A Content-based Movie Recommendation System

Yiting Yuan[1,*], Youyang Qin[2], Zekai Yu[3], Congbai Zhang[4]

[1]Yiting Yuan, Department of business, Rutgers University, Camden, NJ, 08102, United States
[2]Youyang Qin, Abington Friends School, Jenkintown, PA, 19046, United States
[3]Zekai Yu, Department of Statistics and Data Science, University of Washington, Seattle, 98105, United States
[4]Congbai Zhang, Barry Florescue Undergraduate Business Program, University of Rochester, Rochester, NY, 14627, United States

yy573@scarletmail.rutgers.edu

These authors contributed equally to this work and should be considered co-first authors.

Abstract. In this paper, we describe a content-based movie recommendation system and provide an overview of the movie recommendation systems in today's market. Our findings show 1): Summary-Based and Feature-Based movie recommendation systems will provide different recommendation results. 2) Combined recommendation system's result is consistent with the Summary-Based recommendation system but different from the Feature-Based recommendation system. Based on our recommendation system, we also made some innovations and fusion and conducted several control tests to improve the quality of our recommendations.

1. Introduction

It is undeniable that movie recommendation nowadays plays an important factor in a lot of industries, making considerable profits. When it comes to the movie industry, streaming platforms are definitely one that comes into people's minds. Netflix, Hulu, and HBO are the most successful streaming platforms nowadays, and their recommendation system has contributed a large amount of their incomes. Nearly 35 to 40 percent of HBO's revenues come directly from the movie recommendation system (Jena et al. 2022). Personalized Customer service is one of the reasons that the recommendation system is so critical to these platforms.

Another booming industry that involves recommendation algorithms is short video platforms like TikTok. They have large Media Users that make videos of movie breakdowns or reviews that earn a huge viewing flow for the platforms. TikTok only needs to come up with an appropriate recommendation system that sends these videos to those who are interested.

These companies are our inspirations to come up with a recommendation system that relies on the content of the movies. What we want to do is to recommend these movies based on their features and descriptions, so even when we don't have the user information or feedback, we could still run a decent recommendation system. To achieve this goal all we need is the data set that includes descriptions of the movies and other content like directors, actors, and genre. By vectorizing the features and the

adescription, we could derive the item-to-item similarities and all of the recommendations will be based on that.

Uluyagmur et al. (2012) figured out the way to combine the features of movies watched by the user as keywords, and by weighting these keywords, they are able to find the best matches of movies for the user as recommendations. Their approach is very convenient and reasonable. However, without the given ratings of the user on these watched movies, this method is hard to get started, which is usually called the "cold start" problem, since they decided to use the rating as a weighting factor in the algorithm.

Singh et al. (2020) introduced the idea of vectorization. They vectorized each movie by the plot descriptions of the movies, and by calculating the similarities chart, they are able to recommend movies to the user. This is a pure content-based filtering approach, though it solved the cold start problem since data sets of movies are relatively easier to obtain. It seems not to be fully developed yet, what if the user doesn't want their recommendation system to just fully rely on the descriptions.

Our methodology inherits the idea of vectorization, but it is more well-rounded than the description-based recommendation. Now it is an option for users to choose their recommendation to be based on descriptions or movie features. Meanwhile, this method avoided the problem of cold start, since it doesn't require any ratings or user information other than watched movies as input.

2. Related Work

2.1. Recommendation System

Note: *An information processing filtering system known as a recommendation system simply establishes links between pieces of data by first predicting user ratings and preferences, followed by comparison or correlation. We may undoubtedly save a lot of time by using recommendation systems to quickly identify our preferred material among the current deluge of information.*

Sharma and Gera's (2013) paper is based on the study and analysis of the three major mainstream recommendation systems: collaborative, content-based, and hybrid. The paper describes these systems in detail and explores the shortcomings of the current algorithms, such as data sparsity, cold start, scalability, and over-specialization problems. Then, focusing on the above problems will be the core of recommendation system improvement.

Ahmed et al. (2018) proposed a movie recommendation system that separates users using a clustering algorithm in order to find users with similar tastes in movies. Machine learning approaches are used to guess what rating a particular user might give to a particular movie so that this information can be used to recommend movies to viewers. Principal Component Analysis (PCA) was used before clustering to reduce the dimensions from 18 to 6. It uses the k means clustering algorithm while rating gives the best result. A pattern recognition network helps to build the network since there are more than two but a limited number of target classes.

