DOI: 10.15849/IJASCA.221128.11

Int. J. Advance Soft Compu. Appl, Vol. 14, No. 3, November 2022
Print ISSN: 2710-1274, Online ISSN: 2074-8523

Sustainable Development: A Semantics-aware Trends for Movies Recommendation System using Modern NLP

Shadi AlZu'bi, Amjed Zraiqat, and Samar Hendawi

Faculty of Science and Information Technology Al-Zaytoonah University of Jordan
Amman, Jordan
e-mail: smalzubi@zuj.edu.jo, amjad@zuj.edu.jo, samaralhendawi@gmail.com

Abstract

Recommendation systems are an important feature in the proposed virtual life, where users are often stuck with choices most of the time and need help to be able to find what they are looking for. In this work, content-based techniques have been employed in the proposed recommender system in two ways, a deep review for content and features contents such as (cast, crew, keywords, and genres) has been conducted. A preprocessing stage using TF-IDF and CountVectorizer methods have been employed efficiently to prepare the dataset for any similarity measurements. Cosine similarity algorithm has been employed as well with and without sigmoid and linear kernals. The achieved result proves that similarities between movies using TF-IDF with - Cosine similarity (sigmoid kernel) overcomes the TF-IDF with - Cosine similarity (linear_kernel) and Cosine similarity with CountVectorizer in collaborative filtering. The accuracy values of different machine learning models are validated with K-fold Cross Validator techniques. The performance evaluation has been measured using ROOT Mean Square Error and Mean Absolute Error. Five Machine learning algorithms (NormalPredictor, SVD, KNNBasic (with k=20 and K=10), KNNBasic (with sim_options), and NMF (in several rating scales)). Accuracies are finally been validated with 3 folds from each validator. The best achieved RMSE and MAE scores are using SVD (RMSE = 90%) and (MAE = 69%), followed by KNNBasic (with sim_options, K= 20), NMF, KNNBasic (K=20), KNNBasic (K=10), ending with KNNBasic(sim_options, K= 10).

Keywords: *Recommendation System, Sustainable Development, Artificial Intelligence, Collaborative Filtering, Content-Based, Cosine Similarity, Movies Recommendation, NLP, Machine Learning Application.*

1. Introduction

In the digital world now, and with the increase in the amount of data [35] and transactions on this data, which occur every minute significantly with the number of users on the Internet, some of them are useful and give satisfactory results to users, and some of them are not coordinated and need proper processing, so it becomes difficult for users to obtain What they are looking for and what they need from this data. With the

Received 17 June 2022; Accepted 1 October 2022

emergence of Netflix, YouTube, Amazon [1] [33], and many other service provider websites, recommendation systems are increasing and have become inevitable, as recommendation systems deal with a huge amount of information on the site, whether by users or other elements related to their interests, preferences, and behavior. By using a large dataset of data and information about objects and users. The researchers worked on all recommendation systems to solve this problem [2].

Recommendation systems provide relevant data and information to users and are processed and filtered according to user requirements. Machine-learning algorithms can identify the interests of users, and once they know their interests and preferences, they can easily predict related and similar content for them, which make recommender systems important and powerful. In this research, at first, this system gives recommendations through an overview of the movie, but it is more likely that the user will see a specific movie by actors and directors instead of ratings for the movie, that's why this system takes into consideration keywords, genre, actors and directors as well, to provide better results search. By using TfidfVectorizer [3] and CountVectorizer [4] algorithms through cosine similarity [5] To find the two movies that are the most similar, and thus can be recommended together. The remaining parts of this paper are structured as follow: review of the related literature is overviewed in section 2. The proposed methodology is presented in section 3, including proposed system, overview of dataset, and data pre-processing stages. Section 4 presents the conducted experimental results. Finally, the work is concluded and a discussion is provided in section 5.

2. Literature review

Many researchers have researched in the field of recommendation systems [22] [23], and there are many recommendations systems techniques that have been researched by researchers, and with the increase in information and the number of items that can be viewed or purchased, and the ability to analyze information, recommendation systems have become very important and necessary and there are a lot of recommendations techniques which have been investigated by many researchers. And with the increase in the number of items that users can buy/watch [25], recommendation systems [24] [28] have become necessary and available. Therefore, many researchers have directed their attention to recommendation systems and important applications in various fields such as music, e-commerce sites [30] [31], videos, e-learning [32], online review [19] and many others, as reviewed in [1]. Content-based, collaborative, and hybrid filtering techniques are also reviewed, in addition to the problems faced by these systems.

Rujhan Singla et al. introduced in [6] a new model that recommended movies based on various features such as ratings, plots, year of release, and countries of production using a combination of tf-idf and doc2vec algorithm to create a hybrid system that combines content-based technology and collaborative filtering.

M.R. Lee et al. Proposed in [7] a hybrid recommendation system that combined collaborative filtering and content-based using Facebook data and machine language to increase accuracy, the system recommended items to the user, and compared output results with algorithms used in Netflix, YouTube, and Amazon.

Betül Bulut et al. built in [8] a recommendation system based on a publisher's previous articles using the TF-IDF and Cosine Similarity algorithm, they worked on previously published articles of the same publisher and found similar articles to recommend.

Collaborative filtering is the smartest recommendation system and works widely in the technical world and web services [11], [12], [17], it works on users and similar elements [13], i.e. on similar preferences and tastes, and is divided into two parts, the first is memory-based approach and the second is model-based approach [14], the first uses techniques to find the relationship between similar users (User-Based Collaborative Filtering model and Item-Based Collaborative Filtering model) by exploring similar users with similar behaviors and tastes as well as their favorite items and then recommending similar results to them. The model-based approach explores matching models from training data such as rating The user of models and items on the same items and then the process of recommending similar results to them.

In [15], Garg et al. proposed a recommendation system using collaborative filtering technology by using user ratings to suggest lists, and the authors have used the Apache Mahout framework and the comparison was done mainly to show the efficiency and performance of user based and item-based recommendations.

In [16], Nakhliet al. suggest a movie recommendation system to users by displaying the percentage of recommendations, as they found similar and relevant movies to users, and the results were compared to movie names randomly to find out the accuracy of the system.

Based on reviewed literature, recommender systems have been implemented in several ways, the proposed recommender system in this work will employ content-based techniques in two ways, a deep review for content and features contents such as (cast, crew, keywords, and genres) will be conducted. The accuracy of different machine learning models is validated using K-fold Cross validation techniques. The performance evaluation has been measured using RMSE (ROOT Mean Square Error) [39] and MAE (Mean Absolute Error) [40]. Five Machine learning algorithms (NormalPredictor, SVD, KNNBasic (with k=20 and K=10), KNNBasic (with sim_options), and NMF (in several rating scales)). Accuracies are finally been validated with 3 folds from each validator.

3. Methodology

In this work, content-based techniques have been employed in two ways, a deep review for content and features contents such as (cast, crew, keywords, and genres) has been conducted. As illustrated in figure 1, the dataset is collected to cover the proposed techniques, a preprocessing stage using TF-IDF [41] and CountVectorizer methods have been applied efficiently to prepare the dataset for any similarity measurements. Then, cosine similarity algorithm has been applied with and without sigmoid and linear kernals. The performance evaluation process is then applied to evaluated achieved results.

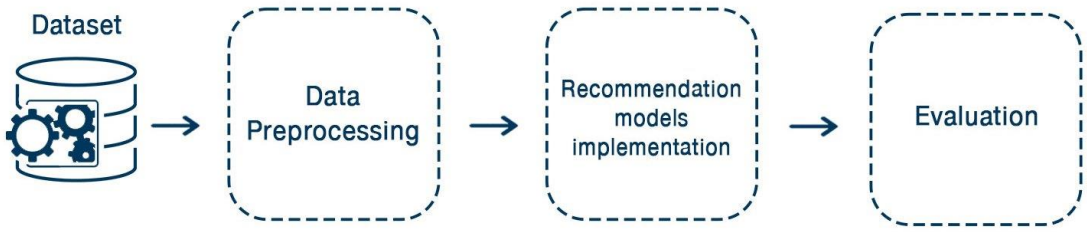
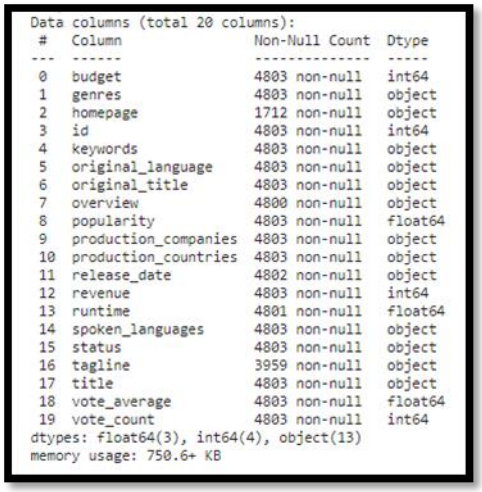


Figure 1. Followed stages for implementing the proposed recommender system

3.1. Dataset

The main input of the proposed system is a metadata for 4,803 movies listed in the MovieLens Dataset which is available at [10]. The dataset is composed of 2 CSV files

(tmdb_5000_credits.csv and tmdb_5000_movies.csv). The tmdb_5000_movies.csv dataset consists of the following feature, and Figure 2 illustrates the available feature of this dataset, where:



```
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   budget                4803 non-null   int64
 1   genres                4803 non-null   object
 2   homepage              1712 non-null   object
 3   id                    4803 non-null   int64
 4   keywords              4803 non-null   object
 5   original_language     4803 non-null   object
 6   original_title        4803 non-null   object
 7   overview              4800 non-null   object
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object
 10  production_countries  4803 non-null   object
 11  release_date          4802 non-null   object
 12  revenue               4803 non-null   int64
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object
 15  status                4803 non-null   object
 16  tagline               3959 non-null   object
 17  title                 4803 non-null   object
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
```

Figure 2: Movie dataset attributes

- budget: It displays budget for the movie.
- genres: It displays movie genres like horror, action, romance, etc. One movie can have multiple genres.
- homepage: It displays the main movie page. Through the movie link on the website.
- id: It Displays movie ID.
- keywords: It displays the keywords of the movie, not just the name of the movie, but some keywords can be added to describe the movie.
- original_language: It displays the original language of the movie either in English or any other languages.
- original_title: Displays title of movie.
- overview: Displays a short description of the movie.
- popularity: Displays indicate popularity.
- production_companies:Displays the names of the companies that produced the movie
- production_countries: Displays the names of the countries in which the movie was produced.
- release_date: Displays the movie's release date.
- revenue: Displays the revenue generated by the movie.
- runtime: Displays the time of the movie, in other words, the length of the movie.
- spoken_languages: Displays the movie languages.
- status: Displays movie status. ex., Was the movie released or not?
- tagline: Displays the movie's tagline.
- title: It displays the movie title.
- vote_average: It displays the average of the votes.
- vote_count: It displays the vote count.

The 'tmdb_5000_credits.csv' dataset consists of the following attributes which are illustrated in Figure 3.

- movie_id: It displays the movie ID.
- title: It displays the movie title.
- cast: Displays the cast Including actresses or actor in the movie.

