Noname manuscript No.
(will be inserted by the editor)

Genre based Hybrid Filtering for Movie Recommendation Engine

Arighna Roy · Simone A. Ludwig

the date of receipt and acceptance should be inserted later

Abstract With the dramatic rise of internet users in the last decade, there has been a massive rise in the number of daily web searches. This leads to a plethora of data available online, which is growing by the days. A recommendation engine leverages this massive amount of data by finding patterns of user behavior. Movie recommendation for users is one of the most prevalent implementations. Although it goes way back in the history of recommendation engines, collaborative filtering is still the most predominant method when it comes to the underlying technique implemented in recommendation engines. The main reasons behind that are its simplicity and flexibility. However, collaborative filtering has always suffered from the Cold-Start problem. When a new movie enters the rating platform, we do not have any user interaction for the movie. The foundation of collaborative filtering is based on the user-movie rating. In this paper, we have proposed a hybrid filtering to combat this problem using the genre labeled for a new movie. The proposed algorithm utilizes the nonlinear similarities among various movie genres and predicts the rating of a user for the new movie with the associated genres for the movie.

Keywords First keyword · Second keyword · More Movie Recommendation Engine · Hybrid Filtering, Genre Similarity · Cold Start · Content-based Filtering · Collaborative Filtering

1 Introduction

The amount of available digital information as well as the number of users on the Internet have increased dramatically of the past years. This has created potential challenges of information overload and fast access to relevant information.

Department of Computer Science
North Dakota State University
E-mail: arighna.roy@ndsu.edu

Department of Computer Science
North Dakota State University
E-mail: simone.ludwig@ndsu.edu

Information retrieval systems have addressed this problem but prioritization and personalization of information is still more or less absent in these systems. This has increased the demand for recommender systems. Recommender systems are information filtering systems and address the problem of information overload by filtering information from large amounts of dynamically generated data. This filtering is usually based on users preferences, interest, and observed behavior about items. Given this filtering process, recommender systems have the ability to predict whether a particular user would prefer an item based on their profile.

The recommendation systems available today can be broadly classified into three major categories; collaborative-filtering, content-based-filtering, and hybrid of the first two [1][2][3] [4].

Collaborative-filtering (CF) is grounded on identifying similar (nearest neighbors) items or users for a new item-user pair and estimate the rating based on a certain aggregation method. For item-based CF, similarities among the items are calculated based on the ratings by mutual users. For user-based CF, similarities among the users are calculated based on their ratings of mutual items watched. These similarity matrices are used to identify the nearest neighbors (items or users). Collaborative-filtering calculates the similarities among user ratings to make predictions. This method is only purposeful when users have rated a significant number of mutual items. Additionally, the absence of content of the items makes it impossible to match similar users unless the same items were rated by them. For example, suppose user A liked the movie “Lord of the Rings”, and user B liked the movie “Lord of the Rings II”. Although there is a similarity in their preferences, collaborative-filtering will fail to match them.

Content-based-filtering has always been important in the field of recommendation engines because it addresses the Cold-Start problem [5]. Both user-user CF and item-item CF are dependent on the user-item interaction. When a new item is introduced to the recommendation engine, it does not have any interaction data with the user at that point. Without existing ratings by the users, it is not possible to find similar items, which is the foundation of item-item CF. Also, as no user has rated the item, there is no way a recommendation can be made. Thus, addressing this shortcoming is the foundation of user-user CF. Additionally, data sparsity is predominant in any item review data set. Therefore, items with an insignificant number of user ratings show inferior performance compared to the ones with a significant number of user ratings [6][7]. However, content-based CF find similar items based on their metadata. In the domain of movie recommendation systems, the metadata of the items could be director, length of the movie, language, etc. [8] proposes a content-based filtering technique where attributes are extracted from the ratings, and this deals with data sparsity. Although it handles cold start problems to some extent, it entirely depends on the content of the items. For the sake of simplicity, we will limit our discussion to movie recommender systems. However, this approach can be extended to any recommender systems where the items are tagged under certain predefined categories. The disadvantages of the approach discussed in [8] are twofold, operations cost, and quality of recommendation. From the standpoint of operations cost, the collection of content of movies is expensive as those are hand-engineered and require a lot of domain knowledge. With the increase in the number of movies across the globe, it is becoming nearly impossible to generate features for all those. Furthermore, the quality of the recommendation is entirely dependent on the quality of the feature generation process, and thus its ability is limited in terms of expanding users' existing preferences.

A hybrid technique of collaboration with content tackles the shortcoming of feature extraction by combining information captured by both content-based-filtering and by collaborative-filtering. In collaboration filtering with content-user, the similarity matrix is built using both the content of the movies as well as the rating of movies in common. The feature extraction based on the content of the movie is done by content-based techniques. The weights of the extracted features designate the importance of the corresponding feature to the user. Like collaborative-filtering, the similarities between users are calculated. However, the weights of the features are used in place of user ratings to calculate the similarity. This approach has lots of features to calculate the similarities between users. Hence, it deals with the issue of users having an insufficient number of rated movies. Moreover, the extracted features are capable of capturing users' preferences outside of their usual environment. This is a considerable uplift from content-based-filtering. However, we still need a significant number of users that have rated a movie for the recommendation. Similar to collaborative-filtering, it is not possible to recommend a brand-new movie unless there is a user who has already rated it.

We have witnessed an uprising popularity of releasing dubbed/translated versions along with the original spoken language of the movie [9]. This way the reachability of the content overcomes the barrier of language. However, communities are formed based on user-movie relationships. While utilizing the existing movie review history, we cannot explore users' interests across those communities. Hybrid filtering can combine the content information and thus can overcome this barrier. The traditional hybrid filtering methods rely on movie meta-data most of which are isolated within those communities. We have proposed a hybrid filtering algorithm to address this problem by utilizing the genre information. Additionally it deals with cold-start problem. We have combined users' watch history with movie genre information to recommend movies across the above mentioned communities. We have shown that the proposed method is capable of recommending movies in foreign languages that were not explored by the test user yet.

The rest of the paper is organized in the following way. Section 2 discusses the limitation of the state-of-the-art methods on cold-start scenarios and how this paper contributes to mitigate those issues. Section 3.1 explains the algorithm with mathematical notations and Section 3.2 shows the flowchart diagrams. Section 4.2 describes the process of data preparation to simulate the cold-start scenario. Section 4.3 and 4.4 show the experimental results on the MovieLens-100k data set with various levels of data sparsity. Section 4.5 shows the experimental results on the MovieLens-25m data set. In addition, we have further improved the hybrid filtering with nonlinear multivariate similarity metrics. The comparative experimental results are shown in Section 4.6.