2.2. Content-based filtering

Note: Using the information about directors, performers, and tags for movies, the characteristic features of related items are mostly used to filter the content-based recommendation engine. Simply said, it involves making user recommendations based on how similar the product or service is to others.

Uluyagmur et al. (2012) designed a content-based movie recommendation system that uses different sets of features, such as "keywords for actors, directors, genres, etc.". A different weighting of the movie features is computed by capturing the movies that users have watched in the past and the ratings they have left for the movie. When weighing the features for each user, special attention should be paid to movies that the user has watched entirely or mostly, and the features extracted from such movies are significant. Based on the different features of the movies, the ratings of each feature set are computed and thus compared, and then the best matching movie recommendation is made to the user.

In this study, Lops et al. (2019) review the trends of content-based recommendation systems, where collaborative and content-based filtering systems, respectively, were the main approaches in the past. In the past, the volume of user preferences served as a barrier for content-based recommendation systems.

Nevertheless, content-based filtering has become a more successful method for making suggestions as a result of more advanced machine learning models mixed with certain supplementary information (movie features). This has been taken into account when forecasting future recommendation systems.
Singh et al. (2020) defined content-based filtering, collaborative filtering, and hybrid filtering and discussed the drawbacks of each algorithm such as the cold start problem, data sparsity problem, and scalability. It uses content-based filtering and cosine similarity to predict the result. After calculating the cosine similarity, it uses a normalized popular score through which it gets the function of computing distance. The angle theta between the two movies will determine the similarity between the two movies. Then by using the k-means nearest neighbor (KNN) functionality, the algorithm has found the nearest neighbor to recommend to the user.

2.3. Collaborating filtering

Note: *A collaborative filtering recommendation system leverages the user's past preference information to compute the distance between users and then recommends products or services to users who have diverse preferences but are geographically close to one another.*

Tewari et al. (2018) designed a recommendation system that generates smaller top-n item recommendations lists by placing users' unseen items in the recommendation list and thus attaining high precision value. The system uses five unique building blocks to generate recommendations. The Profile Builder block constructs the profile of every user and item and creates the profile of each user in the form Keywords Vector (KV). Similarity Finder collects users' keyword vectors from the Profile Builder block. The collaborative classifier block uses collaborative filtering to generate recommendations. The Item Weight and Variance block help in finding the popularity of different items among all users in the form of weights. And the final results are generated by the Final Recommender block.

A movie recommendation system was developed by Wu et al. (2018). This system takes users' previous ratings of movies and then makes recommendations to users based on those ratings. In order to put the system into operation, they utilized a collaborative filtering algorithm in conjunction with the Apache Mahout framework, which is a scientific and highly analytical library based on the Apache Software Foundation. They exploited similarities between users and personality associations between users as a means of filtering users. They locate the data sets that have the maximum similarity speed and merge them by using an algorithm called UserNeighborhood, which is a distance-based clustering machine learning technique.

Keshava et al. (2020) improved the CineMatch algorithm by 10 percent by using fashionable collaborative filtering techniques. It will first need an item-user matrix that represents the movies. Then derive the user-user similarities matrices and item-item similarities matrices from the item-user matrix. They implement a model to predict the ratings that a user will possibly give to the movie called Surprise Model, which will use the previously derived matrices and also minimize the difference between the prediction and the actual score by looking at the implicit feedback of customers, like purchase history, surfing history, seek patterns or even mouse movements.

2.4. Hybrid filtering

Note: *The hybrid filtering system is essentially the integration and improvement of several methodologies (such as content-based filtering and collaborating filtering) to overcome the limitations and issues produced by a single suggestion technology and boost the overall performance of the recommendation system. Single-algorithm recommendation systems are less precise and less practical than hybrid recommendation systems.*

Uddin et al. (2009) developed a hybrid recommender system that was optimized based on both content and collaborative filtering. This was done in order to minimize the constraints that are associated with each method. They developed a program known as Enhanced Content-based Filtering Using Diverse Collaborative Prediction (ECBDP), which takes movie feature sets and user ratings as its two primary forms of data input in order to input and compute similarities between different users. The

multivariate collaborative prediction is then used by the system to locate the items of interest to the users as well as the items that are most comparable for the recommendation.
The approach studied by Afoudi et al. (2021) improves the efficiency of the traditional systematic filtering system approach by creating a hybrid recommender system. They created a new framework that combines traditional collaborative filtering with a content-based filtering approach. The hybrid model comprises four main components: collaborative filtering, content-based filtering, SOM collaborative filtering, and hybrid filtering. Filtering is performed by combining movie and user features and using the SOM-CF model for rating supervision. By combining the scores of all models, the strengths of each model are obtained to improve the precision and accuracy of the overall approach.