- crew: Displays the names of people interested in the production of the movie.

```
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   movie_id  4803 non-null   int64
 1   title     4803 non-null   object
 2   cast      4803 non-null   object
 3   crew      4803 non-null   object
dtypes: int64(1), object(3)
memory usage: 150.2+ KB
```

Figure 3: Credits dataset attributes

The ratings.csv dataset consists of the following attributes, and Figure 4 illustrates the available feature of this dataset

- userId: It Displays the user ID.
- movieId: It Displays the movie ID.
- rating: Displays the rating of the movie.
- timestamp.

```
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   userId     100004 non-null  int64
 1   movieId    100004 non-null  int64
 2   rating     100004 non-null  float64
 3   timestamp  100004 non-null  int64
dtypes: float64(1), int64(3)
memory usage: 3.1 MB
```

Figure 4: Ratings dataset attributes

3.2. Data Pre-processing

The pre-processing steps in the TF-IDF with - Cosine similarity (sigmoid kernel) and TF-IDF with - Cosine similarity (linear_kernel) algorithms include:

- Merging two tables (credits, movies) together by (id) and creating a new frame
- Remove all features that are not needed in the recommendation process (homepage', 'title' 'status','production_countries)
- Removing stopwords which are the words without any contextual meaning, such as; the, for, an, a, or, what and etc.

And in the Cosine similarity with (CountVectorizer ()):

- After merging the tables together by (id) the features ('cast', 'crew', 'keywords', 'genres') are combined with each other.
- Convert all strings to strip names of spaces and lower case.
- Removing stopwords which are the words without any contextual meaning, such as; the, for, an, a, or, what and etc.

3.3. Models Descriptions

The following are the machine learning models [18] used in this recommendation engine

- TF-IDF: TF-IDF stands for "Term Frequency — Inverse Document Frequency". This technique identifies words in a large set of documents, a score is calculated for each word to indicate its importance in the set or document, and this technique is mainly used to Text Mining [27] [34] [36] and retrieve information [41]

$$\text{Tfidf}_{i,j} = \text{tf}_{i,j} \text{ X } \log(\frac{N}{dfi})$$

$\text{tf}_{i,j}$ = total number of occurrences of I in j
df_i = total number of documents (speeches) containing i
N = total number of documents (speeches)

- Cosine Similarity: Cosine Similarity is a measurement that quantifies the similarity between two or more vectors [26] [42]

$$\text{Similarity} = \cos(\theta) = \frac{A.B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

- NormalPredictor: NormalPredictor algorithm predicts a random rating based on the distribution of the training set, which is assumed to be normal. This is one of the most 0basic algorithms that do not do much work [43]
- SVD: SVD is a form of matrix factorization that uses gradient descent to create predictions for a users' ratings, while minimizing the error between the predicted ratings and the actual ratings from our original utility matrix. As a result, gradient descent minimizes RMSE when predicting these new ratings [37]
- KNN(Basic): These algorithms look at the nearest neighbors to determine which movie to predict [29]
- KNNBasic with sim_option: An important parameter for k-NN-based algorithms in Surprise is sim_options, which describes options for similarity calculation. Using sim_options, you can set how similarity is calculated, such as similarity metrics. The default setting is using the Mean Squared Difference (MSD) to calculate pairwise similarity and user-based recommendations. [29]
- NMF: is a collaborative filtering algorithm based on Non-negative Matrix Factorization. It is very similar with SVD [38]
- (RMSE): One of the approaches to measure the accuracy of your result is the Root Mean Square Error (RMSE), in which you predict ratings for a test dataset of user-item pairs whose rating values are already known. The difference between the known value and the predicted value would be the error. Square all the error values for the test set, find the average (or mean), and then take the square root of that average to get the RMSE. [39]
- (MAE), Another metric to measure the accuracy is Mean Absolute Error (MAE), in which you find the magnitude of error by finding its absolute value and then taking the average of all error values. [40]

3.4. Proposed system

The purpose of this section is to introduce the proposed system for recommendations, which consists of two basic recommendation techniques, content-based filtering and collaborative filtering.

3.4.1. Proposed Content based Filtering Technique

As illustrated in figure 5, we propose the Content-based Movie Recommender System [9] to recommend movies to users. By using TF-IDF and Cosine similarity algorithm, with content information (e.g., overview, cast, crew, etc.) to find the recommended movies.

