Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 9, No. 2, May 2024, Pp.159-172

Movie recommender system on twitter using weighted hybrid filtering and GRU

Nico Valentino*[1], Erwin Budi Setiawan[1]
Faculty of Informatics, Telkom University, Indonesia[1]

Article Info
Keywords:
Recommender System, Hybrid Filtering, GRU, Twitter, Netflix, Disney+

Article history:
Received: December 27, 2023
Accepted: March 13, 2024
Published: May 31, 2024

Cite:
N. Valentino and E. B. Setiawan, "Movie Recommender System on Twitter Using Weighted Hybrid Filtering and GRU", KINETIK, vol. 9, no. 2, May 2024. Retrieved from https://kinetik.umm.ac.id/index.php/kinetik/article/view/1941

*Corresponding author.
Nico Valentino
E-mail address:
nicvalentino@student.telkomuniversity.ac.id

Abstract
The development of the industry in the film sector has experienced rapid growth, marked by the emergence of film streaming platforms such as Netflix and Disney+. With the abundance of available films, users face difficulty in choosing films that suit their preferences. Recommender systems can be a solution to this problem for users. Recommender systems rely on user reviews, making Twitter a platform that can be used to collect user reviews of a film. This study will develop a recommender system that has the potential to provide item recommendations to users using the weighted hybrid filtering and GRU methods. The weighted hybrid filtering used is a combination of collaborative filtering and content-based filtering methods. The dataset used in this study was obtained by crawling tweets relevant to the feedback of specific accounts regarding a film. The dataset resulting from the data crawling consists of a total of 854 films, 45 users and 34,086 tweets consisting of film reviews from Twitter users. The GRU model classification is performed on the results of weighted hybrid filtering with model optimization involving testing various test size scenarios and optimizer methods. The test sizes used are 40%, 30%, and 20%. The optimizer methods used include Adam, Nadam, Adamax, Adadelta, Adagrad, and SGD. The research results show that the optimal outcome is obtained using the Nadam optimization method. The performance evaluation yielded 85.74% precision, 88.63% recall, 88.63% accuracy, and 86.30% F1-score.

1. Introduction

Over the past few years, the film industry has witnessed remarkable growth, largely driven by the proliferation of streaming platforms. These platforms, such as Netflix and Disney+, have made films easily accessible to a wide range of audiences, offering a vast catalog of titles across diverse genres. However, this abundance of choice can create a dilemma for users, who may struggle to navigate the extensive content libraries and select films that align with their preferences. To address this challenge, there's a pressing need for solutions that can provide users with tailored film recommendations.

In addition to the ease of watching movies, people now also have easy access to various social media platforms. Twitter is a particularly popular platform that allows users to share and exchange opinions on a wide range of topics. Due to its 280-character limit and global accessibility, Twitter data is easily collected and analyzed, making it a valuable resource for research [1].

Recommender systems are tools that filter information and direct it to users based on their interests and relevance [2]. Recommender systems are now widely used by companies to address the challenges of information overload and to help users make informed decisions [3]. There are many techniques that can be used to build a recommender system. Collaborative filtering (CF) and content-based filtering (CBF) are two of the most popular techniques. CF provides item recommendations based on the similarity of characteristics between users and items that those users have rated highly, while CBF provides item recommendations based on the characteristics of items that a user desires [3], [4]. However, both techniques have their weaknesses, and methods are needed to overcome them. CF has two common problems, namely data sparsity and cold start. Data sparsity is the lack of data because many users only rate a small number of items. This can lead to many empty matrices in the item rating matrix, as users who have not rated an item will cause the corresponding cell in the matrix to be empty [5], [6]. On the other hand, cold start is a condition in which the recommender system lacks historical user data or there are new users who have not yet provided ratings [5]. This can lead to a decrease in the quality of the recommender system. The main weakness of CBF is that it can only recommend items that are similar to those that the user has already rated highly. This can limit the user's exposure to new and

different items. Weighted hybrid filtering (WHF) can overcome these weaknesses because WHF can improve the quality of a recommender system and cover the weaknesses of other models by taking advantage of the strengths of each combined model [7], [8].

One of the previous studies related to recommender systems was conducted in a study titled "Movie Recommender System Using Collaborative Filtering" [9]. In that study, collaborative filtering (CF) and content-based filtering (CBF) were used to build a movie recommender system. Both methods were implemented and combined with the K-Nearest Neighbor (KNN) algorithm. The results of the evaluation, based on the Mean Absolute Error (MAE) value, showed that CF combined with KNN produced results in the range of 0.248 to 0.265, while CBF combined with KNN produced a result of 0.269.

Another study that has been conducted is entitled "Dynamic Weighted Hybrid Recommender Systems" [8]. In that study, a recommender system was built using the dynamic weighted hybrid filtering method. The study combined the CF and CBF methods based on predetermined weights. Then, the results of the recommender system that was created were evaluated based on the Root Mean Squared Error (RMSE) value. The performance of dynamic weighted hybrid filtering in the RMSE test achieved results in the range of 0.946 - 0.965.

This research proposes a movie recommender system that uses the WHF method combined with the Gated Recurrent Unit (GRU) algorithm as a solution to the problems experienced by Netflix and Disney+ users. GRU is used as a classification method to improve performance and accuracy in rating prediction. To the best of our knowledge, no research has combined WHF with GRU as a classification method to build a recommender system. The WHF method is built by combining the CF and CBF methods. The motivation of this research is to improve accuracy in recommender systems by leveraging the combination of the two methods. By combining WHF and GRU, we hoped that the resulting recommender system can provide more accurate recommendations to users according to their individual preferences.

The rest of the paper is structured as follows. Section 2 describes the research method. Section 3 shows the results and research of the conducted research. Section 4 presents the conclusion of this research.

2. Research Method

2.1 Recommender System Design

The recommender system in this study was built in accordance with the design shown in Figure 1.

