**Mini Project Report on**



**MOVIE RECOMMENDATION SYSTEM**



**Submitted in partial fulfillment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**Student Name**  **University Roll No.**

**VEDANSH SHINDE 2023703**

***Under the Mentorship of***

**Dr. Noor Mohd**

**Associate Professor, Computer Science & Engineering**

**Graphic Era (Deemed to be University), Dehradun**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**January-2025**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled **“Movie Recommendation System”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineeringof the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of **Dr. Noor Mohd, Associate Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Name University Roll no

Vedansh Shinde 2023703

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Description** | **Page** |
| 1 | [Introduction](#_INTRODUCTION) | [4-5](#_INTRODUCTION) |
| 2 | [Literature Survey](#_LITERATURE_SURVEY) | [6-8](#_LITERATURE_SURVEY) |
| 3 | [Methodology](#_METHODLOGY) | [9-15](#_METHODLOGY) |
| 4 | [Result & Discussion](#_RESULT_&_DISCUSSION) | [16-17](#_RESULT_&_DISCUSSION) |
| 5 | [Conclusion](#_CONCLUSION_&_FUTURE) | [18-19](#_CONCLUSION_&_FUTURE) |
|  | [References](#_REFRENCES) | [20](#_REFRENCES) |

**Chapter 1**

# INTRODUCTION

**1.1 Background**

The rapid growth of digital platforms has completely transformed how we consume content, especially in the entertainment industry. Platforms like Netflix and Amazon Prime now offer massive catalogues of movies, TV shows, and documentaries. While this incredible variety gives us access to more content than ever before, it also comes with a downside: **decision fatigue**. With so many options, it can feel overwhelming to pick what to watch, leaving users frustrated and sometimes even disengaged.

This is where recommendation systems come in. These smart systems use algorithms to understand user preferences and analyse details about available content, helping to deliver personalized suggestions. By making the process of finding content easier and more enjoyable, recommendation systems improve user satisfaction, keep people engaged, and encourage them to keep coming back. For example, Netflix has revealed that about 80% of the content people watch is driven by its recommendation engine—a clear sign of how crucial these systems are for retaining users.

Figure .1

In this project, we’re focusing on building a **Movie Recommendation System** using ***content-based filtering***. Content-based filtering analyses specific features of the content itself—like genres, cast, and crew—to recommend movies. This approach works well for tackling the **cold-start problem**, where new users or lesser-known, movies might otherwise struggle to receive recommendations. To make the recommendations even more accurate, we’ll use **cosine similarity** to measure how closely related different movies are based on their features.

**1.2 Significance of Recommendation Systems**

Recommendation systems have become a core part of many online experiences, helping people find what they need or love across different industries:

* **Entertainment**: Platforms like Netflix, YouTube, and Spotify use recommendation systems to serve up content that keeps users engaged and coming back for more.
* **E-Commerce**: Online stores like Amazon and eBay suggest products based on what users have bought or browsed, making shopping easier and boosting sales.
* **Education**: Learning platforms tailor their recommendations to suggest courses or materials that match individual interests and goals, improving learning outcomes.

**1.2.1 Why Recommendation Systems Matter in Entertainment**

In the world of entertainment, these systems offer huge benefits:

* **Making Life Easier for Users**: Personalized recommendations help people find movies or shows that match their taste, saving time and reducing frustration.
* **Keeping Users Engaged**: By constantly showing relevant content, platforms make sure users stick around longer and come back often.
* **Driving Revenue**: Highlighting premium or exclusive content encourages users to make purchases or upgrade their subscriptions.
* **Standing Out in the Crowd**: A strong recommendation system can give a platform an edge in a highly competitive market.

Take YouTube, for example. Over 70% of the videos people watch on the platform come from its recommendations. This shows just how much of an impact these systems can have on user engagement.

**1.3 Objectives**

The main goals of this project are:

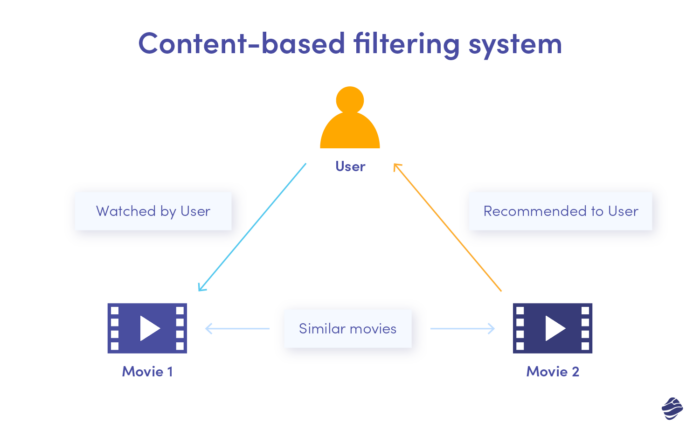
1. **Build an Efficient Recommendation System**:
   * Create a content-based filtering system that uses features like genres, cast, and crew to deliver accurate and personalized recommendations.
   * Use cosine similarity to measure how closely related different movies are, ensuring precise suggestions.
2. **Tackle Common Challenges**:

Figure 1.2

* + Solve the **cold-start problem** by focusing on content features rather than user behaviour, making recommendations effective for new users and less popular movies.
  + Handle **data sparsity** by leveraging rich metadata and efficient computational methods.
  + Ensure the system is **scalable** to handle large datasets as the user base grows.

1. **Improve User Satisfaction**:
   * Provide recommendations that align with individual tastes, making the process of discovering content enjoyable and engaging.
   * Design an intuitive interface that enhances the overall user experience.

By achieving these objectives, this project aims to create a powerful recommendation system that addresses current challenges while paving the way for future innovations.

**Chapter 2**

# LITERATURE SURVEY

**2.1 Historical Background**

Recommendation systems have come a long way since their early days. The first systems were simple and rule-based, relying on manually created rules to generate suggestions. For instance, a music store might recommend songs based on a genre selected by the user. These systems were effective for small datasets, but they struggled to adapt as user needs grew more complex and data became more abundant.

In the 1990s, the landscape changed with the advent of **collaborative filtering**. This method allowed systems to analyse user preferences and behaviours automatically, making recommendations based on patterns found in the data. A well-known example is Amazon’s introduction of the “customers who bought this item also bought” feature, which became a cornerstone of online shopping experiences. This approach was revolutionary because it required minimal manual input while improving the relevance of recommendations.

Another significant milestone was the **Netflix Prize** competition in 2006. Netflix challenged researchers to improve its recommendation algorithm, offering a substantial reward for significant accuracy improvements. This competition drove innovation in the field, particularly in methods like **matrix factorization**, which became a standard technique for analysing and predicting user preferences.

Today, recommendation systems are more advanced and capable than ever before. They leverage sophisticated methods to handle large and complex datasets, making personalized suggestions that feel intuitive and helpful.

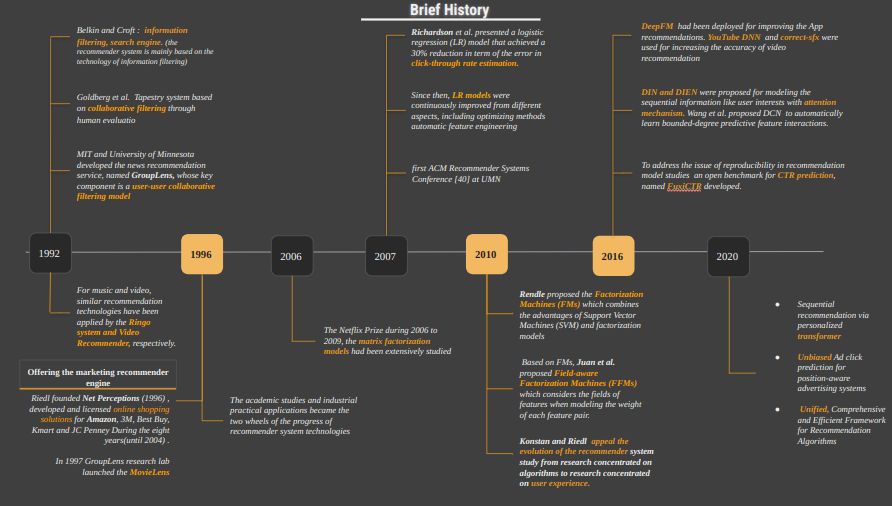


Figure 2.1

**2.2 Approaches in Recommendation Systems**

Modern recommendation systems use a variety of methods to provide personalized suggestions. Each approach has its strengths and is suited to different scenarios:

1. **Content-Based Filtering**:
   * This approach focuses on the attributes of the items themselves. For example, a user who watches a lot of action movies may receive recommendations for similar high-energy films.
   * Content-based systems are particularly useful for new users, as they rely on item metadata rather than extensive user interaction data. However, they can sometimes result in narrow recommendations, as the system may repeatedly suggest similar items.
2. **Collaborative Filtering**:
   * Collaborative filtering analyses patterns of user behaviour to make recommendations. For instance, if two users have similar viewing histories, the system might suggest shows or movies that one user has seen to the other.
   * This method is powerful because it can uncover connections between users and items that aren’t immediately obvious. However, it does require a substantial amount of data to perform well and can struggle with new users or items (a challenge known as the cold-start problem).
3. **Hybrid Models**:
   * Hybrid models combine content-based and collaborative filtering approaches, taking advantage of the strengths of both. For example, while collaborative filtering might identify what similar users are watching, content-based filtering can refine the recommendations by matching item attributes to the user’s preferences.
   * This combination creates more accurate and diverse suggestions, addressing some of the weaknesses found in using either method alone.

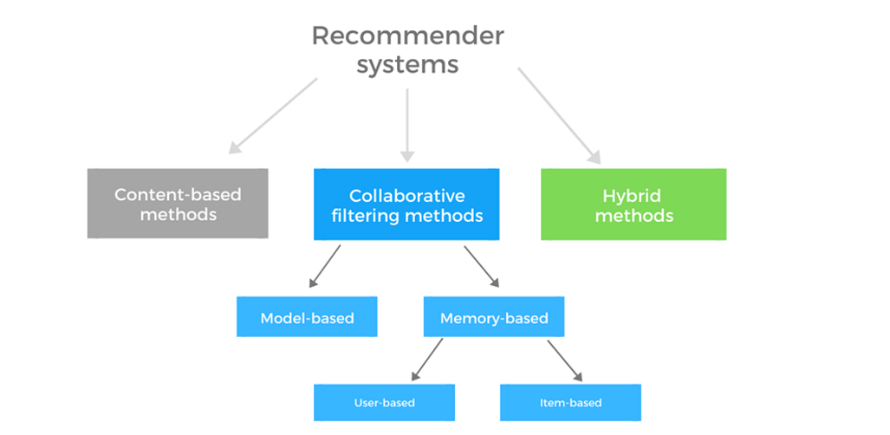


Figure 2.2

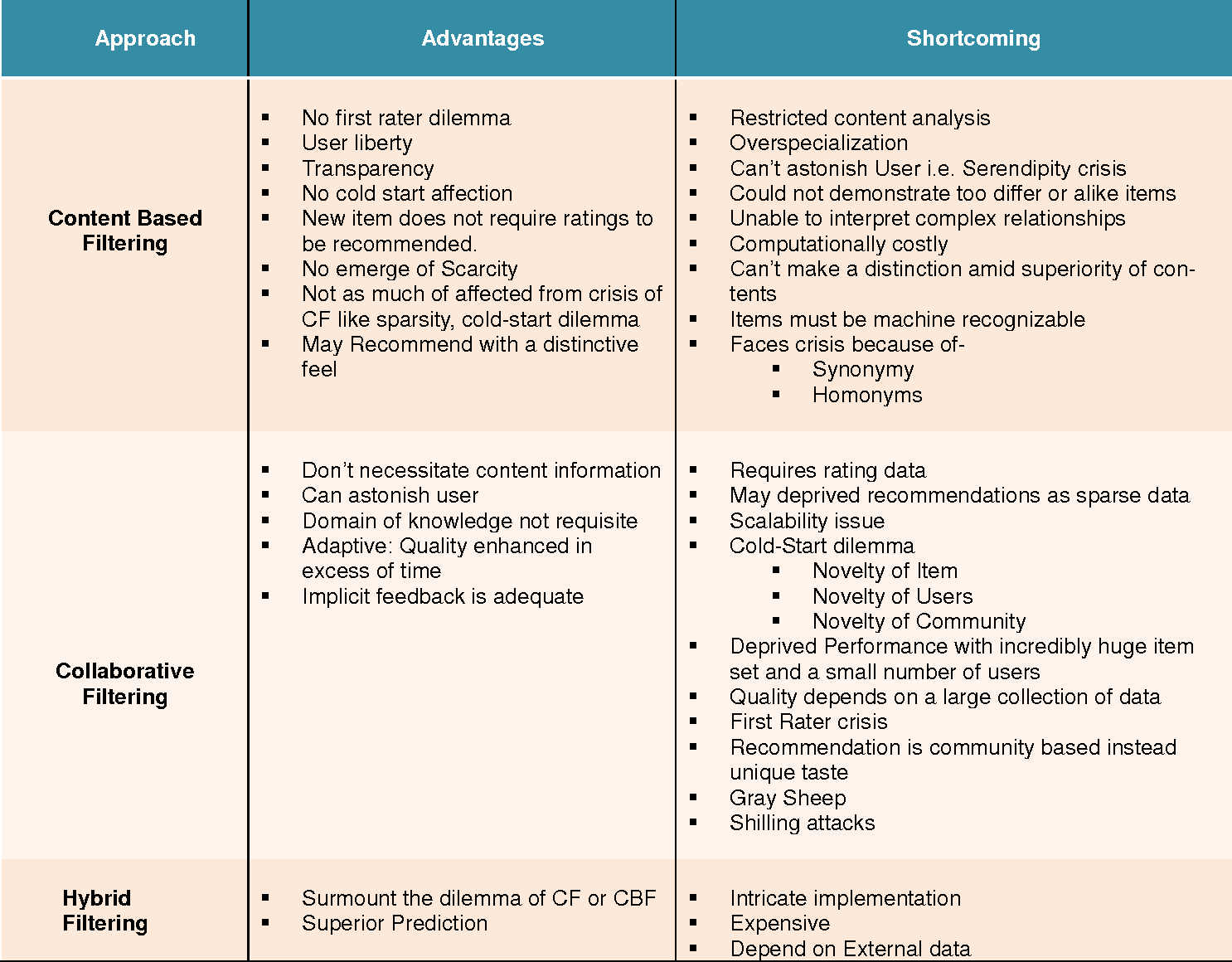


Figure 2.3

**2.2.1 Practical Applications**

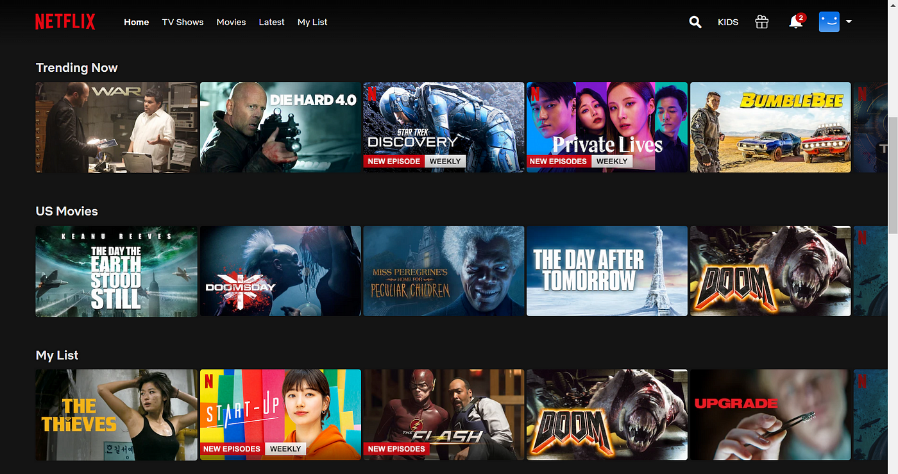
Today, recommendation systems are integral to a wide range of industries. In retail, they help customers discover new products; in entertainment, they guide users to movies, shows, or music they’ll enjoy; and in education, they suggest courses and materials tailored to individual needs. These systems simplify decision-making, save time, and make experiences more enjoyable for users.

Figure .4

**Chapter 3**

# METHODLOGY

The methodology outlines the structured approach adopted to develop the Movie Recommendation System. It involves three primary components: data preprocessing, feature engineering, and algorithm selection. These steps ensure the robustness, scalability, and accuracy of the recommendations.

**3.1 Data Preprocessing**

Data preprocessing is a crucial step in building any machine learning system, ensuring that the raw data is transformed into a suitable format for further analysis. The datasets used for this project were:

* tmdb\_5000\_movies.csv: Contains metadata such as titles, genres, budgets, and revenue.
* tmdb\_5000\_credits.csv: Provides details about cast and crew.

**3.1.1 Key preprocessing tasks included:**

1. **Handling Missing Values:**
   * Missing values in essential columns such as genres and cast were addressed by filling them with placeholders. Rows were removed only when critical information was unavailable, ensuring the integrity of the dataset.
2. **Data Cleaning:**
   * Text fields, such as movie overviews and taglines, were stripped of special characters, HTML tags, and extra spaces. This cleaning improved consistency and enhanced the quality of text-based features.
3. **Normalization:**
   * Numerical attributes, such as budget and revenue, varied significantly in scale. Min-Max normalization was applied to bring these values within a consistent range, thereby preventing bias in the recommendation algorithm.
4. **Merging Datasets:**
   * The two datasets were merged on the movie ID, creating a comprehensive dataset with both metadata and cast/crew details for each movie.

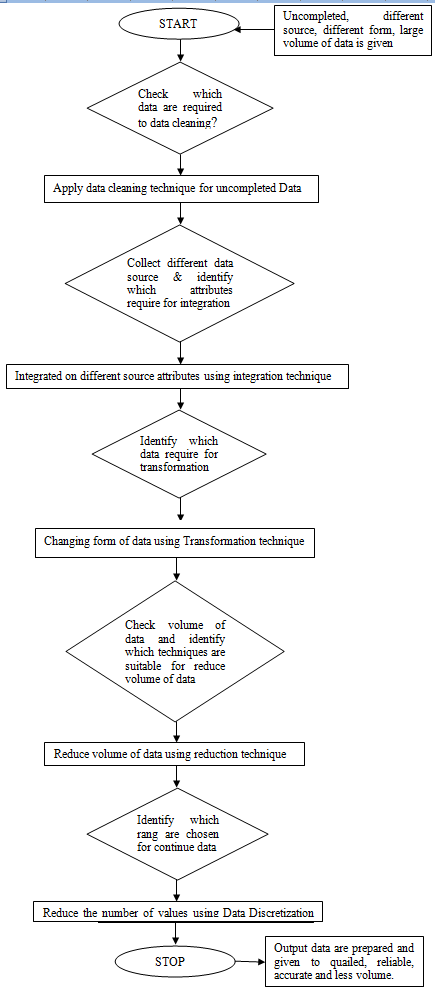


Figure 3.1

**3.2 Feature Engineering**

Feature engineering transforms raw data into meaningful inputs for algorithms. This project employed multiple strategies to enhance the quality of features:

1. **Feature Selection:**
   * Attributes like genres, cast, crew, runtime, budget, and revenue were identified as crucial for recommendations. These features were selected based on their potential influence on user preferences.
2. **Text Feature Extraction:**
   * Text-based features, such as movie overviews and taglines, were processed using Term Frequency-Inverse Document Frequency (TF-IDF). This technique extracted weighted keywords that captured the essence of each movie’s description.
3. **Numerical and Categorical Encoding:**
   * Numerical attributes like runtime and revenue were scaled to a uniform range.
   * Categorical features like genres were one-hot encoded to represent them in a machine-readable format.
4. **Unified Feature Vector:**
   * A unified feature vector was created by combining all selected attributes. For instance, genres, cast, and keywords were merged into a single vector for each movie, ensuring comprehensive representation.
5. **Weight Assignment:**
   * Attributes were assigned different weights based on their importance. For example, genres and keywords were given higher weights than runtime, reflecting their stronger influence on user preferences.
6. **Dimensionality Reduction:**
   * Principal Component Analysis (PCA) was applied to reduce the dimensionality of feature vectors. This step improved computational efficiency while retaining critical information.

**3.3 Algorithm Selection**

The recommendation system primarily utilized content-based filtering, leveraging cosine similarity to identify relationships between movies based on their feature vectors. This approach was chosen for its effectiveness in analysing structured data and providing personalized recommendations.

**3.3.1 Cosine Similarity:**

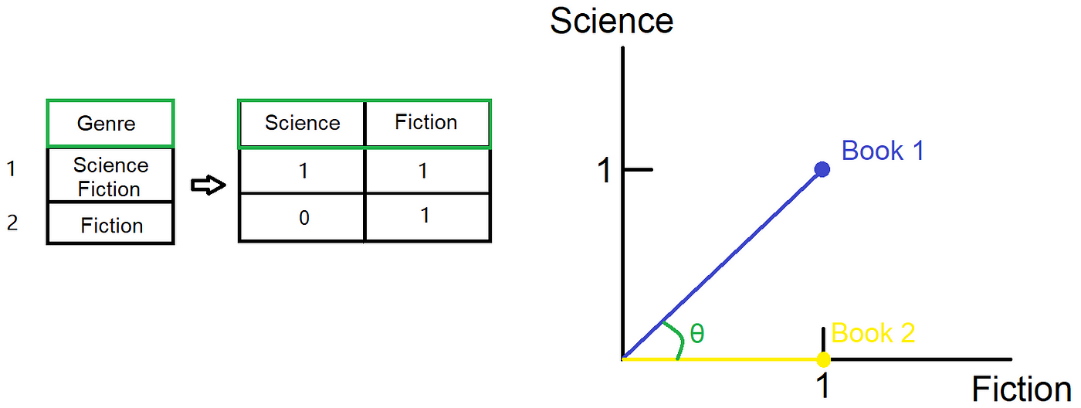
Cosine similarity is a popular method for measuring how similar two things are by looking at their direction in a multi-dimensional space. Imagine you're trying to figure out how similar two items are — like movies, books, or products. Cosine similarity helps by comparing them based on their features or characteristics.

Figure 3.2

The idea behind it is simple: it calculates the cosine of the angle between two vectors (which represent the items). If the angle is small, it means the items are very similar, and you'll get a score close to 1. If they’re completely different, the angle will be large, and the score will approach -1. If the items are totally unrelated, meaning they don’t overlap at all, the cosine similarity will be 0, indicating they’re orthogonal or unrelated.

This measure is particularly useful in recommendation systems, where the goal is to suggest items that are most similar to what a user has liked before, based on shared features. It helps ensure that the suggestions are relevant and closely related to the user's preferences.

**3.3.1.1 Mathematical Representation**

Given two feature vectors **A** and **B**, representing the attributes of two movies, cosine similarity (denoted as cos(θ)) is calculated as:

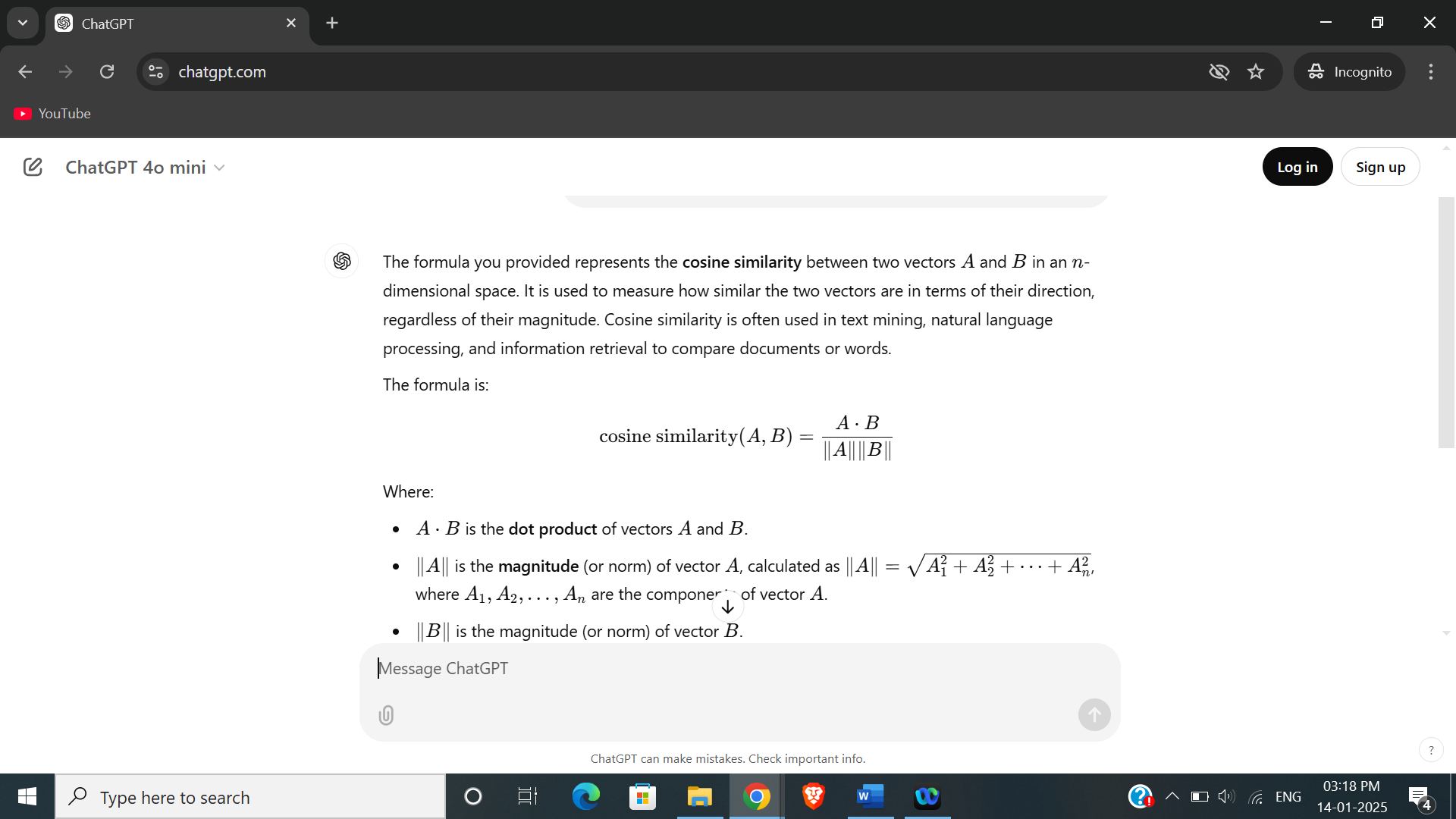


Figure 3.3

​

Where:

* A⋅B is the dot product of vectors **A** and **B**, which measures the overall magnitude of their similarity.
* ∥A∥ and ∥B∥ are the magnitudes (or Euclidean norms) of the vectors **A** and **B**. The magnitude of a vector **A** is calculated as ||A|| = **√ (A12 + A22 + A32 +…..+ An2 )**​ where An and Bn are the components of the vector.
* The cosine similarity essentially normalizes the dot product by dividing it by the product of the magnitudes of the vectors, which eliminates the influence of vector length, making it purely based on the direction or angle between the vectors.

**3.3.1.2 Conceptual Explanation for Movie Recommendations**

In the context of a movie recommendation system, the vectors represent the feature attributes of movies, such as genre, overview, director, and cast. Each movie is represented as a high-dimensional vector, where each dimension corresponds to a specific characteristic (e.g., action, romance, or rating).

For instance, Movie A might have a vector representation that emphasizes action (0.8), thriller (0.6), and a high user rating (0.9), while Movie B might have a vector emphasizing similar dimensions but in different proportions, such as action (0.7), thriller (0.7), and user rating (0.85). By calculating the cosine similarity between these two vectors, the recommendation system can quantify how similar these two movies are in terms of their features.

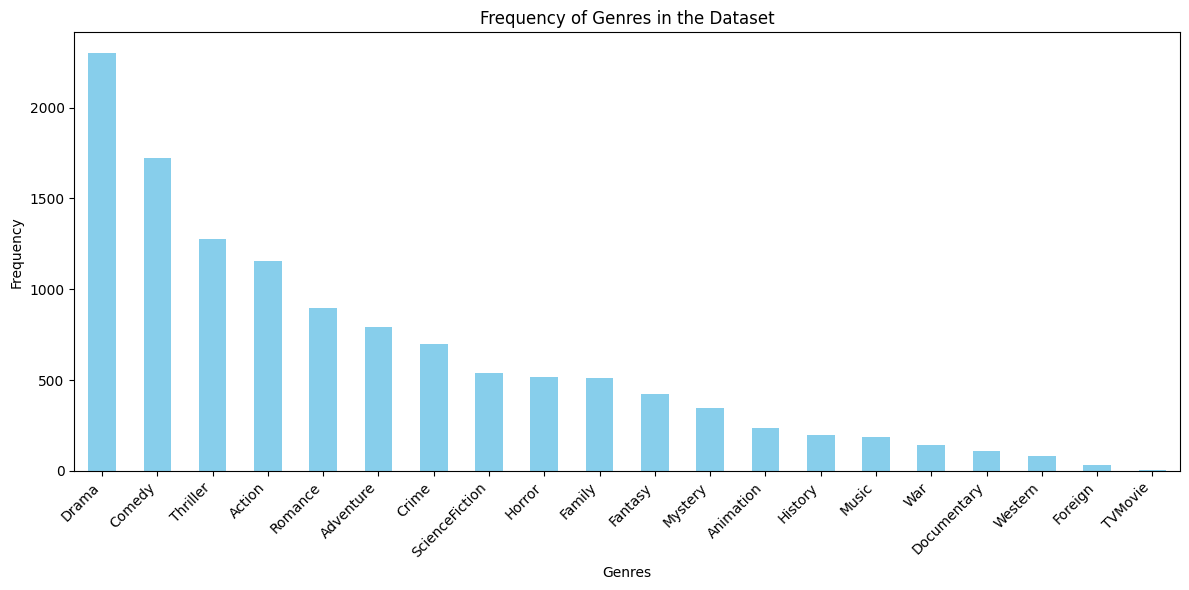


Figure 3.4

**Example**

Imagine a user has watched and rated a movie highly that belongs to the action and thriller genres. The recommendation system computes the cosine similarity between this movie and a large database of other movies. Movies with vectors that are closer in direction to the vector of the highly-rated movie (i.e., those that share similar genre and theme features) will have a higher cosine similarity score. These movies are then recommended to the user, as they are deemed to match the user's tastes based on the features of the movies they have rated highly.