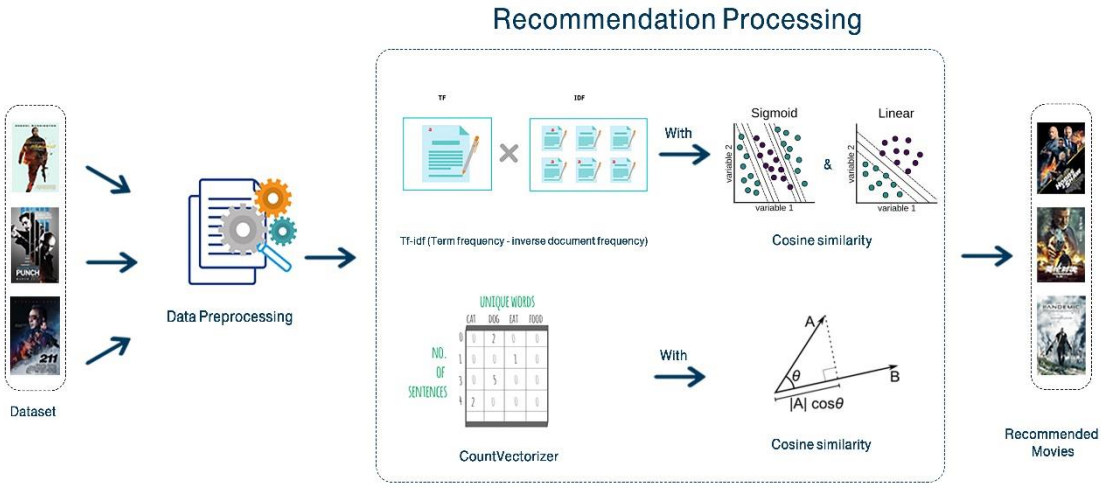


Figure 5. Proposed system architecture content-based filtering

3.4.2. Proposed Collaborative Filtering Technique

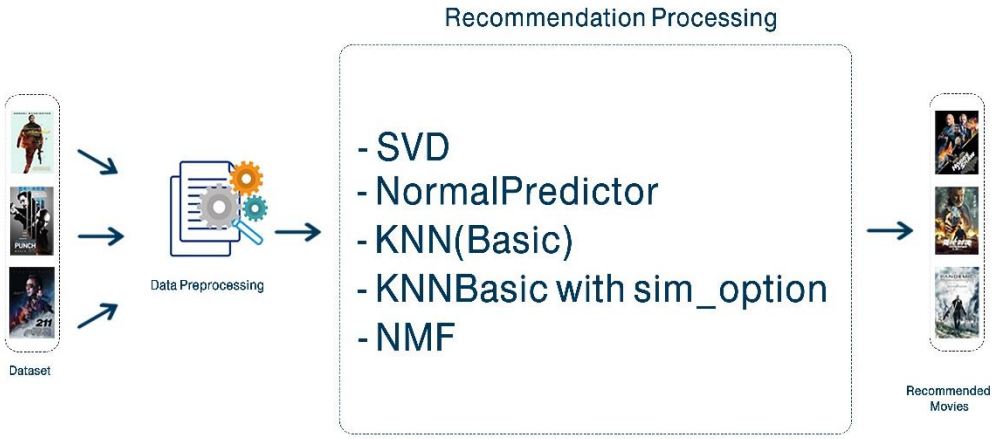


Figure 6. Proposed system architecture collaborative filtering

4. Experiment

The movie plots are transformed as vectors in a geometric space by using TfidfVectorizer. Therefore, the angle between two vectors represents the closeness of those two vectors. Cosine similarity (sigmoid kernel) calculates similarity by measuring the cosine of the angle between two vectors.

4.1. Content-based filtering

4.1.1. TF-IDF with - Cosine similarity (sigmoid kernel)

After applying the pre-processing, and using the TF-IDF with Cosine similarity (sigmoid kernel), Finally, creating a Function to get recommendations for a movie based on the overview:

- Step 1 - get corresponding index of the movie title

- Step 2 - get pairwise similarity scores by compute the sigmoid kernel
- Step 3 - Sort the movies
- Step 4 - Find the scores of 10 most similar movies
- Step 5 - get the movie indices of those top 10 movies
- Step 6 - Return the top 10 most similar movies

4.1.2. TF-IDF with - Cosine similarity (linear kernel)

Table 1. Results achieved using TF-IDF with - Cosine similarity (sigmoid kernel)

Algorithm Name	**TF-IDF with - Cosine similarity (sigmoid kernel)**		
Move Name	**ID**	**Movie Name**	**Similarity**
Toy Story	42	Toy Story 3	0.761615808014468
	343	Toy Story 2	0.761612612823572
	1779	The 40-Year-Old Virgin	0.761604420938317
	3379	Factory Girl	0.761600492653791
	891	Man on the Moon	0.761600323625803
	3873	Class of 1984	0.761600298132932
	3065	Heartbeeps	0.761599844940631
	2869	For Your Consideration	0.761599103118482
	3383	Losin' It	0.761598156407280
	4108	In the Shadow of the Moon	0.761597892308527
Move Name	**ID**	**Movie Name**	**Similarity**
Dead Man Walking	2501	Hachi: A Dog's Tale	0.761608090056399
	3112	Blood Done Sign My Name	0.761603683293596
	1917	The 33	0.761603211552533
	1247	City By The Sea	0.761602900323398
	2242	Flash of Genius	0.761602521572203
	2775	Find Me Guilty	0.761601853365787
	1538	Chicago	0.761601107831435
	980	The Life of David Gale	0.761600685253497
	3220	The Haunting in Connecticut 2: Ghosts of Georgia	0.761600672148274
	2686	An American Haunting	0.761600591291688

After applying the pre-processing, and using the TF-IDF with Cosine similarity (linear kernel), Finally, creating a Function to get recommendations for a movie based on the overview:

- Step 1 - Get the movie index from its title
- Step 2 - get pairwise similarity by compute the linear kernel.
- Step 3 - Sort the list of groups based on similarities.
- Step 4 - Get the top (10)elements of this list.
- Step 5 - get the movie indices of those top 10 movies
- Step 6 - Return the top (10) most similar movies

Table 1 illustrates the results achieved using TF-IDF with - Cosine similarity (sigmoid kernel). And Table 2 illustrates the results achieved using TF-IDF with - Cosine similarity (linear kernel)

4.1.3. CountVectorizer() and Cosine similarity:

In order to reach the similarities between the movies, we need to vectorize, so it was used CountVectorizer rather than TfIdfVectorizer to convert a set of documents into a vector for a vector of token and terms through the chosen features will be actors, directors, genres and keywords, and then using cosine similarity, where the Calculate similarity by measuring the cosine of the angle between two vectors. Table 3 illustrates the results achieved using CountVectorizer() with - Cosine similarity

Table 2. Results achieved using TF-IDF with - Cosine similarity (linear kernel)

Algorithm Name	**TF-IDF with - Cosine similarity (linear_kernel)**		
Move Name	**ID**	**Movie Name**	**Similarity**
Toy Story	42	Toy Story 3	0.5139717597090055
	343	Toy Story 2	0.4419975381029676
	1779	The 40 Year Old Virgin	0.2883149988494541
	2869	For Your Consideration	0.14274741231334165
	891	Man on the Moon	0.14142186365206538
	3873	Class of 1984	0.13829365963627427
	3379	Factory Girl	0.13453437975175306
	3065	Heartbeeps	0.12097667557439606
	3383	Losin' It	0.11213276811682503
	2569	Match Point	0.1063232203349871
Move Name	**ID**	**Movie Name**	**Similarity**
Dead Man Walking	2501	Hachi: A Dog's Tale	0.13426954152366083
	4703	Tadpole	0.13277023891593548
	1468	The Fountain	0.1257701772207817
	2686	An American Haunting	0.12521254762594974
	3307	Driving Miss Daisy	0.11391951024429915
	980	The Life of David Gale	0.11134529178847712
	3528	Diary of a Mad Black Woman	0.10280101725641294
	2499	Enter the Void	0.102179569158528
	3236	The Sound of Music	0.09956831862094445
	2214	Three to Tango	0.09599302581352255

