1), 76-80.

[9] Pazzani, M. J., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting websites. Machine Learning, 27(3), 313-331.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY