

**Project title:** **Movie Recommendation System using Cosine Similarity**

**Group members:**

**Bilal Ali 21K-3153**

**Ehaab Tariq 21K-3372**

**Huzaifa Ahmed 21K-3403**

**Arham Mehmood 21K-3201**

**Abdur Rehman 21K-4529**

**April 20, 2024**

**Background and Motivation**

The constant and rapid production of digital streaming platforms has drastically changed the means of how users consume content. But the overwhelming amount of content available across the internet confuses the users which in turn leads to choice paralysis out of fear of a bad decision. To overcome this issue, movie recommendation systems were introduced, which aims to assist users in discovering new content that aligns with their preferences. But for the Movie Recommendation system to be relevant it needs to recommend the right movies at the right time.

Movie recommendation systems themselves can be implemented in numerous ways and help in enhancing the user experience as well as increasing retention on streaming services. By providing personalized content tailored to each user, a sense of comfort and connection is built between users and the service. Moreover, personalized recommendations can lead to increased user engagement, longer session durations, and higher retention rates, ultimately benefiting both users and streaming platforms alike.

However, not all recommendation systems are implemented the same way. Some take a generic, broad-spectrum approach and only recommend currently trending movies, while others focus on providing niche recommendations to give the illusion of specificity. No system truly considers each user as a separate entity, instead relying on groups and cliques to provide recommendations in a broad sense. This is where our project, and the use of cosine similarity, comes in.

The project makes each user an individual, separate entity by assigning them an personal “Watchlist” in which users can enter movies they have already seen. Once that is done, each “Watchlist” is analyzed independently and compared to all movies in our vast database. This yields the most common movies between the users “Watchlist” and the Movie database, effectively functioning as a fully personalized recommendation system.

By alleviating decision fatigue and analysis paralysis, our system rethinks how users watch and find new movies in this digital, AI-led era.

This paper explores other formulas behind recommendation systems, details the formula behind our system, Cosine Similarity, outlines its working and overall benefits.

**1. Background of existing recommendation systems**

**1.1. Collaborative Filtering**

Collaborative filtering (CF) lies at the core of many movie recommendation systems. This approach analyzes user-item interactions to identify patterns and preferences, which allows the system to provide recommendations based on the patterns of similar users.[3] Early CF models primarily relied on explicit user ratings, reviews and comments to generate recommendations. These recommendations were generated by implementing large and vast user-item matrices with many empty cells, which not only grew computational costs but also yielded sub-optimal results in platforms with diverse content and user preferences, as the matrices grew too large yet still stayed sparse. Moreover, new users could also face the “cold-start” problem, as they could not receive interactions without rating a few movies themselves first.

**1.2. Content-Based Filtering**

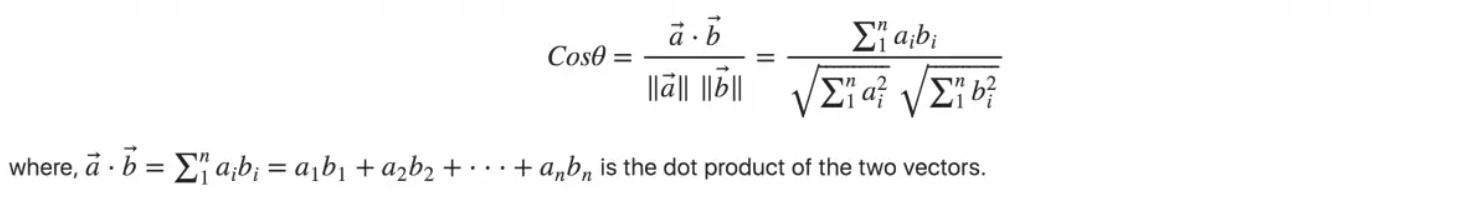
To address the limitations of collaborative filtering, content-based filtering (CBF) was developed to overcome the drawbacks of CF. Unlike CF, which relies solely on user-item interactions, CBF incorporates metadata and item attributes such as genre, director, and cast into the recommendation process [2]. This approach pioneered recommendation systems from then on, and is a stepping stone for many modern recommendation systems, including our Cosine Similarity formula. By not just focusing on user ratings but also the characteristics of movies such as genre and cast, CBF offers a more personalized and nuanced approach to recommendation. Moreover, content-based methods mitigate the cold-start problem encountered by new users or items, as they can generate recommendations based on item features alone, without requiring historical user interactions.

**1.3. Hybrid Recommendation Systems: Integrating Multiple Approaches**

Researchers have created hybrid recommendation systems that blend collaborative and content-based approaches to maximize user satisfaction. By combining the strengths of collaborative filtering in capturing user preferences and the richness of content-based filtering, these systems offer more precise recommendations Techniques like weighted combination, feature augmentation, and cascade models utilize the best features of both approaches, resulting in enhanced accuracy in recommendations tailored to each user’s preferences.

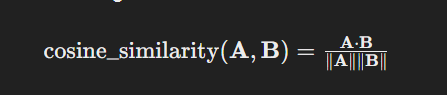
**2. The Math behind Cosine Similarity**

In the simplest sense, cosine similarity measures the similarity between two vectors of an inner product space [1]. This means that given 2 vectors, the cosine similarity formula computes the cosine of the angles between these 2 vectors. The shorter the angle, the more similar the vectors are. It determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis. However, we will instead use it to compute the similarity between movies in order to provide highly specific and personalized recommendations.



As mentioned before, the formula measures the cosine of the angle between two vectors and ranges from -1 to 1. A value of 1 indicates that the vectors are identical, 0 indicates that the vectors are orthogonal (i.e., they have no correlation), and -1 indicates that the vectors are exactly opposite.

A simplified version of the above formula is:



Where:

- **A.B** represents the dot product of the two vectors.

- **||A||** and **||B||** represent the magnitudes (or lengths) of vectors A and B respectively.

**3. How Cosine Similarity is implemented in a Movie Recommendation System**

Before observing the implementation of this formula, let us outline our data. The system has a vast library of movies, each characterized by multiple categories: Actors, Director, Year Released, Reviews, IMDB Rating and Genre. The Genre category is the most important, as it is responsible for the output of our formula.

We can represent each movie as a vector where each dimension corresponds to a feature of the movie (e.g., genre, director, actor). The cosine similarity between two movie vectors then indicates how similar the movies are based on these features. Higher cosine similarity values suggest greater similarity between the movies, making it a useful metric for recommending similar movies to users. This logic is what we will use as the backbone of our recommendation system and deliver users personalized recommendations.

A user will login to the system and add movies that they are fond of or have seen before to their personalized “Watchlist”, unique for every user. Immediately, we run into the “cold-start” problem similar to Collaborative Filtering, where if a user has not seen a movie before, they would be unable to receive recommendations.

To overcome this, we use a technique from the Content Based Filtering Model. Since our movies are not only rated by current users but also have attributes such as Genre, IMDB Score and Actors, we can suggest the most popular movies of the current time to a fresh user. A fresh user who has never seen a movie before would be recommended the top movies of the year in each genre with the most popular actors of the current era.

Once a user has added their favorite movies to the Watchlist, our recommendation system can start delivering personalized recommendations.

All the movies currently in the users Watchlist would be converted into a base vector. To achieve this, the system takes the most defining characteristic of any movie, its Genre, and turns the movie into a binary vector representing the Genre attribute. Each element in the vector would be 1 if the movie belongs to that genre and 0 otherwise.

A simplified example for a movie with the genres “Action, Comedy”: This movie would have 1 in the Action and Comedy categories and 0 in other categories such as Romance or Tragedy.

Thus, all movies in the Watchlist will be converted into binary vectors based on their Genre. After that, the movies NOT in a user’s watchlist will be converted into similar binary vectors. Finally, a Cosine Similarity formula will be applied to movies in “Watchlist” and movies not in “Watchlist”. The most similar movies, or the vectors with the shortest angles will be returned.

This means that users will receive recommendations for movies whose genres are the most similar to the genres of movies they have already seen and liked, ensuring that they are provided with movies that suit their tastes.

**4. Advantages over other models**

**4.1 Advantage over Collaborative Filtering:**

Using a pure cosine similarity approach with movie vectors derived from users' watchlists can offer several advantages over traditional collaborative filtering (CF) methods:

1. **Sparse Data Handling**: As mentioned before, user-item matrices in CF can be very sparse.

This sparsity of data increases in databases consisting of many users and movies, resulting in long lasting computation times. So, using high-dimensional vectors representing movies, this sparsity can be reduced, as even users with not much Watchtime could still have meaningful vector representations.

2. **Scalability**: In CF, with the increase in the number of users and items, the computation time for locating similar users/items increases drastically. While scalability of calculation of cosine similarity between movie vectors is better, especially when using efficient algorithms and data structures.

3. **Cold Start Problem Mitigation**: CF often struggles with the cold start problem, where providing recommendation in limited interaction history is challenging. Moreover, CF does not have baseline categories like genre, actors etc to base its results on. But cosine similarity of movie vectors can still be calculated even if a user does not have a Watchlist by representing movies as vectors on the bases of (e.g., genre, cast, plot keywords). This alleviates the cold start problem.

4. **Fine-Grained Recommendations**: Movie vectors can capture characteristic similarities between movies beyond simple user ratings. For example, two movies might not have been rated similarly by users but could still share similar thematic/genre elements, styles, or narrative structures, which can be captured more effectively through vector representations.

**4.2 Advantages over Content Based Filtering:**

Let's explore how a pure cosine similarity approach can be advantageous compared to traditional content-based filtering (CBF) methods:

1. **Scalability**: Cosine similarity is computed directly on movie vectors while traditional CBF methods utilize feature extraction and processing. Computing cosine similarity between vectors is often faster than feature extraction, especially in recommendation systems with vast datasets.

2. **Handling Multimodal Data:** In scenarios where movies have diverse content types, such as text descriptions, images, audio, or metadata, directly applying cosine similarity to vectors can handle multimodal data more effectively. Traditional CBF methods struggle to analyze different content types effectively, while cosine similarity treats each type equally once it has been turned into a vector.

These benefits outline how cosine similarity is a useful recommendation system for their scalability, ability to keep computational costs low and efficient handling of diverse content types.

**5. Challenges and Future Directions**

Despite the advantages of the Cosine Similarity model, several challenges still exist. The formula is sensitive to the length of the vector. If one vector is significantly larger than the other, it can dominate computation and lead to a skewed result. Luckily, since movie genres are mostly static, this issue is only sometimes relevant, but should still be fixed in future iterations.

There is also the phenomenon of “excessive similarity”. In user Watchlists with an overwhelming number of similar movies, the algorithm will not branch out into new genres and keep recommending similar movies. To overcome this, the model should be merged with deep learning algorithms to recommend both similar and dissimilar movies for a healthy balance.

**6. Conclusion**

In conclusion, movie recommendation systems play a crucial role in enhancing user experience and engagement on streaming platforms. Collaborative filtering, content-based filtering and hybrid approaches have been used as models for recommendation systems. However, they suffer from issues such as cold-starts, high computation times and sparsity. Our proposed Cosine Similarity model makes binary vectors out of movies the user has already seen, and then compares how similar those vectors are to other movies in the database. The most similar movies are recommended to the user. Through this approach, sparsity is significantly reduced, leading to lower computational costs. Through including IMDB ratings as a baseline for recommendations, the cold-start problem is also eliminated. However, vector length sensitivity and excessive similarity remain issues. Nevertheless, the Cosine Similarity model is by far the most efficient model for a movie recommendation system, guaranteeing specific and personalized movie recommendations.

**References**

*,*. (2024, March 5). , - YouTube. Retrieved April 20, 2024, from https://www.sciencedirect.com/topics/computer-science/cosine-similarity

*A Guide to Content-based Filtering in Recommender Systems*. (n.d.). Turing. Retrieved April 20, 2024, from https://www.turing.com/kb/content-based-filtering-in-recommender-systems

Jain, S. (2023, March 31). *Collaborative Filtering in Machine Learning*. GeeksforGeeks. Retrieved April 20, 2024, from https://www.geeksforgeeks.org/collaborative-filtering-ml/