2 Related Work

Various methods have been proposed over time to build product recommendation engines. Due to the variety of available data and the use cases, there is no one-size-fits-all technique. However, most of the proposed methods can be broadly classified into two categories, content based filtering and collaborative filtering.

Content based filtering also includes context based filtering. They use time and space features of the items and users [10]. Collaborative filtering, in contrast, leverage the interaction among the items and the users to predict the rating of a user-movie pair. Furthermore, collaborative filtering methods are broadly classified into two categories, nearest neighbor based models [11] and matrix factorization based models [12]. Nearest neighbor based filtering methods focus on the similar users or items to estimate the likely rating. Matrix factorization is captures the latent relationships between the users and the items, project that to a low-dimensional space and compute the similarity in the reduced space [13]. The most common matrix factorization is Singular Value Decomposition which was also used for the winning algorithm in the Netflix Prize [14] [15]. Although the latter category has gained popularity lately, the former category still dominates the industrial level implementations. The prime reason behind this is its simplicity and intuitiveness [16]. Due to the same reason, Nearest Neighbor based models win over more sophisticated models in a variety of domains. For example, prediction of remaining useful life on degradable products have been dependent on models based on pattern recognition models [17]. These models are grounded on statistical estimation of the decaying process. However, in many scenarios, nearest-neighbor based models win over the former mentioned sophisticated models due to its simplicity [18].

Any form of collaborative filtering suffers from groups of users known as gray and black sheep. Unusual users with inconsistent preference compared to the other users are referred as the Gray sheep [19]. Black sheep refers to the group of users that have idiosyncratic tastes [20]. The time complexities of collaborative filtering is linear and with the growing size of the user-movie rating data, it becomes difficult to keep up with the runtime of these algorithms [21]. It is very common that synonymous (or the exact same) items with different names occur in a recommender system data set. Collaborative filtering fails to detect those unless users have shown similar behavior towards rating them.

[22] has proposed a content-based filtering approach that calculates the contents' similarity. Movie genres are defined by content experts. Hence, it is more reliable than system-generated features. Additionally, it deals with the Cold-Start problem as similarities are calculated only based on content. It calculates the mutual genre count to estimate the similarities among genres. However, it does not consider the user rating of each genre while calculating the similarities. Additionally, the experiment is performed on a test set of 10 movies, which fails to establish the reliability of the proposed method.

[23] proposes a recommendation system based on the extracted low-level visual features from the trailers of the movie. The authors have used visual feature extraction as an alternative to the metadata of the movies. This method is dependent on the machine-generated features, whereas genre information is far more reliable as it is generated by domain experts. Also, the algorithm-guided feature generation is an unstable area in computer vision. Furthermore, this approach fails to expand users' existing tastes as it does not consider the interaction among the high-level components of the movie. [24] proposes a similar approach but for videos. As the videos do not have man-made sample data (trailer for the movie), it utilizes the whole content of the video to extract the feature. This is not feasible for movies.

[25] uses the plot summaries of the movies to generate natural language processing based features. The authors combine the high-level meta-data information with the low-level features generated by the plot summaries. Similar to [23], it

fails to expand users' existing preferences. Also, the bag of words approach can be totally misleading while defining the theme of the movie.

[26] calculates the relative importance of genres to each user from the review matrix. The weighted average importance of the user of the corresponding tagged genres of a movie is estimated as the predicted rating of that user-movie pair. This approach restricts the prediction of users' existing preferences as it does not consider the similarities among the genres. For example, if a user has rated 'Batman' as 5, and he has watched only one sci-fi movie, the relative importance of 'action' genre for the user becomes 5. Whereas, in reality it could be the case that he rated it high because it was an 'action' film. This model would incorrectly recommend all the sci-fi movies to the user.

[27] proposes a content-based recommendation system that leverages six high-level features, such as the origin of the movie, which is a major limitation of this approach. Most of the real-time training data sets contain selection bias. For example, if a user has mostly watched Russian movies, this model will most likely limit the predictions within Russian movies whereas, in today's movie world streaming platforms generate audio in multiple languages. Features like the genre is more universal and does not have this limitation.

[28] extracts the movie interest of users from user tweets. Although the authors aim to solve a different problem with an entirely different type of data set, we share the same foundation in terms of the approach, which is the recognition of user interest.

[29] proposes a recommendation technique for users through mobile devices. The system leverages the GPS information from the movie search operations on mobile devices. The primary pitfall of this approach is the assumption of user behavior. We should not assume the inference of user touchpoints based on intuition. If a user has searched a movie on a mobile device, then it does not necessarily show his interest if he did not watch the movie.

The method implemented in [30] leverages the generated features of the genre. It enriches the diversity of the recommendation as it utilizes genre data. This paper attempts to define genre diversity using a Binomial framework, which is different from our problem statement. However, it shows the robustness of genre data in terms of enhancing the diversity of the recommendation.

The cold start problem can also be addressed using the similarities among the users based on their interactions on social networks [31]. This approach tries to fill the gap in the rating data to tackle the data sparsity. However, it is heavily dependent on an external data source, which is extremely difficult to stitch with the rating data set. In an ideal world where all the users from various platforms are centrally connected with network governance, this method can prove to be effective. Although, that is far from feasible to apply on a real-life recommendation platform.

[32] proposes an algorithm to deal with cold start problem. They calculate the latent features with deep neural network using the users data (when available), genre information and movie title. However, there are various limitations on this method. For Movie Lens 1m data set, this algorithm is dependent on the user demographic information. User demographic information is not readily available for many new recommendation systems. Hence, this method is limited to the recommender systems that are able to capture the user demographic information. For example, MovieLens-100k does not have user information. Hence, it is limited

to the users that allow their demographic information to the system. Even with the provided demographic, the reliability of the information is often questionable. Hence, we do not rely on the user demographic data. Secondly, the simulation of the cold-start data is done based on the acquisition of the user. This paper has considered the users acquired in the last year. We have simulated the data in such a way that the users have at most 20 ratings. For example, in Movie Lens 100k data set, if we consider the ratings from 2003, which is the latest year on the data set, only 878 ratings are coming from the users having less than 21 ratings. Whereas there are 3348 ratings posted in 2003. Our data preparation simulates the cold-start problem closer to this work due to higher data sparsity.

[33] proposes an interesting ensemble method where various components of the rating systems are ensemble with dynamically generated weights. This work achieves an impressive error rate in MovieLens-100k data set. However, it does not establish the method to address the problem of cold-start. The data preparation included the whole data set; there was no cold-start simulation. We have captured the latent associations among the genres and combine that with the collaborative filtering and provided the proof of concept how it outperforms the standard algorithms in the simulated data set.