- Step 1 - Parse the stringified features
- Step 2 - Extract the information required from each feature.
- Step 3 - Convert keywords and names to lowercase and remove all spaces between them.
- Step 4 - Create "metadata soup", It is a string containing all the metadata and information that we want to feed to our routing tool (i.e. keywords, directors, actors)
- Step 5 - Use the CountVectorizer() instead of TF-IDF
- Step 6 - Finaly, get_recommendations() function by passing in the new cosine_sim2 matrix as your second argument.

Table no. 1,2, and 3 shows a comparison of the recommendations of other movies for the following movies (The Dark Knight Rises, The Twilight Saga: Eclipse) using TF-IDF with - Cosine similarity (sigmoid kernel and linear_kernel), and Cosine similarity with (CountVectorizer ()). When looking at the results, we see that the results of the

recommendations were better when using the using TF-IDF with - Cosine similarity (sigmoid kernel)

Table 3. Results achieved using CountVectorizer() with - Cosine similarity

Algorithm Name	**(CountVectorizer()) with Cosine similarity**		
Move Name	**ID**	**Movie Name**	**Similarity**
Toy Story	343	Toy Story 2	0.4705882352941177
	42	Toy Story 3	0.3894904188522601
	2209	3 Ninjas Kick Back	0.2858309752375148
	221	Stuart Little 2	0.2782074420373286
	66	Up	0.2711630722733202
	459	Spirit: Stallion of the Cimarron	0.26462806201248157
	3580	Doug's 1st Movie	0.25928148942086576
	77	Inside Out	0.25854384499750954
	1695	Aladdin	0.2475368857441686
	258	The Smurfs 2	0.24253562503633297
Move Name	**ID**	**Movie Name**	**Similarity**
Dead Man Walking	4564	Straight Out of Brooklyn	0.25000000000000000
	980	The Life of David Gale	0.24253562503633297
	2216	We're No Angels	0.20801257358446093
	4475	Interview with the Assassin	0.20801257358446093
	327	The Lovely Bones	0.18750000000000000
	2030	Derailed	0.18750000000000000
	2520	The Color Purple	0.18750000000000000
	3567	Monster	0.18750000000000000
	2671	Born on the Fourth of July	0.18190171877724973
	3057	American History X	0.18190171877724973

Table 4 illustrates the results achieved using CountVectorizer() and Cosine similarity by returning the list of top 3 elements. And Table 5. illustrates the results achieved using CountVectorizer() and Cosine similarity by returning the list of top 5 elements. Table 6 illustrates the results achieved using CountVectorizer() and Cosine similarity by returning the list of top 7 elements.

Table 4.Results achieved using CountVectorizer() and Cosine similarity by returning the list of top 3 elements

Algorithm Name	**(CountVectorizer()) with Cosine similarity** with director's name from the crew feature and returns the list of top 3 elements		
Move Name	**ID**	**Movie Name**	**Similarity**
Toy Story	42	Toy Story 3	0.6666666666666667
	343	Toy Story 2	0.5555555555555556

	533	Monster House	0.4444444444444444
	1983	Meet the Deedles	0.408248290463863
	3403	Alpha and Omega: The Legend of the Saw Tooth Cave	0.408248290463863
	927	Christmas with the Kranks	0.3779644730092272
	120	Madagascar: Escape 2 Africa	0.35355339059327373
	725	The Shaggy Dog	0.35355339059327373
	1451	Zoom	0.35355339059327373
	1580	The Nut Job	0.35355339059327373
Move Name	**ID**	**Movie Name**	**Similarity**
Dead Man Walking	327	The Lovely Bones	0.40089186286863654
	2030	Derailed	0.40089186286863654
	2216	We're No Angels	0.3779644730092272
	4564	Straight Out of Brooklyn	0.3779644730092272
	3414	Incendies	0.3585685828003181
	2079	Moonlight Mile	0.3380617018914066
	3497	The Greatest	0.3380617018914066
	2058	Stone	0.3086066999241838
	4676	Middle of Nowhere	0.3086066999241838
	1307	The Hurricane	0.28571428571428564

Table No. 4,5, and 6 shows a comparison of the recommendations for the mentioned movie (The Dark Knight Rises and The Twilight Saga: Eclipse) using the Cosine similarity and CountVectorizer() with the director's name from the crew feature and returns the list of top 3,5, and 7 elements, When looking at the results, we see that when using (returns the list of top 5) showed better results for recommending films in terms of similarity.

Table 5. Results achieved using CountVectorizer() and Cosine similarity by returning the list of top 5 elements

Algorithm Name	(**CountVectorizer()) with Cosine similarity** with director's name from the crew feature and returns the list of top 5 elements		
Move Name	**ID**	**Movie Name**	**Similarity**
Toy Story	42	Toy Story 2	0.501280411827603
	343	Toy Story 3	0.461538461538461
	221	3 Ninjas Kick Back	0.336336396998156
	77	Stuart Little 2	0.326860225230306
	3580	Up	0.320256307610174
	820	Spirit: Stallion of the Cimarron	0.307692307692307
	258	Doug's 1st Movie	0.296499726664440
	533	Inside Out	0.296499726664440
	2209	Aladdin	0.296499726664440
	1983	The Smurfs 2	0.294174202707276
Move Name	**ID**	**Movie Name**	**Similarity**
Dead Man Walking	4564	Straight Out of Brooklyn	0.2886751345948129
	2216	The Life of David Gale	0.2611164839335468