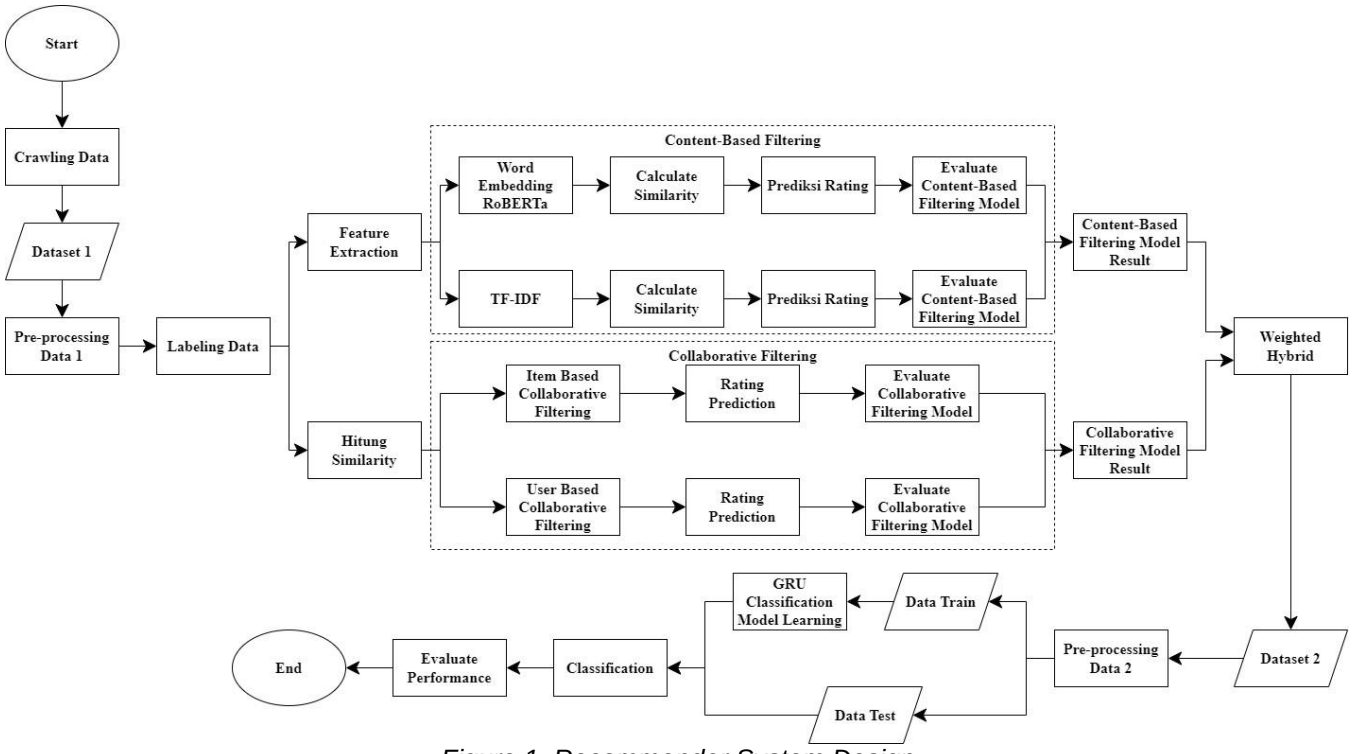

Figure 1. Recommender System Design

The system design shown in Figure 1 can be divided into five stages: crawling data, collaborative filtering (CF), content-based filtering (CBF), weighted hybrid filtering (WHF), and GRU model classification. Section 2 will explain each process in the design.

2.2 Data Crawling

This study used two datasets obtained from different sources. The first dataset contained Netflix and Disney+ movies obtained from IMDB using a search filter. The movie dataset were then extracted using the PyMovieDb Python library, which added features such as 'description', 'keyword', etc. Table 1 shows some movies after feature extraction.

Table 1. Example of Film Crawling Result

Film	Genre	Date Published
14 Cameras	[“Crime”, “Horror”, “Thriller”]	2018-07-27
17 Again	[“Comedy”, “Drama”, “Fantasy”]	2009-04-17
1BR	[“Drama”, “Horror”, “Thriller”]	2020-04-24

The second dataset used in this study is a dataset of tweets from Twitter users who are known to be experts in film reviewing. Table 2 shows the results of collecting tweets using Twitter-Harvest.

Table 2. Example of Tweet Crawling Result

Username	Text
djaycoholyc	Grown Ups pertama ini masih asyik. Asyik banget
CenayangFilm	Film biografi indonesia yg paling pas emang baru azrax sih. Baru Gie dan Habibie Ainun.
danieldokter	Buset. Don't Knock Twice full house. Apa-apaan ini...

2.3 Preprocessing Data

The preprocessing process focused on converting film review data obtained from Twitter into rating data with a range of values from 0 to 5. This conversion process consisted of three stages: translation, text cleaning, and polarity scoring.

In the translation stage, each film review was translated into English using the Python Deep Translator Library. Then, the text cleaning stage was carried out to remove irrelevant information from the film review data. Polarity scoring used the TextBlob library in Python to evaluate film review data on a scale of 0 to 5. Table 3 shows an example of the results of preprocessing data against the film review dataset.

Table 3. Example of Preprocessing Data Result

Username	Film	Score
AnakNonton	3 Days to Kill	2.84
BFI	Paradise	2.85
CenayangFilm	Tanda Tanya	2.43

2.4 Recommender System

A recommender system is a system that filters information and directs it to a specific user based on the user's interests and relevance [2]. Recommender systems are designed to help users choose from a wide range of available items [10]. Recommender systems can provide item recommendations by modeling and analyzing user behavior [11]. Recommender systems are now widely used by platforms that have large amounts of data to offer to their users.

There are several methods that can be used to build a recommender system. CF and CBF are two of the most popular methods. WHF is another method that can be built by combining CF and CBF. WHF can improve the accuracy and variety of recommendations by taking advantage of the strengths of each method [12].

2.5 Weighted Hybrid Filtering

Hybrid filtering (HF) is one of the methods that can be used to build a recommender system. HF is built by combining more than one filtering model [7]. HF aims to improve the quality of a recommender system and overcome the weaknesses of other models by leveraging the strengths of each combined model [7], [8].