[34] provides a list of techniques that leverage the Internet Of Things (IoT). These recommender systems utilize activities collected from various other platforms that are interconnected. Although these approaches are very effective for personalized recommendations, they are limited to availability of the interconnected IoT applications data. Another set of recommender systems such as ISoTrust-Seq [35] leverage social media data. [36] provides a great list of such techniques. However, like the IoT based approaches, these methods are also limited to the availability of the external data sources.

Collaborative Filtering methods have always been vulnerable to 'shilling attacks'. In these scenarios, fake users spoil recommendation systems' predictions by systematically injecting fake profiles and reviews. [37] proposes a method that tackles this problem to some extent by factoring in the objective and subjective trust among the users.

In a nutshell, we are proposing a hybrid filtering method where the genre information of the movies is used with the association among the users in terms of similar movies. Unlike many other contents of movies such as extracted features from the media, genre data is very easy to gather. Additionally, genre information is reliable as it is not machine-generated. This proposed algorithm will be effective in dealing with the Cold-Start problem. The major contribution of this algorithm is to expand users' preferences across genres. We have implemented the algorithm using the ML-100k data set and performed robust experiments to validate the effectiveness of the proposed method.

3 Approach and Methodology

3.1 Algorithm Description

In this section, the methodology of the proposed algorithms for predicting movie ratings based on genre information will be explained. Let us start by denoting the entities involved in this experiment, in particular we have three major components.

$\{U_1, U_2, \ldots, U_p\}$ is the set of users that contribute to the review data set. $\{M_1, M_2, \ldots, M_q\}$ is the set of movies that are utilized for rating by the set of users mentioned earlier. Each of the movies belongs to multiple genres, and $\{G_1, G_2, , G_{18}\}$ is the set of genres. The rating data set is:

$$R \in r(i, j)$$

where $r(i, j)$ denotes the rating of user U_i for M_j.

We first calculate the average rating of each user for each genre. This is calculated from the user-movie rating matrix combined with the movie-genre adjacency matrix.

3.1.1 Genre-Average Hybrid Filtering - GAHF

Genre-average hybrid filtering (GAHF) is applicable to the test cases where the movie has at least one genre that has been tagged by the user. For example, if (U_a, M_b) is the test case where we are trying to predict the rating of a movie denoted by M_b by the user denoted by U_a. If M_b is tagged under genres $G_{1 \ldots i}$, and under the assumption that U_a has rated all the tagged genres, then the predicted rating is calculated as:

$$\sum_{n=1}^{i} r(a, n)$$

where $r(a, n)$ denotes the average ratings for movies of genre G_k by user U_a.

3.1.2 Genre-Similarity Hybrid Filtering - GSHF

We propose genre-similarity hybrid filtering (GSHF) that utilizes the similarities between every pair of genres and overcomes the limitation of of the test user having to rate at least one genre of the test-case. In other words, it is capable of working on the use cases where the user has never rated a genre that the movies fall under. We calculate the genre-genre similarity matrix from the user-genre average rating matrix. This provides the similarity between every pair of genres based on the average rating of every user to the genres. We have considered various similarity metrics which will be discussed in Section 3.3.

We will utilize the genre-genre similarity matrix to predict the rating for a user-movie pair where the user has never rated the genres of the movie. For example, if $(U_a,\ M_b)$ is the test case for which we are predicting the rating of movie denoted by M_b by the user denoted by U_a. If M_b is tagged under genres $G_{1,2,\ldots,i}$, the predicted rating is calculated as:

$$\sum_{n=1}^{i} r(a).sim(n)$$

where, $r(a)$ is the average-rank vector by U_a for the genres except the ones in G; and $sim(n)$ is the similarity vector of G_n with the corresponding genres.

3.2 Process Flowchart Diagrams

Figure 1 is a process flowchart that visually describes the data preparation for our experiments to simulate the cold-start situation. And, Figure 2 is a process flowchart that explains the overview of the execution of the model on a new test case.

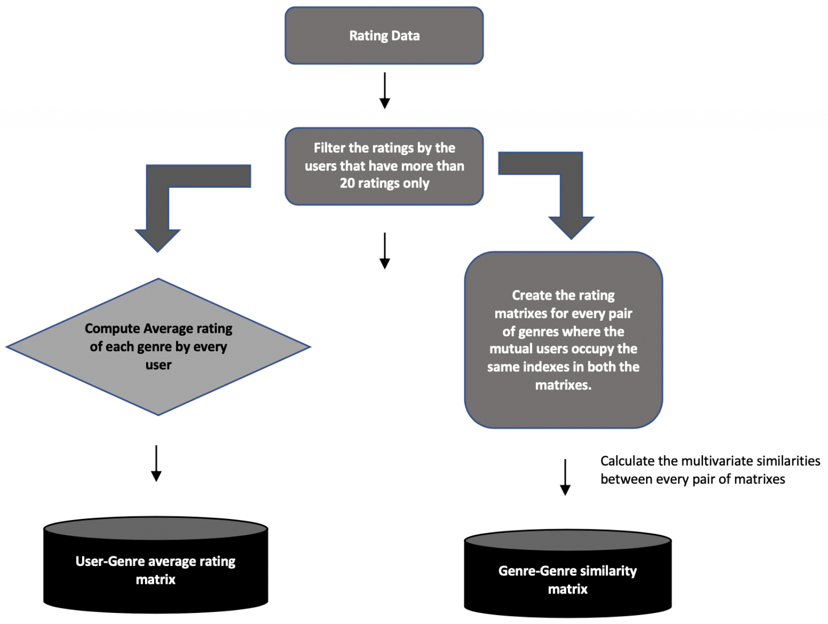
Rating Data
Filter the ratings by the users that have more than 20 ratings only
Compute Average rating of each genre by every user
Create the rating matrixes for every pair of genres where the mutual users occupy the same indexes in both the matrixes.
Calculate the multivariate similarities between every pair of matrixes
User-Genre average rating matrix
Genre-Genre similarity matrix