3. Dataset

The dataset (Mubarak, 2021) used to build the movie recommendation system consists of movies and series featured on Netflix in 2021 and before. The dataset is based on 100,000 user records for developing recommendation systems. There are in total 9,425 observations in this dataset, and each observation is a unique movie. Each observation is described by 28 different variables, of which we used 6 fields to build our movie recommendation system (Table 1). The variables can be categorized into four groups. The first category of variables is the inherent attributes of the media: genre, tags, category rating (PG-14/R), runtime, and a summary of the media. The second category is cast information, which includes the director, writers, actors/actresses, and Production House. The third category can be described as feedback or evaluation from audiences and the market. To name a few examples, it has audience ratings from various websites like IMDb and Rotten Tomatoes, box office, and nomination and award records. The last category is peripheral to the media itself: release dates and links related to the media.

Table 1. Field Summary.

Field	Description	Variable Category
Title	the name of the movie	Inherent Attribute
Genre	the cateogry of movie characterized by a particular style	Inherent Attribute
Tags	descriptive phrases that capture features of a movie	Inherent Attribute
Director	the director(s) of the movie	Cast Information
Actors	the actor(s)/actress(es) of the movie	Cast Information
Summary	an abstract of the plot of the movie	Cast Information

Note: This table shows the 6 fields we used from the Kaggle dataset along with their descriptions for the 100,000 user records and 9.425 unique movies

Since our goal is to build a content-based movie recommendation system, we will only use variables in the first two categories (Inherent Attribute and Cast Information). All the above variables are in the form of natural language. The title will be the input and output of our recommendations system. Genre, tags, directors, and actors consist of words or phrases separated by commas. Summary is a complete sentence that describes the plot of a movie.

4. Methodology

4.1. Data Preprocessing

To prepare the dataset for our model, we went through steps (figure 1) to clean up the original dataset. In the process, we filter out observations according to certain criteria and only keep variables that we assume will contribute the most to differentiating movies.

First, we use the “movie or series” and IMDb rating score as filters to drop the observations that are not movies (i.e., we dropped all series in the dataset) and received a rating that is lower than 3.0. Then, we dropped all the variables that are irrelevant to our predictive analysis. The variables that will be

included in the model are listed in Table 1 in the dataset section. Lastly, we dropped all the observations with missing values so that no blank cells will be fed into the vectorizer. After completing the above steps, we have 6747 observations left in the data frame.

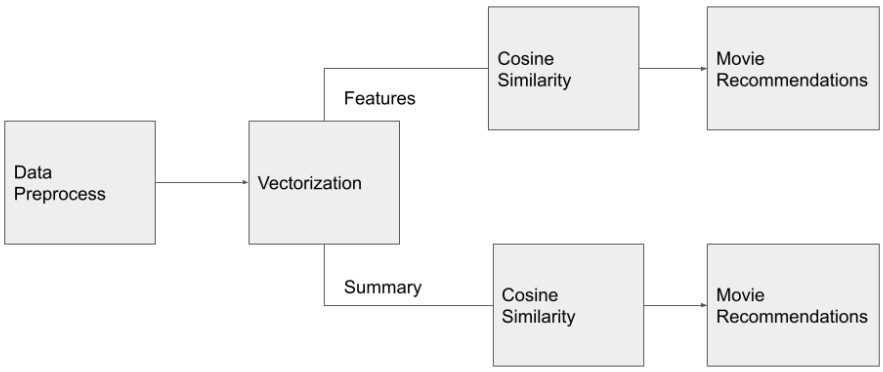

Figure 1. Flow diagram of our methodology.

4.2. Vectorization

A word embedding refers to a real number, a vector representation of a word. Here, when creating a word embedding space, we attempt to capture the relationships of the casting crew and grenes of the movies.