	2520	We're No Angels	0.2611164839335468
	327	Interview with the Assassin	0.25000000000000006
	2030	The Lovely Bones	0.25000000000000006
	4475	Derailed	0.25000000000000006
	980	The Color Purple	0.24019223070763074
	2671	Monster	0.24019223070763074
	3414	Born on the Fourth of July	0.2314550249431379
	2079	American History X	0.2182178902359924

Table 6. Results achieved using CountVectorizer() and Cosine similarity by returning the list of top 7 elements

Algorithm Name	(**CountVectorizer()) with Cosine similarity** with director's name from the crew feature and returns the list of top 7 elements		
Move Name	**ID**	**Movie Name**	**Similarity**
Toy Story	343	Toy Story 2	0.4705882352941177
	42	Toy Story 3	0.3894904188522601
	2209	3 Ninjas Kick Back	0.2858309752375148
	221	Stuart Little 2	0.2782074420373286
	66	Up	0.2711630722733202
	459	Spirit: Stallion of the Cimarron	0.26462806201248157
	3580	Doug's 1st Movie	0.25928148942086576
	77	Inside Out	0.25854384499750954
	1695	Aladdin	0.2475368857441686
	258	The Smurfs 2	0.24253562503633297
Move Name	**ID**	**Movie Name**	**Similarity**
Dead Man Walking	4564	Straight Out of Brooklyn	0.25000000000000000
	980	The Life of David Gale	0.24253562503633297
	2216	We're No Angels	0.20801257358446093
	4475	Interview with the Assassin	0.20801257358446093
	327	The Lovely Bones	0.18750000000000000
	2030	Derailed	0.18750000000000000
	2520	The Color Purple	0.18750000000000000
	3567	Monster	0.18750000000000000
	2671	Born on the Fourth of July	0.18190171877724973
	3057	American History X	0.18190171877724973

4.2. Collaborative filtering:

KNN collaborative filtering algorithm, which is a KNN algorithm combined with collaborative filtering algorithm, use KNN algorithm to select the nearest neighbor. The basic steps of the KNN algorithm are user similarity calculation and predict score calculation [20] [21].

For the purpose of evaluation, the accuracy values of different machine learning models are validated with K-fold Cross Validator techniques. Two accuracy measures i.e., MAE (Mean Absolute Error) and RMSE (ROOT Mean Square Error). Five Machine learning

algorithms (NormalPredictor(), SVD(), KNNBasic(k=20) and (K=10), KNNBasic(sim_options), NMF(), in rating_scale=(1, 5) and rating_scale=(0.5, 4) accuracies have been validated with 3 folds from each validator. Tables 7 and 8 presents the k-Nearest Neighbors with Cosine similarity at n_neighbors=10 and n_neighbors=5 respectively. While table 9 presents RMSE and MEA achieved results using the proposed models.

Table 7. Results achieved using the k-Nearest Neighbors with Cosine similarity at n_neighbors=10

Algorithm Name	**k-Nearest Neighbors with Cosine similarity n_neighbors=10**		
Move Name	**No.**	**Movie Name**	**Distance**
Toy Story	1	Frankie Starlight	0.887183
	2	Made in America	0.885979
	3	The Stars Fell on Henrietta	0.877353
	4	Carrington	0.869415
	5	White Man's Burden	0.847254
	6	The Swan Princess	0.842476
	7	The Adventures of Priscilla, Queen of the Desert	0.837181
	8	Roula	0.805907
	9	Star Trek: Generations	0.628148
Move Name	**No.**	**Movie Name**	**Distance**
Dead Man Walking	1	Pather Panchali	0.625620
	2	The Beans of Egypt, Maine	0.600583
	3	Boomerang	0.590750
	4	The Three Musketeers	0.585393
	5	Homeward Bound II: Lost in San Francisco	0.574251
	6	Some Folks Call It a Sling Blade	0.566521
	7	Billy's Holiday	0.550769
	8	White Squall	0.545411
	9	Colonel Chabert	0.524174

Figure 7 represents K-fold cross-validation mean accuracy results of models used in the proposed recommender system in rating scale= (1, 5). The lower value of RMSE and MAE are considered to be good accuracy model. From Table 9, it can be noticed that the best RMSE and MAE scores are attained by SVD i.e. 90% (RMSE) and 69% (MAE) followed by KNNBasic(sim_options, K= 20), NMF, KNNBasic (K=20), KNNBasic (K=10), KNNBasic(sim_options, K= 10).

Table 8. Results achieved using the k-Nearest Neighbors with Cosine similarity at n_neighbors=5

Algorithm Name	**k-Nearest Neighbors with Cosine similarity n_neighbors=5**		
Move Name	**No.**	**Movie Name**	**Distance**
Toy Story	1	Frankie Starlight	0.887183
	2	Made in America	0.885979
	3	The Stars Fell on Henrietta	0.877353
	4	Carrington	0.869415
	5	White Man's Burden	0.847254
	6	The Swan Princess	0.842476

	7	The Adventures of Priscilla, Queen of the Desert	0.837181
	8	Roula	0.805907
	9	Star Trek: Generations	0.628148
Move Name	**No.**	**Movie Name**	**Distance**
Dead Man Walking	1	Pather Panchali	0.625620
	2	The Beans of Egypt, Maine	0.600583
	3	Boomerang	0.590750
	4	The Three Musketeers	0.585393
	5	Homeward Bound II: Lost in San Francisco	0.574251
	6	Some Folks Call It a Sling Blade	0.566521
	7	Billy's Holiday	0.550769
	8	White Squall	0.545411
	9	Colonel Chabert	0.524174