Fig. 1 Process Flowchart for the data preparation for cold-start simulation

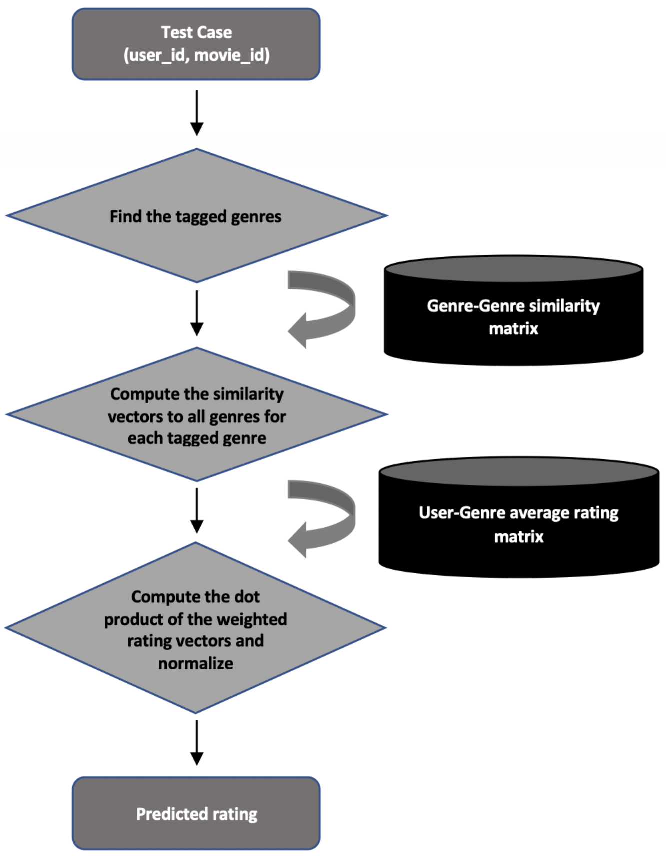
Test Case
(user_id, movie_id)
Find the tagged genres
Genre-Genre similarity matrix
Compute the similarity vectors to all genres for each tagged genre
User-Genre average rating matrix
Compute the dot product of the weighted rating vectors and normalize
Predicted rating

Fig. 2 Process Flowchart for rating prediction for a test case

3.3 Multivariate Similarity Metrics

Canonical Correlation (CC) capture the linear relationship between two multivariate data sets. It finds the linear combinations of each set of the variables that yield the maximum correlation between them. In other words, we construct a weighted average of the variables, which is called variate. Then, we calculate the Pearson Correlation between the variates.

For example, let us suppose, **X** and **Y** are the ratings vectors by the mutual set of the users between G_i and G_j, respectively. These set of vectors are denoted as,

$\mathbf{X} = (X_1, \ldots, X_p)'$ and $\mathbf{Y} = (Y_1, \ldots, Y_q)'$

To find the Canonical Correlations, we need to construct the linear combinations:

$$Z = u_1 Y_1 + u_2 Y_2 + \cdots + u_p Y_p$$

and

$$W = v_1 X_1 + v_2 X_2 + \cdots + v_q X_q$$

such that r_{ZW} is a maximum.

To calculate Canonical Correlation, we start with the covariance matrix between $\mathbf{X}$ and $\mathbf{Y}$.

Let S be XY covariance matrix denoted as,

$$S = \begin{bmatrix} S_{yy} \; S_{yx} \\ S_{xy} \; S_{xx} \end{bmatrix}$$

Next, we partition the covariance matrix as below,

$$A = S_{yy}^{-1} S_{yx} S_{xx}^{-1} S_{xy}$$

Eigenvalues of A are canonical correlations squared, therefore

$$R_c = \sqrt{\mu_i^2} = \mu_i$$

As mentioned earlier, Canonical Correlations can capture the linear relationships between two multivariate data sets. There are a few non-linear algorithms that capture the nonlinear relationships. Both Nearest Neighbor Intersection (NNI) [38] and Cluster Overlap (CO) [39] work well on these tasks.

We are going to implement NNI to substitute CC in Section 4.6 to portray how the performance improves with this similarity metric.

NNI calculates the nearest neighbors based on the average rating vectors for each user based on both the genres and calculates the mutual nearest neighbors to estimate the similarity between them. First, we identify the of the nearest neighbors of an instance m in both data sets. Every neighbor has a probability of $k_{nn}/(N-1)$ to be in the subset defined by x_m and a probability of $k_{nn}/(N-1)$ to be in the subset that's defined by y_m. Aggregated, the expected number of mutual neighbors between any pair of random neighborhoods is $\frac{k_{nn}^2}{N-1}$.

The number of mutual observations of neighborhoods is aggregated over every possible instances m yields in a sum of

$$n_{\text{expect}} = E\left(\sum_m |U_l^x \cap U_l^y|\right) = \frac{k_{nn}^2}{N-1}$$

if column vectors $\mathbf{X}$ and $\mathbf{Y}$ are unrelated.

Post aggregation, the expected sums are big enough for a Poisson distribution to be approximated by a Gaussian distribution. We used Poisson distribution for comparison. After the approximation, we collect the z-score, which is how many standard deviations above the mean.

For the CO algorithm, we count the number of overlapping data points between every pair of clusters, each generated on different data sets. And, we compare that with the expected number of overlaps with random chance.

$$\chi^2 = \sum_{i=1}^{k_{\text{clust}}} \sum_{j=1}^{k_{\text{clust}}} \frac{\left(|C_i^X \cap C_j^Y| - \frac{|C_i^X|\,|C_j^Y|}{N}\right)^2}{\frac{|C_i^X|\,|C_j^Y|}{N}} \tag{1}$$

where C_i^X and C_j^Y are the members of clusters X_i and Y_j respectively, i.e. $|C_i^X \cap C_j^Y|$ is the number of observations in the overlap of the two clusters, and $\frac{|C_i^X|\,|C_j^Y|}{N}$ is its expected overlap between the two independent clusters. We can consider the χ^2 distribution to calculate the probabilities which show the strength of the relationship.

4 Experimental Setup and Results

4.1 data sets

We have used the MovieLens data set [40] to build a proof of concept and to evaluate our proposed approach. It has various versions of the data set which are identified by the size of the rating matrix. We have considered the most commonly used version, which is also the smallest, ML-100k. Additionally, we have used the largest version, ML-25ml, to establish the robustness of the algorithm. ML-100k has 100,000 ratings where $1,682$ movies are rated by 943 users. Each rating is an integer ranging from 1 to 5. It has 93% sparsity, which means if we create an adjacency matrix on the user-movie combinations, then 93% of the cells will be empty. ML-25ml consists of 25 million ratings as the suffix identifies. It has $162,541$ users and $59,047$ movies. The ratings range from 1 to 5, but with an interval of 0.5.

Table 1 shows the data based on the user id, movie id, and the rating.