There are several approaches that can be used to build a recommender system using the HF method. WHF is one of the HF approaches that is implemented by combining the scores of several recommender system methods based on the weight of each combined method [8].

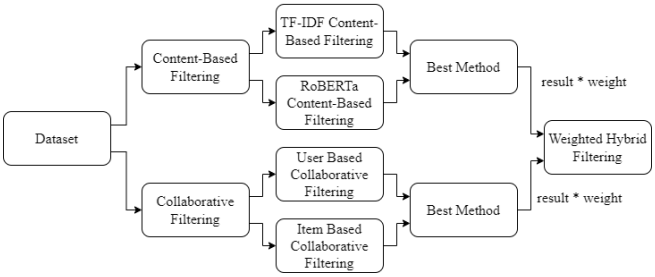

Figure 2. Weighted Hybrid Filtering Implementation

Figure 2 shows the details of the merging process of the two methods, CBF and CF. The best method in CBF and CF was selected based on the smallest RMSE in each method. Then, the best method of CF and CBF were combined based on the predetermined weight. The weight of each method is determined based on the RMSE and relevance of each method in providing recommendations that are in line with the user's preferences. By determining the weight based on RMSE, it can improve the performance and accuracy of WHF. In addition, WHF can also become more adaptive in merging multiple methods.

$$Weight = \left(\frac{RMSE}{\sum_{i=1}^{n} RMSE_i}\right)^{-1} \quad (1)$$

Equation 1 shows the equation for calculating the weights used in the CF and CBF methods. The weight of a method is calculated by taking the inverse of the RMSE of that method, divided by the total RMSE of all methods. Therefore, the method with the lower RMSE will receive a higher weight.

2.6 Gated Recurrent Unit

Gated recurrent unit, also known as GRU, is a type of recurrent neural network architecture developed in 2014 by Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio [13]. GRU was developed to address the problem of vanishing gradients in long-term memory [13]. GRU is a simplified version of Long Short-Term Memory (LSTM) with improved performance [14].

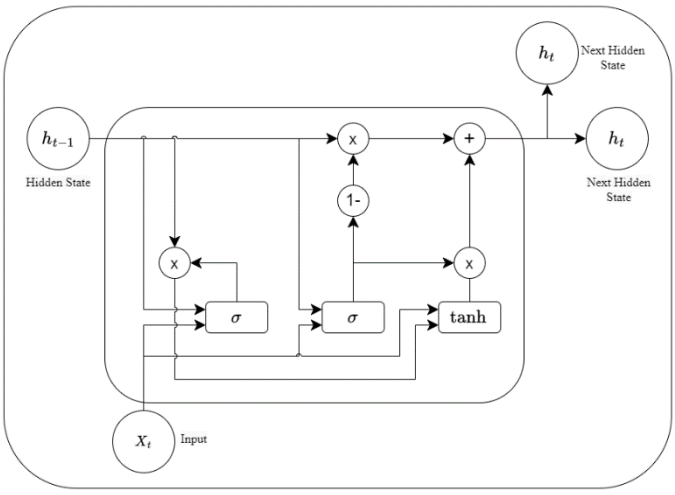

Figure 3. GRU Illustration

The process that occurs in the GRU model is illustrated in Figure 3. The GRU model, like any artificial neural network, consists of several hidden layers, or memory cells. The memory cell in GRU is used to store data that has been processed and will be used as a reference when there is new input data. Each memory cell contains two gates: the reset gate and the update gate [13]. The reset gate controls which data in the memory cell will be erased or forgotten. This is done to ensure that the memory cell only retains relevant data. The update gate controls how much weight is given to new input data. If the value of the update gate is close to 1, then the new data will have a large impact on the memory cell. Conversely, if the value of the update gate is close to 0, then the new data will have a small impact on the memory cell.

2.7 Twitter

Twitter is a social media platform that allows users to share and exchange opinions on various issues in the form of text and images with a maximum of 280 characters. Currently, Twitter can be accessed by anyone via the web or mobile devices, so Twitter has a lot of user data related to various issues discussed by other Twitter users. Therefore, Twitter has a large amount of data and makes it a good option to be raised as research material [13].

2.8 Memory Based Collaborative Filtering

Memory-based collaborative filtering is a recommender system technique that predicts ratings by leveraging user-item feedback [15], [16]. This technique can predict ratings by relying on the similarity between users or the similarity between items to produce accurate rating predictions [15]. There are two approaches that are generally used in memory-based collaborative filtering, namely user-based and item-based.

2.8.1 User Based Collaborative Filtering

User-based collaborative filtering is one of the techniques of memory-based collaborative filtering that predicts ratings by identifying similar users to other users, and provides item recommendations based on what similar users like [9], [17].

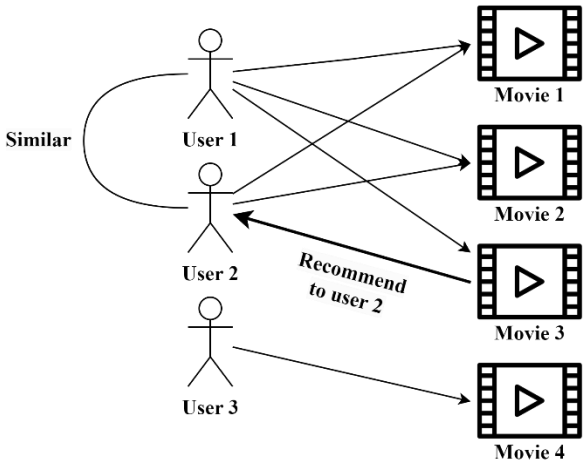

Figure 4. User Based Collaborative Filtering Illustration

Figure 4 shows that user 1 and user 2 are identified as similar because they both like movie 1 and movie 2. Therefore, user 2 will be recommended movie 3 because movie 3 is liked by a user who is similar to user 2. The user-based collaborative filtering rating prediction in this study will be calculated using Equation 2.

$$R(u,i) = \frac{\sum_{n=1}^{TotalUser}\left(R(n,i) \times S(u,n)\right)}{\sum_{n=1}^{TotalUser} S(u,n))} \quad (2)$$

In Equation 2, R(u,i) is the rating value of user u for item i, while S(u,n) is the similarity value between user u and user n. The rating of user u for item i will be calculated based on the total sum of the ratings of other users for item i multiplied by the similarity of the user with user u. After that, the total sum of the ratings will be divided by the total sum of the similarity of user u with other users so that the resulting numbers have the same interval.