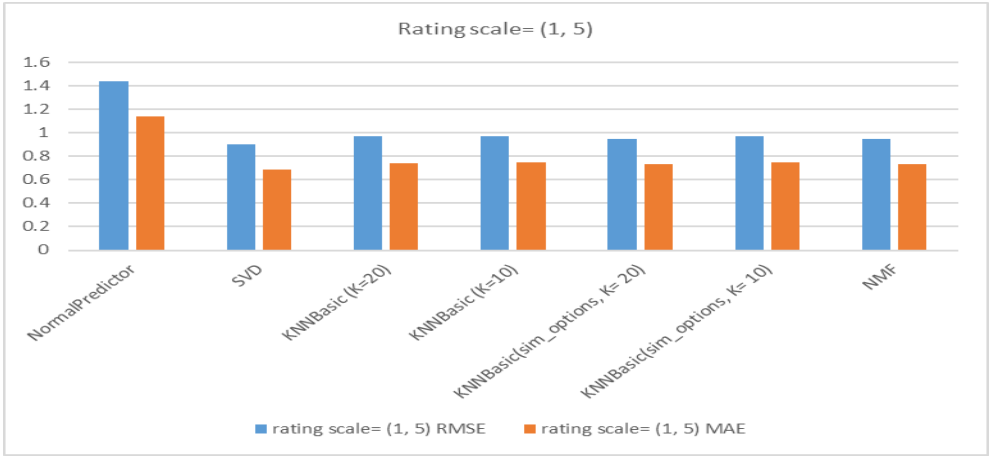


Figure 7. K-fold cross-validation mean RMSE and MEA achieved results at rating scale = (1 , 5)

Table 9. RMSE and MEA achieved results using the proposed models at rating scale = (1 , 5)

rating scale= (1, 5)		
Models	**RMSE**	**MAE**
NormalPredictor	1.44	1.14
SVD	0.90	0.69
KNNBasic (K=20)	0.97	0.74
KNNBasic (K=10)	0.97	0.75
KNNBasic(sim_options, K= 20)	0.95	0.73
KNNBasic(sim_options, K= 10)	0.97	0.75
NMF	0.95	0.73

Table 10 represents the RMSE for the experimented 3 K-Folds using several methods at rating scale = (1 , 5). The best RMSE score achieved from K-Fold CV fold in the SVD algorithm is 0.89 which is almost equal in each fold, the achieved results here are visually illustrated in figure 8. While Table 11 represents the MAE for the experimented 3 K-Folds using several methods. The best MAE score achieved from K-Fold CV fold in the SVD algorithm is 0.69, the achieved results here are visually illustrated in figure 9.

Table 10. RMSE for the experimented 3 K-Folds using several methods at rating scale = (1 , 5)

K-Fold (RMSE)	**Fold 1**	**Fold 2**	**Fold 3**
NormalPredictor	1.43	1.44	1.44
SVD algorithm	0.90	0.89	0.90
KNNBasic(k=20)	0.96	0.98	0.97

KNNBasic(k=10)	0.98	0.97	0.98
KNNBasic(sim_options, k=20)	0.95	0.94	0.95
KNNBasic(sim_options, k=10)	0.97	0.97	0.98
NMF algorithm	0.96	0.94	0.93

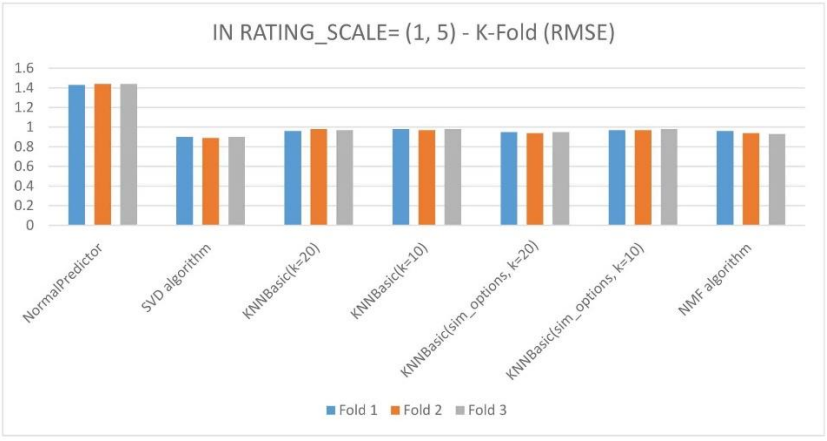


Figure 8. RMSE for the experimented 3 K-Folds using several methods at rating scale = (1 , 5)

Table 11. MAE for the experimented 3 K-Folds using several methods at rating scale = (1 , 5)

K-Fold (MAE)	Fold 1	Fold 2	Fold 3
NormalPredictor	1.14	1.14	1.14
SVD algorithm	0.69	0.69	0.7
KNNBasic(k=20)	0.74	0.75	0.74
KNNBasic(k=10)	0.75	0.74	0.75
KNNBasic(sim_options, k=20)	0.73	0.73	0.73
KNNBasic(sim_options, k=10)	0.75	0.75	0.75
NMF algorithm	0.73	0.72	0.72

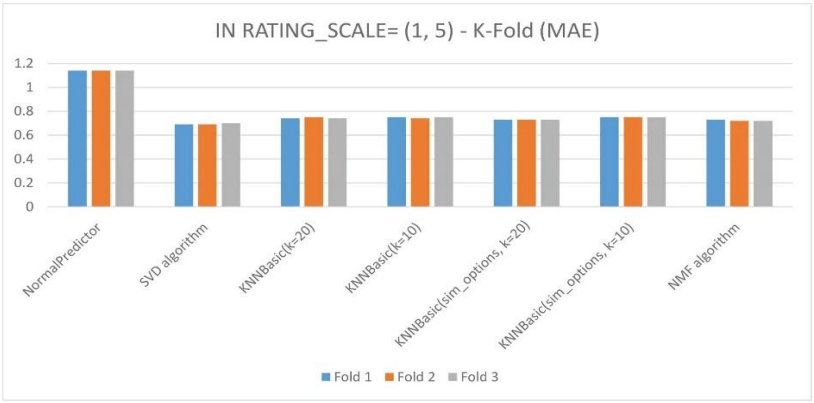


Figure 9. MAE for the experimented 3 K-Folds using several methods at rating scale = (1 , 5)