Table 1 Movielens-100K rating data set

userId	movieId	rating
877	381	4
815	602	3
94	431	4
416	875	2
500	182	2

4.2 Data Preparation for Experiments

The prime motivation of this paper is to predict the rating of movies that have never (or very few ratings) been rated by the user population. This is a classic problem in the world of recommendation engines and is referred to as the 'Cold Start' problem. We intend to utilize the genre information of the movie to address this issue. We build the training and testing data set in such a way that reflects this situation and can establish the effectiveness of our algorithm.

We have calculated the average rating of each genre for the user. For each user-movie combination, we calculate the average of average ratings of each tagged genre; it is zero if a user has not rated a certain genre. The rationale for the same comes from the 'Missing Not At Random' (MNAR), which states that the missing values indicate non-ignorable non-responses.

We have executed two sets of experiments. First, we executed experiments to establish the Genre similarity based hybrid filtering with comparative results for

Memory based Collaborative filtering methods. We prepared the data in such a way that the test items appears on the train data set less than a certain frequency. We calculated the similarity matrix between every pair of genres and estimated the rating of a movie considering the average rating of each genre by the users combining the similarities among them. The second experiments expand the comparison results on the performance of the algorithm using various similarity metrics.

4.2.1 ML-100k Data Set

We performed the experiments on the subsets where there are limited number of ratings for each user or item. In one experiment, we prepared the data in a way that the users have rated less than a certain number of movies. In the other experiment, we prepared the data in a way that the movies have been rated by less than a certain number of users.

Furthermore, for the subset, we have split the data into train and test buckets with random sampling with a 9:1 ratio. The data set is large enough for random sampling to maintain similar distributions over various strata. For the interest of future reference, we are going to call these frequencies as per-user-rating-count and per-item-rating-count, respectively.

Table 2 shows the size of the training data for various per-user-rating counts and per-item-rating counts.

Table 2 Sizes of training and testing split for various frequency counts

Frequency	Train size for per-user-rating	Train size for per-item-rating
30	4257	7446
40	7299	10476
50	10376	14656

As we simulate the cold-start environment, we are purposefully introducing data sparsity, which results in test cases where the user has no rating in the train split. We have dropped such cases from the test case. If it is a movie that has never been rated in the train split, we can utilize the genre similarities to come up with a prediction, which is however not the case for a new user. Also, collaborative filtering algorithms fail to predict the rating if there is no mutual users or items. For those cases, we have used the overall average rating of the movie.

Table 3 shows the percentage of data for various ratings on both the training and the testing splits. This split is shown for the subset where the users have less than 50 ratings.

Table 3 Percentage share on data for various ratings on training and testing split for per-user-rating-count 50

Rating	Training split	Testing split
1	5.8%	6.1%
2	10.9%	9.4%
3	24.3%	26.8%
4	34.6%	36.3%
5	24.3%	22.4%

Table 4 shows the percentage of data for various ratings on both the training and the testing splits. This split is shown for the subset where the items have less than 50 ratings. For the interest of future reference, we are going to call this frequency as per-item-rating-count.

Table 4 Percentage of various ratings on training and testing split for per-item-rating-count 50

Rating	Training split	Testing split
1	14.6%	14.6%
2	17.4%	18.4%
3	31.1%	29.1%
4	25.3%	25.3%
5	11.6%	11.4%

4.2.2 ML-25ml Data Set

We have performed similar experiments on a subset of the ML-25m data set. We have filtered in the users that have 20 ratings in the full data set. From $4,611$ such users, we have randomly picked 500 for our experiments. Table 5 shows the percentage of the data for each rating value before the random sampling, training split after random sampling, and the testing split. We can see that the distribution is maintained after the random sampling.

Table 5 Percentage of ratings on training and testing split for 25m data set

Rating	Before random Sampling	Training split	Testing split
0.5	3.2%	2.9%	4.5%
1	4.2%	4.2%	4.0%
1.5	2.6%	2.4%	3.2%
2	6.8%	6.9%	5.6%
2.5	4.6%	4.6%	3.5%
3	19.1%	18.8%	18.1%
3.5	8.3%	8.6%	8.3%
4	23.6%	24.5%	24.0%
4.5	7.8%	7.5%	7.5%
5	19.8%	19.9%	20.8%

4.3 Experiments on Per-User-Rating-Counts

4.3.1 Comparative Performance

To measure the performance on the test split, the Mean Absolute Error (MAE) and the Mean Squared Error (MSE) are calculated. The results are collected with three decimal places. We have compared the performance of the proposed algorithm with both Item-Item Collaborative Filtering (IIHF) and User-User Collaborative Filtering (UUCF). Furthermore, as we are working with genre information, we have also reported the performance of Genre-Average Hybrid Filtering (GAHF).

Figure 3 and Figure 4 show the comparative performance of the Item-Item Collaborative Filtering (IIHF), User-User Collaborative Filtering (UUCF), Genre-Average Hybrid Filtering (GAHF), and Genre-Similarity Hybrid Filtering (GSHF) for various settings of the data sets.

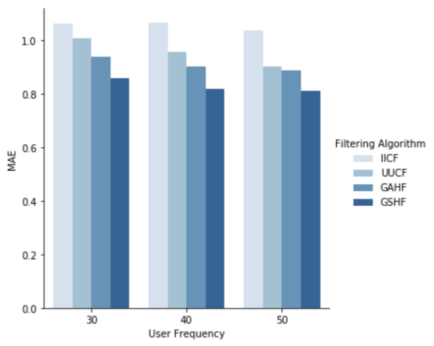

Fig. 3 MAE for various per-user-rating counts

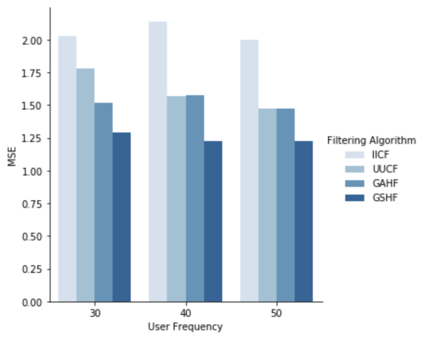

Fig. 4 MSE for various per-user-rating count

We observe a downward trend in loss for UUCF as the per-user-ratings-count increases. As the per-user-ratings-count increases, there are more mutual users for a test-movie to leverage which leads to lower loss. GSHF, on the other hand, maintain a steady performance across the per-user-ratings-count. It establishes that GSHF is not impacted by the sparsity of the data as much, which makes it a perfect choice for the movie recommendations under the cold-start scenario.

Additionally, we observe the loss of each of the algorithms are much higher than the reported loss on the ML-100k data set by the state-of-the-art algorithms. This is, for obvious reasons, due to the cold-start simulation. We are going to continue with the data preparation for the cold-start simulation in this paper.