2.8.2 Item Based Collaborative Filtering

Item-based collaborative filtering is a technique of memory-based collaborative filtering that predicts ratings by identifying items that are similar to items that have been rated highly or interacted with the target user [17], [18].

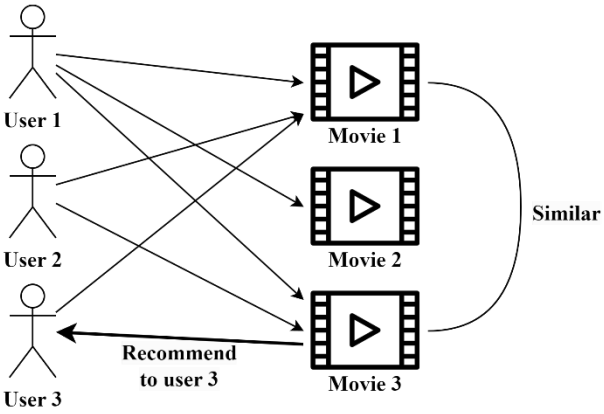

Figure 5. Item Based Collaborative Filtering Illustration

Figure 5 shows that users 1 and 2 both rated movies 1 and 3 highly, so it is identified that these two movies are similar. On the other hand, user 3 rated movie 1 highly, but has not yet rated movie 3. Because movies 1 and 3 are identified as similar, movie 3 is recommended to user 3. The prediction of the rating of item-based collaborative filtering in this study will be calculated using Equation 3.

$$R(u,i) = \frac{\sum_{n=1}^{TotalItem}(R(u,n) \times S(n,i))}{\sum_{n=1}^{TotalItem} S(n,i))} \quad (3)$$

In Equation 3, R(u,i) is the rating of user u for item i, while S(n,i) is the similarity value between item n and item i. The rating of user u for item i will be calculated based on the total sum of the ratings of other items by user u multiplied by the similarity of the item with item i. After that, the total sum of the ratings will be divided by the total sum of the similarity of item i with other items so that the resulting numbers have the same interval.

2.9 Content-Based Filtering

CBF is an approach in a recommender system that utilizes the attributes of an item and user preferences [19], [20]. This approach will provide item recommendations by identifying other items that are similar to the target user's preferences. Item attributes that can be used for this identification include item descriptions, item names, item keywords, etc.

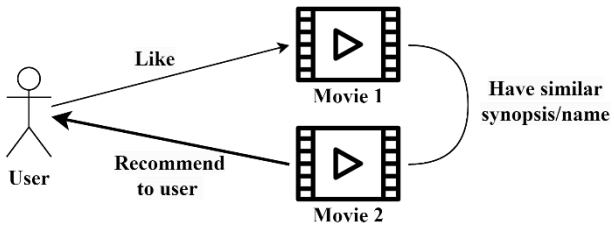

Figure 6. Content-Based Filtering Illustration

Figure 6 shows that the user likes movie 1. Movie 1 and movie 2 are identified as similar films because they have similar attributes. Since the user has not interacted with movie 2, movie 2 will be recommended to the user.

2.10 Cosine Similarity

Cosine similarity is a method for calculating the similarity between two vectors of the same dimension [21]. Cosine similarity is a method for calculating the similarity between two vectors of the same dimension. This method measures the extent to which the two vectors are aligned in the same direction. The smaller the angle between the two vectors, the greater the similarity between the two vectors [22].

$$CosineSimilarity(A,B) = \frac{A \,.\, B}{||A|| \,.\, ||B||} \quad (4)$$

Equation 4 is a commonly used formula for calculating cosine similarity. In this study, the sklearn library in the Python programming language was used, resulting in a cosine similarity value interval of 0 to 1.

2.11 Feature Extraction

Feature extraction is a method for extracting relevant information or important features from the original dataset [23], [24]. The purpose of feature extraction is to reduce unimportant information, reduce data dimensions and improve the efficiency of data analysis without losing important information from the dataset [23]. Feature extraction is used so that the CBF model can work more efficiently and effectively in identifying similarity between item attributes. In this study, two feature extraction methods were used, namely Term Frequency-Inverse Document Frequency (TF-IDF) and word embedding using the Robustly optimized BERT approach (RoBERTa).

2.11.1 TF-IDF

TF-IDF as stated in Equation 5, Equation 6, and Equation 7 is a text processing method used to measure the importance of a word in a text [25], [26].

$$TF(t,d) = \frac{Frequency\ of\ term\ t\ in\ dokumen\ d}{The\ number\ of\ terms\ in\ document\ d} \tag{5}$$

$$IDF(t,D) = log\ (\frac{The\ number\ of\ documents\ in\ D}{Number\ of\ documents\ with\ term\ t}) \tag{6}$$

$$TF - IDF(t,d,D) = TF(t,d) \times IDF(t,D) \tag{7}$$

Term frequency (TF) in TF-IDF refers to the occurrence of certain words in a document. The higher the TF value of a word, the more important that word is in a document [25]. Inverse document frequency (IDF) in TF-IDF refers to the inverse of how often a word appears in a document. The more often a word appears in a document, the less important that word is in that document [25].

2.11.2 Word Embedding RoBERTa

Word embedding is one of the methods in natural language processing (NLP). Word embedding is used to convert words into the form of numerical vectors [27]. The purpose of word embedding is to obtain semantic and syntactic relationships between words, so that words that are similar in a certain context have vector representations that are close to each other. RoBERTa is an improvisation of BERT that uses word embedding with a contextualized word embedding approach [28]. This study used the RobertaTokenizer from the transformers library in the Python programming language.

2.12 Performance Evaluation

In this study, several performance testing techniques were used to measure the quality of the program that was produced. Precision, recall, accuracy, and F1-score were used through the confusion matrix table to test the performance of classification. RMSE was used to test the performance of WHF rating prediction by determining the difference between the predicted value and the actual value.

2.12.1 Confusion Matrix

Confusion matrix is a Table 4 that is used to evaluate the performance of a classification model. This table provides information about the number of correct and incorrect predictions made by a model.

Table 4. Confusion matrix

	Predicted positive	Predicted negative
Actual positive	True Positive (TP)	False Negative (FN)
Actual negative	False Positive (FP)	True Negative (TN)

Confusion matrix can be used to calculate several performance evaluation techniques that was used in this study, namely precision, recall, accuracy, and F1-score. Precision is a metric that measures how accurately a model identifies positives from all predicted positives [9]. Recall is a metric that measures how many positives are identified by the model from all the actual positives [9]. Accuracy is a measurement of how accurately a model can identify all categories

correctly [9]. F1-Score is the harmonic mean of precision and recall that identifies the balance between precision and recall [9]. The performance evaluation used in this study is stated in Equation 8 – Equation 11, respectively as precision, recall, accuracy and F1 -score.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{10}$$

$$F1 - Score = \frac{2 \times precision \times recall}{precision + recall} \tag{11}$$

2.12.2 Root Mean Square Error

RMSE is one of the performance evaluation methods based on the difference between the predicted value and the observed or measured value. The RMSE as depicted in Equation 12 is generally used in the context of evaluating statistical models or predictive models to measure how close the model's predictions are to the actual value.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(predicted_i - actual_i)^2} \tag{12}$$

3. Results and Discussion

This research is divided into 5 stages, namely data preparation, CF, CBF, WHF, and GRU model classification. The data preparation stage includes crawling data from Twitter, preprocessing data, and labeling data before being used in the CF and CBF stages. In the CF stage, data that has been prepared previously is used. The purpose of this stage is to predict ratings based on user-item interaction data. In the CBF stage, data that has been prepared previously is also used, and it predicts ratings based on the similarity of content between items. After the CF and CBF stages were completed in parallel, the output of these two stages were combined in the WHF stage. In the WHF stage, the output of CF and CBF were combined based on weights determined through the RMSE of the two methods. The last stage is the GRU model classification that involves output of the WHF rating prediction stage. The GRU model classification was evaluated using a confusion matrix. The details of each test and analysis can be seen in the sub-paragraphs in section 3.

3.1 Data Preparation

The data preparation stage produces two datasets: the film dataset and the film review dataset. The film dataset, obtained from IMDB, produces a dataset of 854 films with 15 features after feature extraction using PyMovieDb as shown in Table 5.

Table 5. Film Crawling Result

Film	Genre	…	Date Published	Duration
14 Cameras	[“Crime”, “Horror”, “Thriller”]	…	2018-07-27	PT1H30M
17 Again	[“Comedy”, “Drama”, “Fantasy”]	…	2009-04-17	PT1H42M
…	…	…	…	…
3 Days to Kill	[“Action”, “Comedy”, “Drama”]	…	2014-02-25	PT1H57M
3 Idiots	[“Comedy”, “Drama”]	…	2009-12-25	PT2H50M

The film review dataset was obtained by crawling tweets on Twitter using Twitter-Harvest. Tweet crawling was performed on 39 Twitter accounts that are experts in film reviewing. The crawling results obtained a total of 34,086 tweets in the form of film reviews with 3 features as shown in Table 6.

Table 6. Tweet Crawling Result

Username	Film	Text
AnakNonton	Thor: Ragnarok	Dengan $121 juta, 'Thor: Ragnarok' jadi film MCU dgn debut terbesar ke-7 sekaligus memuncaki box-office minggu ini!
AnakNonton	Headshot	Penata Efek Visual Terbaik #FFI2016 : Andi Novianto - 'Headshot' #MalamPuncakFFI2016
…	…	…
zavvi	Turning Red	Disney Pixar's #TurningRed hits @disneyplus today! Who else is watching this cute and cuddly coming of age film?
zavvi	What If	Well, new animated Marvel show What If...? looks like it will be plenty of fun #DisneyInvestorDay

Preprocessing was then performed on the film review dataset. The focus of the preprocessing was to convert film reviews to a scale of 0 to 5. The results of the translation, text cleaning, and polarity scoring processes are shown in Table 7.

Table 7. Film Reviews Preprocessing Data Result

Username	Film	Score
AnakNonton	3 Days to Kill	2.84
AnakNonton	65	2.89
…	…	…
zavvi	What If	3.26
zavvi	You People	2.68

The film review dataset and the film dataset were then combined to form a dataset of size 854×45, with 854 indicating the number of films and 45 indicating the number of users observed. The dataset still contains many user-item interactions with a value of 0. This condition is commonly known as 'data sparsity'. A dataset can be said to experience data sparsity if most of the values in the matrix are still empty [4]. Sparsity ratio dari dataset adalah 80,9%. After that, films and users with fewer than 5 interactions will be deleted to reduce the sparsity ratio and improve the quality of the model that is built. In addition, normalization is also performed on the dataset so that the ratings on a scale of 0 to 5 are converted to a scale of 0 to 1. The results show that the dataset has a sparsity ratio of 73.3%, and the size of the dataset is reduced to 552×44, with 552 indicating the number of movies and 44 indicating the number of users observed. The final dataset that will be used in CF and CBF is shown in Table 8.

Table 8. User-item Interaction Dataset

Film name	Elbert_Reyner	IMDB	…	zavvi
14 Cameras	0	0.479167	…	0
17 Again	0.45	0.666667	…	0
…	…	…	…	…
Zombieland	0.644	0.791667	…	0

3.2 Collaborative Filtering

In the CF stage, rating predictions are made based on cosine similarity using two approaches: user based CF and item based CF. User based CF rating predictions are calculated based on the similarity between users, the results of which are shown in Table 9.