**3.3.1.3 Why It Works in Recommendation Systems**

Cosine similarity is a powerful tool in recommendation systems, particularly in collaborative filtering and content-based approaches. Here’s why it shines, even when data is sparse—like when a user has only rated a few movies.

1. **Focus on Direction, Not Magnitude**: Unlike Euclidean distance, which considers how far apart two points are, cosine similarity looks at the angle between vectors. This means it’s better for comparing items that might have different lengths. For instance, if one movie has lots of features and another has just a few, cosine similarity helps us understand how similar they are based on their content, rather than getting hung up on their length.
2. **Capturing Preferences**: By examining the angles between vectors, cosine similarity effectively identifies how closely related two items are regarding user preferences. This is crucial for recommending movies that align well with a user's tastes—helping the system suggest films that users are likely to enjoy based on shared characteristics.
3. **Normalization**: Cosine similarity normalizes vectors, which means it looks at items relative to one another. This approach helps avoid biases that might occur when comparing items with different feature sets, ensuring that every movie gets a fair assessment.

**3.3.1.4 Practical Use in Recommender Systems**

* **Content-Based Filtering**: When a user rates a movie highly, the system can find other films with similar attributes—like genre, themes, or directors—by calculating cosine similarity. This helps users discover new favourites that match their interests.
* **Collaborative Filtering**: In user-based systems, cosine similarity measures how closely users' rating patterns align. By identifying users with similar tastes, the system can recommend movies that those users have enjoyed, tailoring suggestions based on collective preferences.

Overall, cosine similarity offers a straightforward and efficient way to gauge the similarity between movies (or other items). This capability enables recommendation systems to provide personalized suggestions, enhancing the user experience by connecting them with films they’re likely to love based on their past behaviour and preferences.

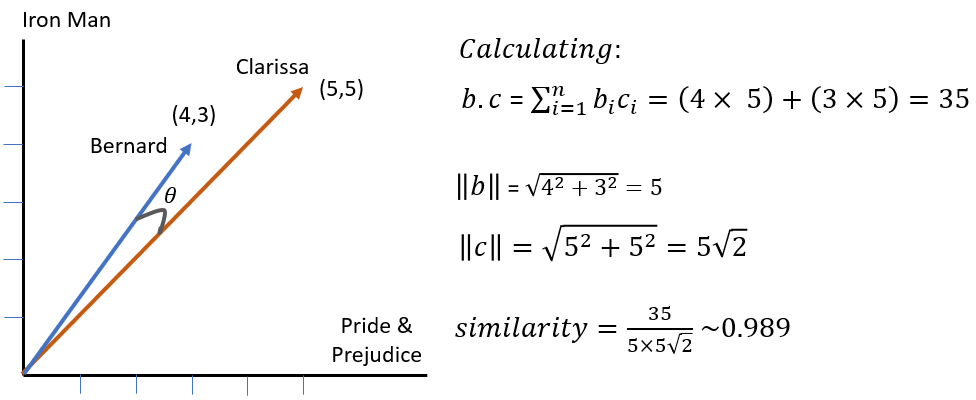


Figure 3.5

**Chapter 4**

# RESULT & DISCUSSION

In this section, we summarize the outcomes of the Movie Recommendation System, focusing on how well it worked, user feedback, challenges faced, and how the choice of cosine similarity contributed to the system's effectiveness.

**4.1 Effectiveness of the Recommendation System**

**Relevance of Recommendations:**

The system did a great job of recommending movies that matched users' tastes. By using features like genre, overview, cast, and crew, it suggested films similar to the ones users had enjoyed in the past. For example, if a user liked action and thriller movies, the system recommended more films within those genres, which kept the suggestions relevant and accurate.

**Handling the Cold-Start Problem:**

The cold-start problem, where new users or movies struggle to get good recommendations due to limited data, was well-handled. Since the system relies on movie attributes (like genre and cast) instead of user ratings, even new movies and new users received quality suggestions right from the start.

**Diversity of Recommendations:**

While the system was great at recommending similar movies, it sometimes focused a little too much on a narrow range of genres or themes, especially if a user liked a very specific type of movie.

**4.2 Challenges Encountered**

**Data Quality and Missing Information:**

A key challenge was dealing with incomplete or missing data, especially in critical areas like movie genre or cast. While missing values were handled by using placeholders or removing rows with essential missing information, this occasionally meant some good movies were excluded from the recommendations.

**Feature Selection and Representation:**

Choosing the right features to represent each movie was another challenge. Attributes like genre, cast, and crew were straightforward, but text-based features, like movie overviews, required additional processing to extract useful information. Ensuring that these features accurately represented the movies was essential to making good recommendations.

**4.3 Discussion on the Use of Cosine Similarity**

**How Cosine Similarity Worked:**

Cosine similarity was an excellent choice for comparing movies based on their feature vectors. It measured how similar two movies were by looking at the angle between their feature vectors, which normalized for differences in the number of features, allowing the system to compare movies accurately even when they had varying amounts of data.

**Advantages:**  
Cosine similarity’s biggest advantage was its simplicity and efficiency. It worked well for movies with different feature sets, ensuring that even if one movie had many attributes and another had only a few, they could still be compared meaningfully. Its straightforward computation also made it well-suited for large datasets.