4.3.2 Genre Count

Table 6 shows how the prediction loss of GSHF varies over the genre counts. We observe that the prediction losses are stable across the various genre counts. It proves the robustness of the performance of the algorithm. Although GSHF leverages the genre information, it holds equal power while dealing with the cases with small number of tagged genres. The similarities of the tagged genres with other genres suffice the information.

Table 6 Prediction loss for various genre counts of movies

genre count	**MAE**	**MSE**
1	0.845	1.318
2	0.803	1.177
3	0.749	1.046
4	0.882	1.48
5	0.5	0.5

We observe interesting values for MAE and MSE for genre count 5. Once investigated further, there were only 4 such test cases and the predictions are shown in Table 7.

Table 7 Predictions for the cases with 5 tagged genres

Actual rating	Predicted rating
3	4.0
5	4.0
4	4.0
3	3.0

4.3.3 Confusion Matrix

Figure 5 shows how the prediction values are distributed around the actual ratings. The actual labels are given on the y-axis whereas the predicted values are given on the x-axis. The first diagonal contains the correct predictions. Based on the figure we can see that most of the ratings revolve around rating class 3 and 4. Also note that a rating of 1.0 was never predicted.

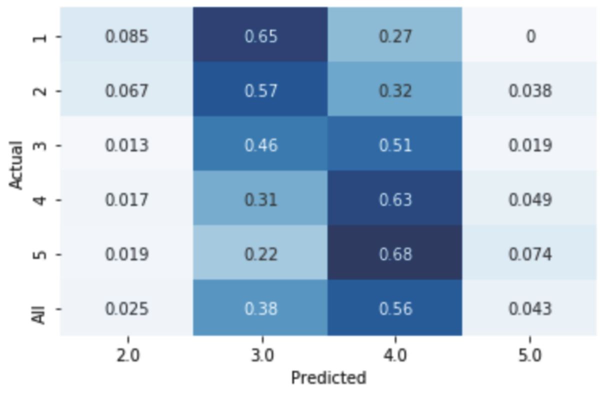

Fig. 5 Heatmap of the confusion matrix

4.4 Experiments on Per-Item-Rating-Counts

4.4.1 Comparative Performance

For the next experiments we prepared the data in a way that the items have received less than a certain number of ratings. Figure 6 and Figure 7 show the comparative performance of the algorithms for the data preparation with various per-item-rating counts.

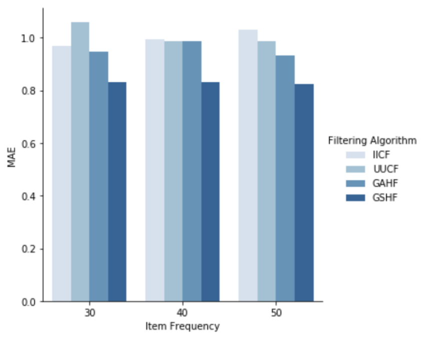

Fig. 6 MAE for various per-item-rating counts

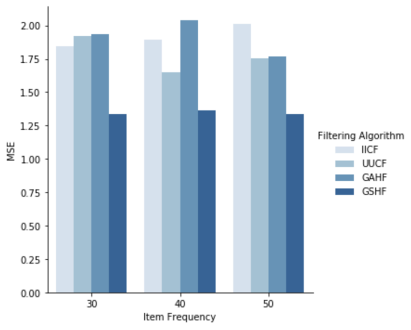

Fig. 7 MSE for various per-item-rating count

Unlike Figure 3 and Figure 4, Figure 6 and Figure 7 do not show any trend on the performance of UUHF or IIHF with per-item-ratings-count. However, GSHF shows stability and superiority in performance.

4.4.2 Genre Count

Table 8 shows how the prediction loss of GSHF varies over the genre counts.

Table 8 Prediction loss various genre counts of movies

genre count	**MAE**	**MSE**
1	0.809	1.298
2	0.826	1.329
3	0.912	1.574
4	0.895	1.421
5	1	1

Similar to the Table 6, MAE and MSE for genre count 5 are also interesting in Table 8. Once investigated further, there were only 2 such test cases and the predictions are shown in Table 9.

Table 9 Predictions for the cases with 5 tagged genres

Actual rating	**Predicted rating**
3	4.0
4	3.0

4.4.3 Confusion Matrix

Figure 8 shows how the prediction values are distributed around the actual ratings. The matrix indicates that most ratings predict rating value 3 as indicated by the color shading.

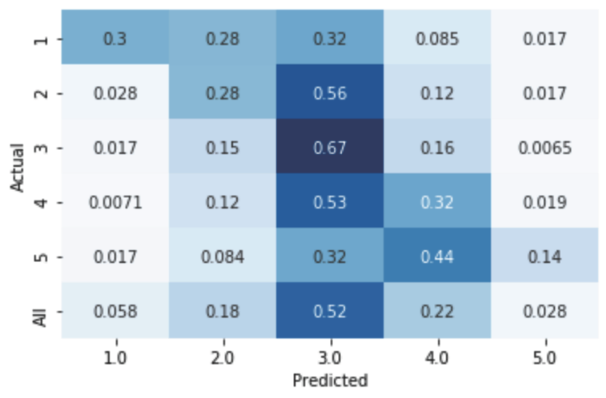

Fig. 8 Heatmap of the confusion matrix

4.5 Experiments on 25ml Data Set

4.5.1 Comparative Performance

Table 10 shows the comparative performance on all 4 algorithms for the aforementioned subset of the ML-25m data set. We can clearly see that both GAHF and GSHF have smaller MAE and MSE values compared to the comparison approaches UUCF and IICF. GSHF outperforms GAHF with the lowest values of 0.792 and 1.104 for MAE and MSE, respectively.

Table 10 Comparative prediction loss on ML-25ml

Algorithm	**MAE**	**MSE**
UUCF	1.119	1.987
IICF	1.029	1.962
GAHF	0.847	1.266
GSHF	0.792	1.104

4.5.2 Genre-count

Table 11 shows how the prediction loss varies over the genre counts. We can observe a sharp decrease in loss for genre-count higher than 4. There are only 9 cases with genre-count 6, and 6 cases with genre-count 7. We can explain this decrease in loss as a bias. However, there are 45 cases with genre-count 5, which is large enough to eliminate the random chance of the loss to be low.

Table 11 Prediction loss various genre counts of movies on ML-25ml

genre count	MAE	MSE
1	0.838	1.139
2	0.754	1.096
3	0.797	1.133
4	0.835	1.169
5	0.767	0.839
6	0.611	0.472
7	0.667	0.667

4.5.3 Confusion Matrix

Figure 9 shows how the prediction values are distributed around the actual ratings. The ratings are shown in 0.5 increments. Given the color shading we can see that most of the predicted ratings range from 2.5 to 4.0.