Table 9. User Based Collaborative Filtering Result

Film name	Elbert_Reyner	IMDB	…	zavvi
14 Cameras	0.276921	0.479167	…	0.233445
17 Again	0.45	0.666667	…	0.468272
…	…	…	…	…
Zombieland	0.644	0.791667	…	0.591778

On the other hand, item based CF rating predictions are calculated based on the similarity between items, the results of which are shown in Table 10.

Table 10. Item Based Collaborative Filtering Result

Film name	Elbert_Reyner	IMDB	…	zavvi
14 Cameras	0.541324	0.479167	…	0.6073
17 Again	0.45	0.666667	…	0.615652
…	…	…	…	…
Zombieland	0.644	0.791667	…	0.62099

From the results of the two methods, one result is selected as the best CF method to be combined in the WHF stage. The best method is determined based on the best RMSE performance test. The results of the RMSE test are shown in Figure 7.

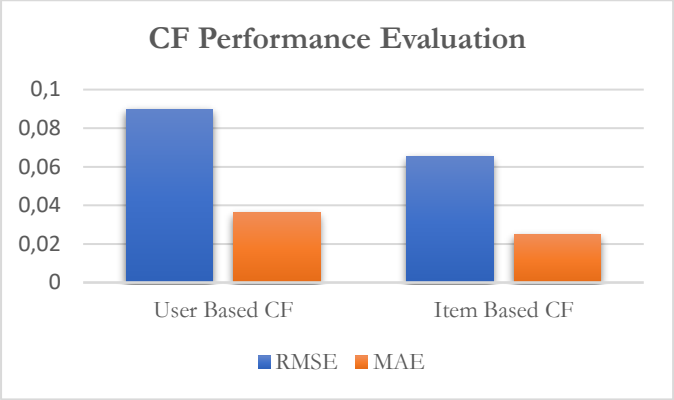

Figure 7. Collaborative Filtering Performance Comparison

The results of the RMSE evaluation show that item-based CF performs 37% better than user-based CF in rating predictions. Therefore, the CF stage uses item-based CF as the best method for the WHF stage.

3.3 Content-Based Filtering

In the CBF stage, rating predictions are made based on the similarity of the content of items. The features used to calculate similarity are keywords, description, and genre. The three features were then extracted using two different language processing methods: TF-IDF and Word Embedding RoBERTa. The results of the CBF rating predictions using TF-IDF are shown in Table 11, while the results of the CBF rating prediction using Word Embedding RoBERTa are shown in Table 12.

Table 11. Content-based Filtering using TF-IDF Result

Film name	Elbert_Reyner	IMDB	…	zavvi
14 Cameras	0.173328	0.479167	…	0
17 Again	0.45	0.666667	…	0
…	…	…	…	…
Zombieland	0.644	0.791667	…	0

Table 12. Content-based Filtering using RoBERTa Result

Film name	Elbert_Reyner	IMDB	…	zavvi
14 Cameras	0.179101	0.479167	…	0.045947
17 Again	0.45	0.666667	…	0.045987
…	…	…	…	…
Zombieland	0.644	0.791667	…	0.046063

From the results of both language processing, one result is taken as the best method representing CBF to be combined in the WHF stage. The best method is determined based on the testing of the best RMSE performance. The results of the RMSE evaluation are shown in Figure 8.

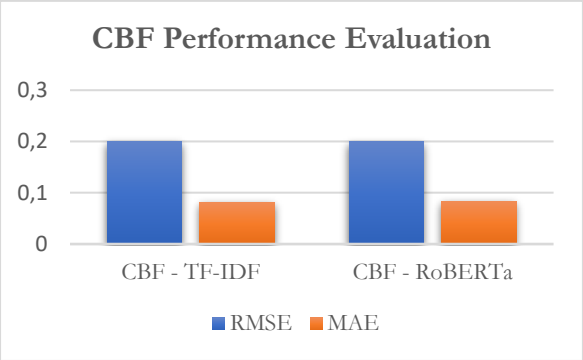

Figure 8. Content-based Filtering Performance Comparison

The results of the RMSE evaluation showed that CBF with RoBERTa resulted in slightly more accurate performance than CBF with TF-IDF in predicting ratings. Therefore, the CBF stage uses CBF with RoBERTa as the best method to be used in the WHF stage.

3.4 Weighted Hybrid Filtering

In the WHF stage, the output of CF and CBF were combined with weights that were determined based on RMSE. The weights are calculated using Equation 1, resulting in the weights of CF and CBF that can be seen in Table 13.

Table 13. Weight for Each Weighted Hybrid Filtering Method

Method	Weight
CF	0.75348
CBF	0.24652

Next, each user-item interaction in the two methods was multiplied by their respective weights, and then summed. The result of this sum is the output of WHF. The final result of WHF can be seen in Table 14.

Table 14. Weighted Hybrid Filtering Result

Film name	Elbert_Reyner	IMDB	…	zavvi
14 Cameras	0.45203	0.479167	…	0.468917
17 Again	0.45	0.666667	…	0.475221
…	…	…	…	…
Zombieland	0.644	0.791667	…	0.479261

3.5 GRU Classification Model

The GRU model classification stage uses dataset 2, which is the output of the WHF stage. In this stage, each user-item interaction is assigned to one of two classes: class 0 or class 1. Class 0 represents not recommended items, while class 1 represents recommended items. The class assignment is based on the average value of all user-item interactions. Interactions with values above the average are assigned to class 1, while interactions with values below the average are assigned to class 0.