**Limitations:**  
The main drawback of cosine similarity is that it doesn’t take into account user-specific preferences. It focuses only on movie features, so it might miss nuances in a user’s personal taste. For example, two movies might share similar features but appeal to different users based on their individual preferences. Incorporating user behaviour through a hybrid model, combining both content-based and collaborative filtering, could help overcome this limitation and provide more personalized recommendations.

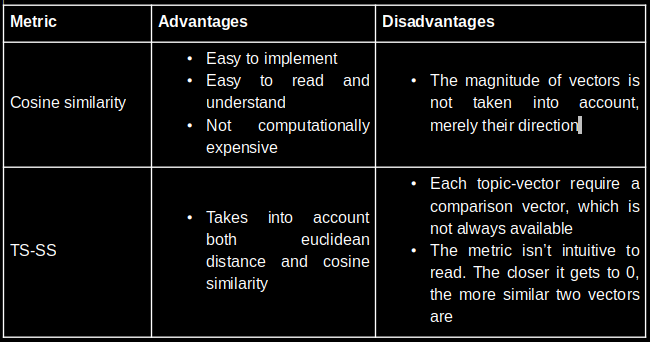


Figure 4.1

**Chapter 5**

# CONCLUSION & FUTURE IMPLEMENTATION

**5.1 Conclusion**

The Movie Recommendation System created for this project effectively helped users discover movies they’d love by offering personalized and relevant suggestions. Using content-based filtering and cosine similarity, the system analysed movie features like genre, overview, cast and crew to match users with films that aligned with their preferences. It also successfully overcame the cold-start problem, making it useful for both new users and lesser-known movies.

While the system performed well overall, there were some challenges, such as handling missing data, choosing the right features, and optimizing performance. These obstacles provided valuable lessons and pointed to areas where the system could be further refined.

**5.2 Future Implementation**

Looking ahead, there are several ways to make the Movie Recommendation System even better:

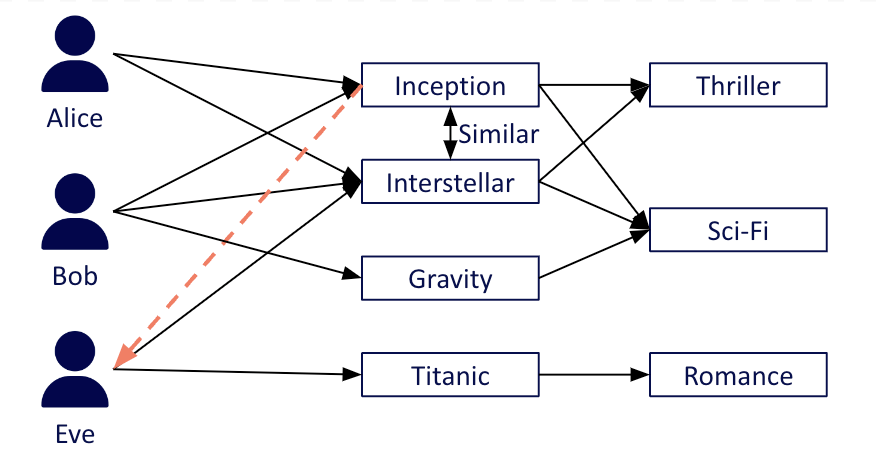
1. **Hybrid Recommendation Approach**: Combining content-based filtering with collaborative filtering would bring the best of both worlds. This would allow the system to not only consider movie features but also user preferences, resulting in more accurate and diverse recommendations.

Figure 5.1

1. **Incorporating User Behaviour**:

Including more personalized data, like a user’s viewing history and ratings, would make the system smarter. By learning from a user's past choices, the system could suggest movies that are even more tailored to their unique tastes.

1. **Improved Data Handling**: To overcome issues like missing data, more advanced techniques could be used to fill in gaps or work with larger, more complete datasets. This would ensure that all relevant movies are considered for recommendations.
2. **Scalability and Performance Optimization**: As the system grows, it could slow down when calculating similarities. To keep it fast and efficient, using methods like approximate nearest neighbour search or more advanced dimensionality reduction techniques would help speed things up and ensure scalability.
3. **User Customization**: Giving users more control over their recommendations, like the ability to filter by genre, director, or release year, would make the system more flexible. This could appeal to a wider range of users, whether they want to explore niche content or discover the latest releases.

By making these improvements, the Movie Recommendation System can become even more accurate, efficient, and enjoyable to use. These updates would ensure that the system remains relevant and useful to a broader audience, paving the way for future developments across different platforms.

# REFRENCES

1. **Datasets**:

* The Movie Database (TMDb). (n.d.). TMDb 5000 Movie Dataset. Retrieved from <https://www.kaggle.com/tmdb/tmdb-movie-metadata>

1. **Research Papers**:

* Kumar, M., Yadav, D. K., Singh, A., & Gupta, V. K. (2015). A Movie Recommender System: MOVREC. *International Journal of Computer Applications, 124*(3). DOI: <https://www.researchgate.net/publication/283042228_A_Movie_Recommender_System_MOVREC>

1. **Websites**:

* Netflix Technology Blog. (2017). The Netflix Recommender System: Algorithms, Business Value, and Innovation. Retrieved from <https://medium.com/netflix-techblog>