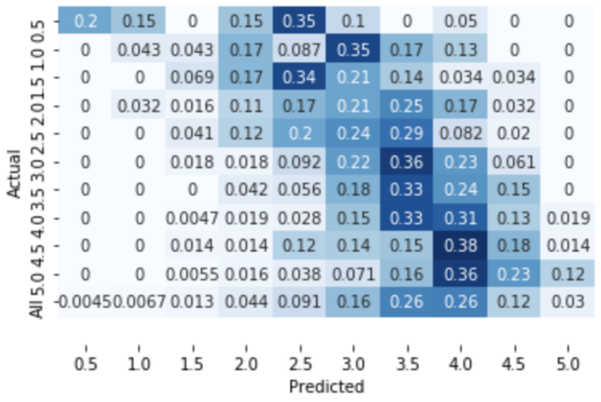

Fig. 9 Heatmap of the confusion matrix

4.6 Similarity of GSHF

4.6.1 Experiments on ML-100k Data Set

In this section, we will compare the performance of GSHF with two different similarity metrics, namely NNI and CCA. As we discussed in Section 3.3, NNI is capable of capturing the non-linear relationships between two data sets, which is not the case for CCA. Figure 6 and Figure 7 show how the prediction loss varies for GSHF with CCA or NNI at various per-user-rating-count values. We observe a slightly downward trend in the loss as the per-user-rating-count increases, although there is no monotonous trend. GSHF consistently performs better (lower loss) with NNI than with CCA as the similarity metric.

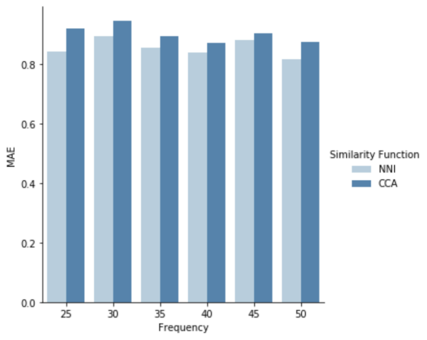

Fig. 10 MAE for various per-user-rating counts

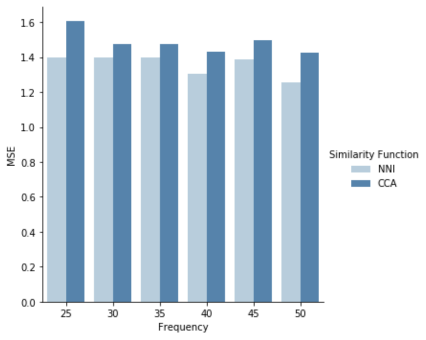

Fig. 11 MSE for various per-user-rating-count

Figure 12 and Figure 13 show how the prediction loss varies for GSHF with CCA or NNI at various per-movie-genre-count for ML-100k. CCA performs slightly better than NNI if we consider the MAE for genre-count 4.

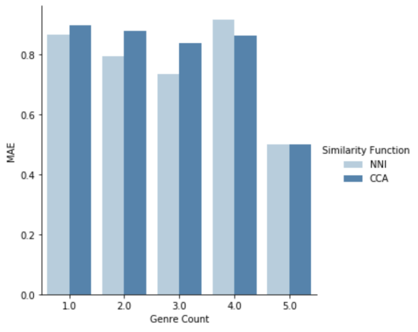

Fig. 12 MAE for various genre counts

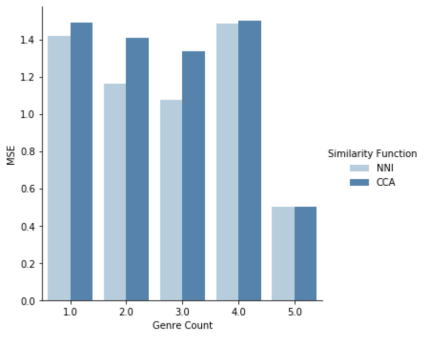

Fig. 13 MSE for various genre counts

4.6.2 Experiments on ML-25ml data set

We performed similar experiments on the 25ml data set. However, for the interest of execution time, we only performed the experiment for per-user-rating-count as 50. Figure 14 and Figure 15 show how the prediction loss varies for GSHF with CCA or NNI at various per-movie-genre-count for ML-25ml.

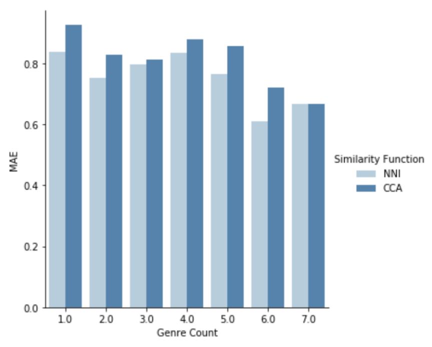

Fig. 14 MAE for various genre counts

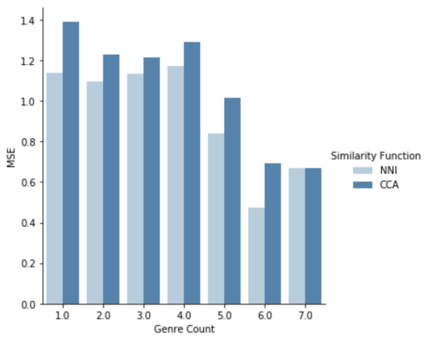

Fig. 15 MSE for various genre counts

5 Conclusion and Future Work

We have proposed a hybrid filtering technique that considers movie genres as the content of the movies. Genre is a reliable content measure on the movie as it is generated by the content experts such as directors. This algorithm is not sensitive to Cold-Start problems or limitations on users' preferences. We have applied this approach to the Movie Lens data sets for various use cases. It has consistently shown superior performance over the existing approach to establish its effectiveness. For instance, on the subset of ML-25ml data set, where each user has less than 20 ratings, GAHF and GSHF show a 18% and 23% lower MAE respectively than that of item-item collaborative filtering. Furthermore, the current existing approaches fail to expand users' current current preference. The proposed method can help an online streaming service to grow by expanding users' interests. Additionally, we have plugged in a similarity metric that is able to capture the nonlinear relationship between genres and reported the superiority in effectiveness.

As part of future work, we aim to apply our approach to various domains within the product rating platform. For example, we can implement this algorithm for online retail stores with product categories where users rate the products online. Furthermore, we can include additional reliable content on the movie or users such as demographic data to enhance the Genre based Collaborative Filtering model.