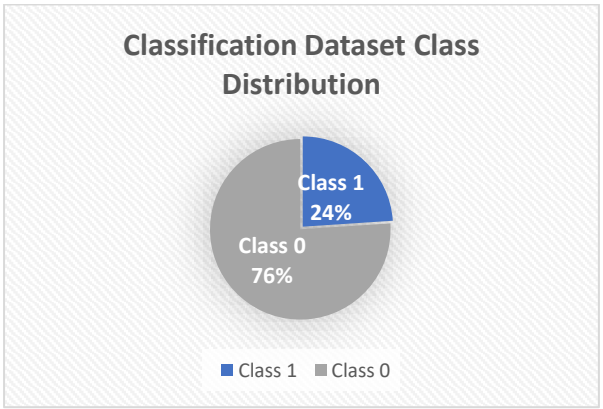

Figure 9. Classification Dataset Class Distribution

Figure 9 shows that the class distribution in the classification dataset after the values were adjusted results in 24% of the data classified as class 1 and 76% of the data classified as class 0. After that, three testing scenarios were conducted with different test sizes: 40%, 30%, and 20%. A comparison of the performance evaluation of each test size testing is shown in Table 15.

Table 15. Performance Metrics of Baseline Model

Test Size	Performance Metrics (%)			
	Precision	*Recall*	*Accuracy*	*F1-Score*
40%	85.38%	88.53%	**88.53%**	85.98%
30%	84.83%	88.02%	88.02%	85.48%
20%	86.08%	88.08%	88.08%	86.23%

Based on Table 15, the highest accuracy was obtained at 88.53% when testing with a test size of 40%. Testing with a test size of 40% also produced good results in other performance evaluations, with precision, recall, and f1-score values of 85.38%, 88.53%, and 85.98%, respectively.

To improve the performance of test size testing results, we conducted tests using six different optimizers. The goal was to find the optimal combination of methods. The optimizers used were Adam, Nadam, Adamax, Adadelta, Adagrad, and SGD. Each optimizer was implemented using standard parameters and learning rate with a test size of 40%. The performance evaluation comparison of each optimizer test is shown in Table 16.

Table 16. Performance Metrics of Each Optimizer

Optimizer	Performance Metrics (%)			
	Precision	Recall	Accuracy	F1-Score
Baseline	85.38%	88.53%	88.53%	85.98%
Adam	84.97%	88.39%	88.39%	85.75%
Nadam	**85.74%**	**88.63%**	**88.63%**	**86.30%**
Adamax	79.77%	88.13%	88.13%	83.07%
Adadelta	71.36%	39.49%	39.49%	38.14%
Adagrad	80.55%	86.98%	86.98%	82.77%
SGD	77.54%	87.84%	87.84%	82.27%

Table 16 shows that the Nadam optimizer produces the best performance improvement. The performance of Nadam reaches precision of 85.74%, recall of 88.63%, accuracy of 88.63%, and f1-score of 86.30%. This shows that the performance produced when using the Nadam optimizer is consistently better than the Baseline in all performance metrics.

Testing was conducted on providing recommendations to specific users using an optimized recommender system. The optimal recommendation system used in the testing employs a test size of 40% with the Nadam optimizer. The recommendation provision was tested on the username 'zavvi'. The test results are shown in Table 17.

Table 17. Optimized Recommender System Test Result

No.	Film Name
1.	Ballerina
2.	65
3.	AKA
4.	Leo
5.	Turning Red
6.	Paradise
7.	Hunger
8.	Troll
9.	Noise
10.	Meter

4. Conclusion

Overall, this study details the analysis of several test size scenarios and optimizer scenarios aimed at optimizing the performance of the GRU model as a classification model. The hope is that this model can be a solution to the issues faced by Netflix and Disney+ users. In baseline classification testing, satisfy actory results were achieved, with the highest performance value of precision 85.38%, recall 88.53%, accuracy 88.53%, and f1-score 85.98% achieved when using a test size of 40%. In addition to baseline testing, testing with several optimizers showed better performance than classification without optimizer or baseline. This is evidenced by the highest performance achieved when using the Nadam optimizer, with a performance value of precision 85.74%, recall 88.63%, accuracy 88.63%, and f1-score 86.30%. This study shows that testing several test size scenarios and testing several optimizer scenarios are necessary to create an optimal model. Although favorable performance metrics have been achieved, future research can develop recommender system optimization by combining other classification models and adjusting the optimal learning rate for each optimizer used to create a recommender system.

References

[1] K. Sailunaz and R. Alhajj, "Emotion and sentiment analysis from Twitter text," *J. Comput. Sci.*, vol. 36, p. 101003, 2019, doi: https://doi.org/10.1016/j.jocs.2019.05.009.

[2] H. Tahmasebi, R. Ravanmehr, and R. Mohamadrezaei, "Social movie recommender system based on deep autoencoder network using Twitter data," *Neural Comput. Appl.*, vol. 33, no. 5, pp. 1607–1623, 2021, doi: https://doi.org/10.1007/s00521-020-05085-1.

[3] G. Geetha, M. Safa, C. Fancy, and D. Saranya, "A Hybrid Approach using Collaborative filtering and Content based Filtering for Recommender System," *J. Phys. Conf. Ser.*, vol. 1000, no. 1, 2018, doi: https://doi.org/10.1088/1742-6596/1000/1/012101.

[4] N. Ifada, T. F. Rahman, and M. K. Sophan, "Comparing collaborative filtering and hybrid based approaches for movie recommendation," *Proceeding - 6th Inf. Technol. Int. Semin. ITIS 2020*, pp. 219–223, 2020, doi: https://doi.org/10.1109/ITIS50118.2020.9321014.

[5] S. Natarajan, S. Vairavasundaram, S. Natarajan, and A. H. Gandomi, "Resolving data sparsity and cold start problem in collaborative filtering recommender system using Linked Open Data," *Expert Syst. Appl.*, vol. 149, 2020, doi: https://doi.org/10.1016/j.eswa.2020.113248.

[6] S. Ahmadian, M. Afsharchi, and M. Meghdadi, "A novel approach based on multi-view reliability measures to alleviate data sparsity in recommender systems," *Multimed. Tools Appl.*, vol. 78, no. 13, pp. 17763–17798, 2019, doi: https://doi.org/10.1007/s11042-018-7079-x.

[7] R. Logesh and V. Subramaniyaswamy, *Exploring hybrid recommender systems for personalized travel applications*, vol. 768. Springer Singapore, 2019, doi: https://doi.org/10.1007/978-981-13-0617-4_52.