We used CountVectorizer for our genres, keywords, cast, and crew. As for the summary and the taglines of the movies, we used TF-IDF Vectors. CountVectorizer is a way to convert a given set of strings into a frequency representation (Heidenreich 2018). TF-IDF Term Frequency - Inverse Document Frequency (formula 1) is a method that is similar to CountVectorizer, but includes the concept of weighting, which means besides counting the frequency of a word appearing, we also need to consider the importance of that word.

$$W_{i,j} = tf_{i,j} \times log(\frac{N}{df_i})$$

Where, $tf_{i,j}$ is the number *of occurrence of i in j, df_i is the number of documents containing the i, N is the total number of the documents.*

Formula 1: The TF-IDF formula *(Saket 2020).*

Our algorithm used CountVectorizer for genres, keywords, cast and crew, since there is no need to down-weight any element, each keyword is equally important and unique for describing the movie. CountVectorizer will tokenize every element of these four columns, and for each movie, it will use binary representation indicating the existence of each keyword.

4.3. Cosine Similarity

Cosine Similarity measures the angle between the two vectors projected in a multi-dimensional space. It compares two documents on a normalized scale. Cosine Similarity is one way to calculate the text similarity. Text Similarity has to determine how the two text documents close to each other in terms of their context or meaning.We can find the cosine similarity (formula 3) between two movies by finding the dot product.

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{||\vec{a}|| \ ||\vec{b}||} = \frac{\sum_1^n \ a_i b_i}{\sqrt{\sum_1^n \ {a_i}^2}\sqrt{\sum_1^n \ {b_i}^2}}$$

Where, $\vec{a} \cdot \vec{b} = \sum_1^n \ a_i b_i = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$ is the dot product of the two vectors.

Formula 3: Cosine Similarity

||a|| is the Euclidean norm of vector a = (a_1, a_2,...,a_p) , defined as $\sqrt{a_1^2 + a_2^2 + \ldots + a_p^2}$. Conceptually, it is the length of the vector. Similarly, ||b|| is the Euclidean norm of vector b. The dot product of two vectors divided by the multiplication of two lengths will calculate the angle between two vectors, a and b.

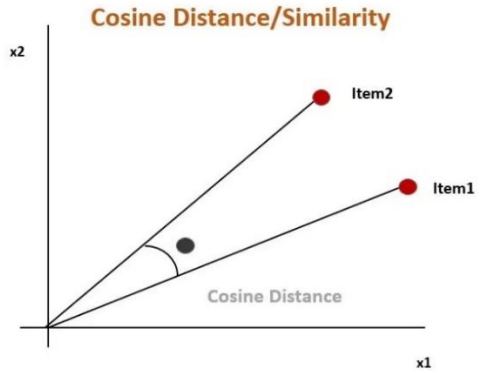

Figure 2. The graph is a visual representation of Cosine Similarity (Dangeti 2022)

As the figure 2 shows, the closer the items are by angle, the higher the cosine similarity. It means that if the angle is big, the two items are different from each other whereas if the angle is small, the two items are similar to each other.

The angle θ between two movies will determine how similar they are to each other. The θ ranges from 0 to 1 with 0 being the least similar and 1 being the most similar (Singh et al. 2020).

Our function will compute the cosine similarity between the one movie that is fed into the algorithm and the other movies in the dataset, then rank the movies based on cosine similarity, and rank the most similar ten movies as an output. As we used two different vectorizers before, we can compare the difference in the output of the top 10 movies.

5. Result

In the content-based framework recommendation system, we have used cosine similarity to generate movie recommendations based on either the summary or the features. We also have explored different approaches for the best result.

5.1. Summary-based recommendation

After vectorizing the summary of movies in our dataset using the TF-IDF vectorizer and calculating the cosine similarity scores, here are the top 10 movie recommendations of "*Knives Out*" (Table 2).

Table 2. Top 10 Summary Based Recommendations of "Knives Out".

1	A Girl's Tears
2	In Bed with Santa
3	Agatha Christie's Crooked House
4	Motherless Brooklyn
5	Serious Men
6	State of Play
7	I Am the Pretty Thing That Lives in the House
8	Hubert und Staller - Unter Wölfen
9	Only the Animals
10	What Keeps You Alive

5.2. Feature-based recommendation

For Feature-Based Vectorization, we used "Director", "Actors"(Top three leading actors), "Genre", and "Tags" as our vectorization keywords. Different from Summary Based Recommendation, we need to create a new column that is the integration of these four columns, and then vectorize each movie and calculate the cosine similarities. Only after that, we could begin our vectorization of the movies by using CountVectorizer, and for each movie, we extract the top ten movies of its cosine similarity chart as our recommendation. Here are the top 10 Feature Based Recommendations of "*Knives Out*" (Table 3).