Declarations

Funding: None
Conflicts of interest/Competing interests: None
Availability of data and material: Data set investigated is freely available
Code availability: Code will be made available once the paper is accepted

References

1. Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

2. Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*, (6):734–749, 2005.
3. Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.
4. Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.
5. Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, 2014.
6. Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.
7. Daniel Billsus and Michael J Pazzani. Learning collaborative information filters. In *Icml*, volume 98, pages 46–54, 1998.
8. Guibing Guo, Jie Zhang, and Daniel Thalmann. Merging trust in collaborative filtering to alleviate data sparsity and cold start. *Knowledge-Based Systems*, 57:57–68, 2014.
9. Rinske Eisma. Attitudes towards dutch subtitling and dubbing.
10. Charu C Aggarwal. An introduction to recommender systems. In *Recommender systems*, pages 1–28. Springer, 2016.
11. Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction*, 25(5):427–491, 2015.
12. Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
13. Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science, 2000.
14. Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19, 2015.
15. Inc Netflix. The netflix prize, 2013.
16. Dheeraj kumar Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. Role of matrix factorization model in collaborative filtering algorithm: A survey. *CoRR, abs/1503.07475*, 2015.
17. Anunay Gupta, Om Prakash Yadav, Arighna Roy, Douglas DeVoto, and Joshua Major. Degradation modeling and reliability assessment of capacitors. In *International Electronic Packaging Technical Conference and Exhibition*, volume 59322, page V001T06A018. American Society of Mechanical Engineers, 2019.
18. Ameneh Forouzandeh Shahraki, Arighna Roy, Om Prakash Yadav, and Ajay Pal Singh Rathore. Predicting remaining useful life based on instance-based learning. In *2019 Annual Reliability and Maintainability Symposium (RAMS)*, pages 1–6. IEEE, 2019.
19. Abhishek Srivastava, Pradip Kumar Bala, and Bipul Kumar. New perspectives on gray sheep behavior in e-commerce recommendations. *Journal of Retailing and Consumer Services*, 53, 2020.
20. Anna Zatevakhina, Natalia Dedyukhina, and Oleg Klioutchnikov. Recommender systems-the foundation of an intelligent financial platform: Prospects of development. In *2019 International Conference on Artificial Intelligence: Applications and Innovations (IC-AIAI)*, pages 104–1046. IEEE.
21. Quanming Yao, Xiangning Chen, James T Kwok, Yong Li, and Cho-Jui Hsieh. Efficient neural interaction function search for collaborative filtering. In *Proceedings of The Web Conference 2020*, pages 1660–1670, 2020.
22. Sang-Min Choi and Yo-Sub Han. A content recommendation system based on category correlations. In *2010 Fifth International Multi-conference on Computing in the Global Information Technology*, pages 66–70. IEEE, 2010.
23. Yashar Deldjoo, Mehdi Elahi, and Paolo Cremonesi. Using visual features and latent factors for movie recommendation. CEUR-WS, 2016.
24. Yashar Deldjoo, Mehdi Elahi, Massimo Quadrana, and Paolo Cremonesi. Toward building a content-based video recommendation system based on low-level features. In *International Conference on Electronic Commerce and Web Technologies*, pages 45–56. Springer, 2015.

25. Michael Fleischman and Eduard Hovy. Recommendations without user preferences: a natural language processing approach. In *IUI*, volume 3, pages 242–244. Citeseer, 2003.
26. Mark Van Setten. Experiments with a recommendation technique that learns category interests. In *ICWI*, pages 722–725, 2002.
27. Chumki Basu, Haym Hirsh, William Cohen, et al. Recommendation as classification: Using social and content-based information in recommendation. In *Aaai/iaai*, pages 714–720, 1998.
28. Sakshi Bansal, Chetna Gupta, and Anuja Arora. User tweets based genre prediction and movie recommendation using lsi and svd. In *2016 Ninth International Conference on Contemporary Computing (IC3)*, pages 1–6. IEEE, 2016.
29. Sang-Ki Ko, Sang-Min Choi, Hae-Sung Eom, Jeong-Won Cha, Hyunchul Cho, Laehyum Kim, and Yo-Sub Han. A smart movie recommendation system. In *Symposium on Human Interface*, pages 558–566. Springer, 2011.
30. Saúl Vargas, Linas Baltrunas, Alexandros Karatzoglou, and Pablo Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 209–216. ACM, 2014.
31. Shaghayegh Sahebi and William W Cohen. Community-based recommendations: a solution to the cold start problem. In *Workshop on recommender systems and the social web, RSWEB*, page 60, 2011.
32. R Kiran, Pradeep Kumar, and Bharat Bhasker. Dnnrec: A novel deep learning based hybrid recommender system. *Expert Systems with Applications*, 144:113054, 2020.
33. P. M. Henriques and J. Mendes-Moreira. Combining recommendation systems with a dynamic weighted technique. In *2016 Eleventh International Conference on Digital Information Management (ICDIM)*, pages 203–208, Sep. 2016.
34. Alexander Felfernig, Seda Polat-Erdeniz, Christoph Uran, Stefan Reiterer, Muesluem Atas, Thi Ngoc Trang Tran, Paolo Azzoni, Csaba Kiraly, and Koustabh Dolui. An overview of recommender systems in the internet of things. *Journal of Intelligent Information Systems*, 52(2):285–309, 2019.
35. Vahideh Nobahari, Mehrdad Jalali, and Seyyed Javad Seyyed Mahdavi. Isotrustseq: a social recommender system based on implicit interest, trust and sequential behaviors of users using matrix factorization. *Journal of Intelligent Information Systems*, 52(2):239–268, 2019.
36. Jyoti Shokeen and Chhavi Rana. Social recommender systems: techniques, domains, metrics, datasets and future scope. *Journal of Intelligent Information Systems*, pages 1–35, 2019.
37. Liangmin Guo, Jiakun Liang, Ying Zhu, Yonglong Luo, Liping Sun, and Xiaoyao Zheng. Collaborative filtering recommendation based on trust and emotion. *Journal of Intelligent Information Systems*, 53(1):113–135, 2019.
38. Arighna Roy and Anne Denton. Nearest-neighbor-intersection algorithm for identifying strong predictors using high-dimensional data. In *2019 IEEE International Conference on Electro Information Technology (EIT)*, pages 416–421. IEEE, 2019.
39. Anne M Denton and Arighna Roy. Cluster-overlap algorithm for assessing preprocessing choices in environmental sustainability. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 4212–4220. IEEE, 2017.
40. F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.