[8] H. Q. Do, T. H. Le, and B. Yoon, "Dynamic Weighted Hybrid Recommender Systems," *Int. Conf. Adv. Commun. Technol. ICACT*, vol. 2020, pp. 644–650, 2020, doi: https://doi.org/10.23919/ICACT48636.2020.9061465.

[9] M. Gupta, A. Thakkar, Aashish, V. Gupta, and D. P. S. R. Rathore, "Movie Recommender System Using Collaborative Filtering," no. Icesc, pp. 415–420, 2020, doi: https://doi.org/10.1109/ICESC48915.2020.9155879.

[10] Y. Fu and T. Wang, "Item-based collaborative filtering with BERT," *Proc. Annu. Meet. Assoc. Comput. Linguist.*, vol. 2020-July, no. Ecnlp 3, pp. 54–58, 2020, doi: https://doi.org/10.18653/v1/2020.ecnlp-1.8.

[11] X. Wang, Z. Dai, H. Li, and J. Yang, "A New Collaborative Filtering Recommendation Method Based on Transductive SVM and Active Learning," *Discret. Dyn. Nat. Soc.*, vol. 2020, no. 1, 2020, doi: https://doi.org/10.1155/2020/6480273.

[12] S. Wan and Z. Niu, "A hybrid e-learning recommendation approach based on learners' influence propagation," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 5, pp. 827–840, 2020, doi: https://doi.org/10.1109/TKDE.2019.2895033.

[13] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," pp. 1–9, 2014. https://doi.org/10.48550/arXiv.1412.3555

[14] K. E. ArunKumar, D. V. Kalaga, C. M. S. Kumar, M. Kawaji, and T. M. Brenza, "Forecasting of COVID-19 using deep layer Recurrent Neural Networks (RNNs) with Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) cells," *Chaos, Solitons and Fractals*, vol. 146, p. 110861, 2021, doi: https://doi.org/10.1016/j.chaos.2021.110861.

[15] D. Valcarce, A. Landin, J. Parapar, and Á. Barreiro, "Collaborative filtering embeddings for memory-based recommender systems," *Eng. Appl. Artif. Intell.*, vol. 85, no. May, pp. 347–356, 2019, doi: https://doi.org/10.1016/j.engappai.2019.06.020.

[16] G. R. Lima, C. E. Mello, A. Lyra, and G. Zimbrao, "Applying landmarks to enhance memory-based collaborative filtering," *Inf. Sci. (Ny).*, vol. 513, pp. 412–428, 2020, doi: https://doi.org/10.1016/j.ins.2019.10.041.

[17] C. Ajaegbu, "An optimized item-based collaborative filtering algorithm," *J. Ambient Intell. Humaniz. Comput.*, vol. 12, no. 12, pp. 10629–10636, 2021, doi: https://doi.org/10.1007/s12652-020-02876-1.

[18] A. S. Tewari, "Generating Items Recommendations by Fusing Content and User-Item based Collaborative Filtering," *Procedia Comput. Sci.*, vol. 167, no. 2019, pp. 1934–1940, 2020, doi: https://doi.org/10.1016/j.procs.2020.03.215.

[19] Y. Afoudi, M. Lazaar, and M. Al Achhab, "Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network," *Simul. Model. Pract. Theory*, vol. 113, no. June, p. 102375, 2021, doi: https://doi.org/10.1016/j.simpat.2021.102375.

[20] D. Wang, Y. Liang, D. Xu, X. Feng, and R. Guan, "A content-based recommender system for computer science publications," *Knowledge-Based Syst.*, vol. 157, pp. 1–9, 2018, doi: https://doi.org/10.1016/j.knosys.2018.05.001.

[21] M. Abdel-Basset, M. Mohamed, M. Elhoseny, L. H. Son, F. Chiclana, and A. E. N. H. Zaied, "Cosine similarity measures of bipolar neutrosophic set for diagnosis of bipolar disorder diseases," *Artif. Intell. Med.*, vol. 101, p. 101735, 2019, doi: https://doi.org/10.1016/j.artmed.2019.101735.

[22] R. H. Singh, S. Maurya, T. Tripathi, T. Narula, and G. Srivastav, "Movie Recommendation System using Cosine Similarity and KNN," *Int. J. Eng. Adv. Technol.*, vol. 9, no. 5, pp. 556–559, 2020, doi: https://doi.org/10.35940/ijeat.e9666.069520.

[23] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, "A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction," *J. Appl. Sci. Technol. Trends*, vol. 1, no. 2, pp. 56–70, 2020, doi: https://doi.org/10.38094/jastt1224.

[24] S. Hakak, M. Alazab, S. Khan, T. R. Gadekallu, P. K. R. Maddikunta, and W. Z. Khan, "An ensemble machine learning approach through effective feature extraction to classify fake news," *Futur. Gener. Comput. Syst.*, vol. 117, pp. 47–58, 2021, doi: https://doi.org/10.1016/j.future.2020.11.022.

[25] S. W. Kim and J. M. Gil, "Research paper classification systems based on TF-IDF and LDA schemes," *Human-centric Comput. Inf. Sci.*, vol. 9, no. 1, 2019, doi: https://doi.org/10.1186/s13673-019-0192-7.

[26] J. M. Kudari, "Fake News Detection using Passive Aggressive and TF-IDF Vectorizer," *Int. Res. J. Eng. Technol.*, pp. 1601–1603, 2020.
[27] S. Pericherla and E. Ilavarasan, "Performance analysis of Word Embeddings for Cyberbullying Detection," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1085, no. 1, p. 012008, 2021. https://doi.org/10.1088/1757-899x/1085/1/012008
[28] T. Schick and H. Schütze, "BERTRAM: Improved word embeddings have big impact on contextualized model performance," *Proc. Annu. Meet. Assoc. Comput. Linguist.*, pp. 3996–4007, 2020, doi: https://doi.org/10.18653/v1/2020.acl-main.368.