Table 3. Top 10 Feature Based Recommendations of "Knives Out".

1	Under the Silver Lake
2	Hustlers
3	WHAT DID JACK DO?
4	Wind River
5	Preman Pensiun
6	Crimi Clowns: De Movie
7	Suburbicon
8	Joker
9	Matchstick Men
10	The Art Of Self-Defense

5.3. Combined recommendation

We found a way to mix up Summary-based and Feature-Based Recommender by integrating the two Cosine Similarity charts together, so when we have a movie title we can select the top ten from the mixed up chart as our recommendations. The result showed slightly different results than the previous two methods. Here are the top 10 Combined Recommendations of "*Knives Out*" (Table 4).

Table 4. Top 10 Combined Recommendations of "Knives Out".

1	In Bed with Santa
2	Agatha Christies Crooked House
3	Motherless Brooklyn
4	Serious Men
5	State of Play
6	I Am the Pretty Thing That Lives in the House
7	Hubert und Staller - Unter Wölfen
8	Only the Animals
9	What Keeps You Alive
10	Perfetti sconosciuti

In overall, the combined recommendation successfully performed our goal, which is mixing up two different recommenders. And further requirements could be added into our combined recommendations, like rank the result by IMDB scores.

5.4. Controlled Experiment

In the previous recommendation systems, we implemented two vectorizers for the best result because TF-IDF would down-weight the presence of an actor/director if they have acted or directed in relatively more movies. However, in this experiment, we vectorized the summary using CountVectorizer and features using TF-IDF Vectorizer while other parts of the system remain the same to see how this change

will affect the result (Table 5 and Table 6). To better compare the result, we used the movie "*Knives Out*" as we did for the previous recommendation systems.

Table 5. Top 10 Summary-Based recommendations of "Knives Out" after using CountVectorizer.

1	Agatha Christie's Crooked House
2	A Girl's Tears
3	In Bed with Santa
4	Shot caller
5	Waves
6	The Photograph
7	Motherless Brooklyn
8	The Irishman
9	No Such Thing as Housewives
10	Only the Animals

Table 6. Top 10 Feature-Based recommendations of "Knives Out" using TF-IDF Vectorizer.

1	Under the Silver Lake
2	Striptease
3	Hustlers
4	Joker
5	Fracture
6	Prisoners
7	Double Jeopardy
8	WHAT DID JACK DO?
9	Rush Hour
10	Child 44

To compare the results generated by the new recommendation systems to results of the old recommendation systems, we calculated the cosine similarity between the two results. For the summary-based recommendation system, the cosine similarity score is 50%, and for the feature-based recommendation system, the cosine similarity score is 40%. The overlaps between different recommendation systems are shown in Diagram 1 and Diagram 2.

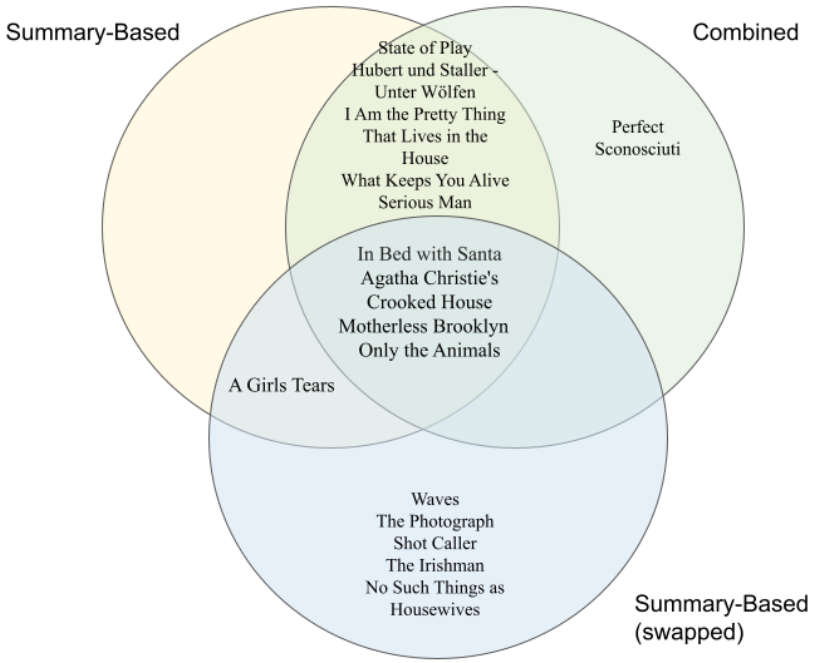

Diagram 1. Summary-Based.

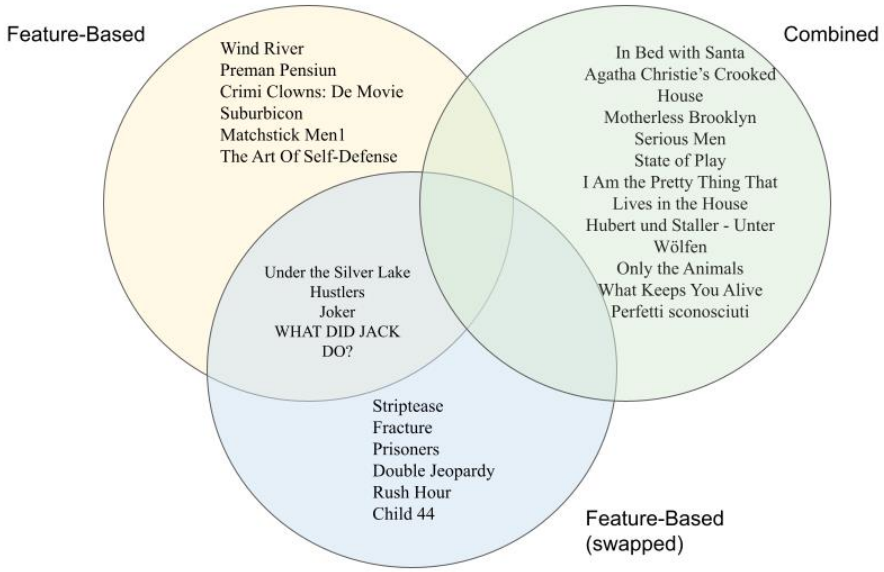

Diagram 2. Feature-Based.

6. Conclusion

As the results show, the summary-based recommender and the feature-based (which are the contents of variables "Tags", "Actors", "Directors" and "Genre" combined) gave significantly different results. We inferred that the difference is due to that summary-based recommender is plot-oriented and looks for similar components in stories of other movies whereas featured-based recommender searches for any movie with same genres, tags, director or actors. To illustrate this point, we will continue with our *Knives Out* example. *Knives Out* is a detective story in which the detective tries to unravel secrets in a family. Some key elements in the summary include "detective", "secrets", "unravel" and "family". Then, the

summary-based recommender will search for movies with similar elements. Consequently, the movie recommended is very likely to have a plot involving uncovering secrets and a storyline developed within families.

Unlike the summary-based recommender, feature-based recommender does not take plots and storylines into account. Instead, the only criteria of the algorithm is features. Therefore, the movie recommended can be anything that has the same genre or actors as the original input, while the plot and storyline can be completely unrelated to the original input. Despite the difference in results, both summary-based or feature-based recommenders are useful and contributive to some extent.

The result of our combined model is highly consistent with that of the summary-based recommender, and vastly different from that of the feature-based recommender. This high consistency suggests that movie abstracts can potentially be a more accurate and consistent criteria than features like genre, tags and actors.

As for the controlled experiment, the comparison also indicates that the type of vectorizer can also affect the recommendation results. In general, it lacks consistency for the same recommender using different vectorizers. The number of shared movies for both recommenders is no more than half, with 5 shared movies between two summary-based recommenders and 4 shared movies between two feature-based recommenders.

There are certainly limitations to our takeaways. First, our sample size is relatively small (only 6747 observations). Second, we did not acquire any numeric validation for our content-based recommendation system and our conclusions are solely based on a few examples. To address the limits, we plan to further investigate a hybrid model that bridges user profile and movie content so that we can test the accuracy of the recommendation system and also in hope of achieving a more consistent model.

References

[1] Jena, A. (2022, March 17). *Role of a movie recommender system in the streaming industry*. Muvi One. Retrieved July 8, 2022, from https://www.muvi.com/blogs/movie-recommender-system.html

[2] Sharma, L., & Gera, A. (n.d.). *A Survey of Recommendation System: Research Challenges*. Redirecting. Retrieved July 8, 2022, from https://answers.microsoft.com/en-us/windows/forum/all/cusersusernamedocumentsfile-folder-name/3c43589c-b582-433b-99ea-cfe3e1b2a270

[3] Ahmed, M. (n.d.). *Movie recommendation system using clustering and Pattern Recognition Network*. IEEE Xplore. Retrieved July 8, 2022, from https://ieeexplore.ieee.org/document/8301695/

[4] Uluyagmur, M. (n.d.). *Content-based movie recommendation using different feature sets*. Retrieved July 8, 2022, from http://www.iaeng.org/publication/WCECS2012/WCECS2012_pp517-521.pdf

[5] Lops, P., Jannach, D., Musto, C., Bogers, T., & Koolen, M. (2019, March 7). *Trends in content-based recommendation - user modeling and user-adapted interaction*. SpringerLink. Retrieved July 8, 2022, from https://link.springer.com/article/10.1007/s11257-019-09231-w

[6] Singh, R. H. (n.d.). *Movie recommendation system using cosine similarity and KNN*. Retrieved July 8, 2022, from https://www.researchgate.net/publication/344627182_Movie_Recommendation_System_using_Cosine_Similarity_and_KNN

[7] Tewari, A. S., Singh, J. P., & Barman, A. G. (2018, June 8). *Generating top-N items recommendation set using collaborative, content based filtering and rating variance*. Procedia Computer Science. Retrieved July 1, 2022, from https://www.sciencedirect.com/science/article/pii/S1877050918308718

[8] Wu, C.-S. M. (n.d.). *Movie recommendation system using collaborative filtering*. IEEE Xplore. Retrieved July 8, 2022, from https://ieeexplore.ieee.org/abstract/document/8663822

[9] Keshava, M. C., Srinivasulu, S., Reddy, P. N., & Naik, B. D. (2020). Machine learning model for

movie recommendation system. *International Journal of Engineering Research & Technology (IJERT)*, *9*(04).

[10] Uddin, M. N. (n.d.). *Enhanced content-based filtering using diverse collaborative prediction for movie recommendation*. IEEE Xplore. Retrieved July 8, 2022, from https://ieeexplore.ieee.org/document/5175981

[11] Afoudi, Y., Lazaar, M., & Achhab, M. A. (2021, July 24). *Hybrid recommendation system combined content-based filtering and collaborative prediction using Artificial Neural Network*. Simulation Modelling Practice and Theory. Retrieved July 1, 2022, from https://www.sciencedirect.com/science/article/pii/S1569190X21000836

[12] Mubarak, S. (2021, August 18). *Netflix dataset latest 2021*. Kaggle. Retrieved July 8, 2022, from https://www.kaggle.com/datasets/syedmubarak/netflix-dataset-latest-202

[13] Heidenreich, H. (2018, August 16). *Introduction to word embeddings*. Medium. Retrieved July 8, 2022, from https://towardsdatascience.com/introduction-to-word-embeddings-4cf857b12edc

[14] Saket, S. (2020, January 12). *Count vectorizers vs TFIDF vectorizers: Natural language processing*. Medium. Retrieved July 8, 2022, from https://medium.com/artificial-coder/count-vectorizers-vs-tfidf-vectorizers-natural-language-processing-b5371f51a40c

[15] Ahmed, I. (2020, May 16). *Getting started with a movie recommendation system*. Kaggle. Retrieved July 8, 2022, from https://www.kaggle.com/code/ibtesama/getting-started-with-a-movie-recommendation-system

[16] Han, J., & Pei, J. (n.d.). *Cosine similarity*. Cosine Similarity - an overview | ScienceDirect Topics. Retrieved July 8, 2022, from https://www.sciencedirect.com/topics/computer-science/cosine-similarity

[17] Dangeti, P. (n.d.). *Statistics for Machine Learning*. O'Reilly Online Learning. Retrieved July 8, 2022, from https://www.oreilly.com/library/view/statistics-for-machine/9781788295758/eb9cd609-e44a-40a2-9c3a-f16fc4f5289a.xhtml

[18] Singh, Ramni & Maurya, Sargam & Tripathi, Tanisha & Narula, Tushar & Srivastav, Gaurav. (2020). Movie Recommendation System using Cosine Similarity and KNN. International Journal of Engineering and Advanced Technology. 9. 2249-8958. 10.35940/ijeat.E9666.069520.