Figure 10 represents K-fold cross-validation mean accuracy results of models used in our recommendation engine in rating scale= (0.5, 4). The lower value of RMSE and MAE are considered to be good accuracy model. From Table 12, it can be noticed that the best RMSE and MAE scores are attained by SVD i.e. 91% (RMSE) and 70% (MAE) followed by KNNBasic(sim_options, K= 20), NMF, KNNBasic (K=20), KNNBasic (K=10), KNNBasic(sim_options, K= 10).

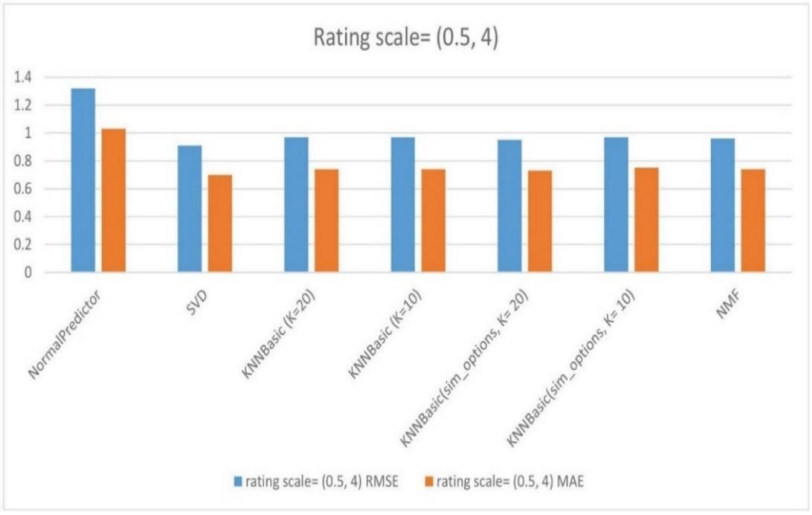


Figure 10. K-fold cross-validation mean RMSE and MEA achieved results at rating scale = (0.5 , 4)

Table 12. RMSE and MEA achieved results using the proposed models at rating scale = (0.5 , 4)

rating scale= (0.5, 4)		
Models	**RMSE**	**MAE**
NormalPredictor	1.32	1.03
SVD	0.91	0.70
KNNBasic (K=20)	0.97	0.74
KNNBasic (K=10)	0.97	0.74
KNNBasic(sim_options, K= 20)	0.95	0.73
KNNBasic(sim_options, K= 10)	0.97	0.75
NMF	0.96	0.74

Table 13 represents the RMSE for the experimented 3 K-Folds using several methods at rating scale = (0.5, 4). The best RMSE score achieved from K-Fold CV fold in the SVD algorithm is 0.9145 which is almost equal in each fold, the achieved results here are visually illustrated in figure 11. While Table 14 represents the MAE for the experimented 3 K-Folds using several methods at rating scale = (0.5 , 4). The best MAE score achieved from K-Fold CV fold in the SVD algorithm is 0.7092, the achieved results here are visually illustrated in figure 12.

Table 13. RMSE for the experimented 3 K-Folds using several methods at rating scale = (0.5 , 4)

K-Fold (RMSE)	Fold 1	Fold 2	Fold 3
NormalPredictor	1.3262	1.3252	1.3217
SVD algorithm	0.9109	0.9163	0.9145
KNNBasic(k=20)	0.9768	0.9751	0.971
KNNBasic(k=10)	0.9689	0.979	0.9782
KNNBasic(sim_options, k=20)	0.9592	0.9561	0.9561
KNNBasic(sim_options, k=10)	0.9736	0.9729	0.9851
NMF algorithm	0.9684	0.9636	0.9615

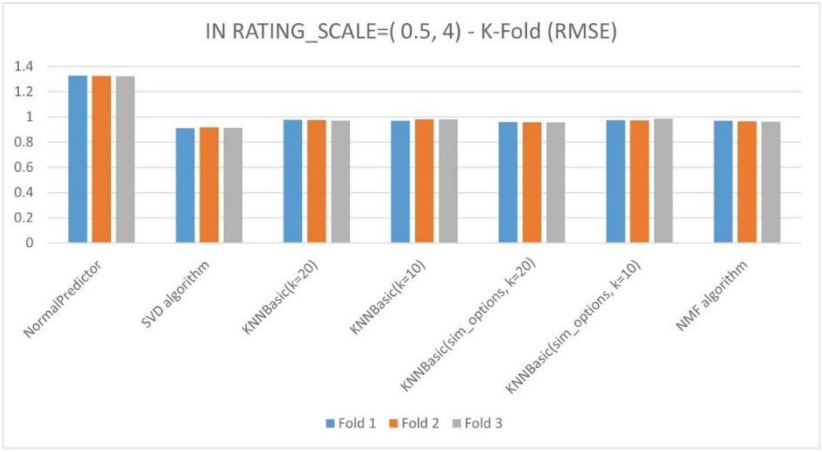


Figure 11. RMSE for the experimented 3 K-Folds using several methods at rating scale = (0.5 , 4)

Table 14. MAE for the experimented 3 K-Folds using several methods at rating scale = (0.5 , 4)

K-Fold (MAE)	Fold 1	Fold 2	Fold 3
NormalPredictor	1.0342	1.0335	1.0342
SVD algorithm	0.7017	0.7092	0.7077
KNNBasic(k=20)	0.7433	0.7469	0.7443
KNNBasic(k=10)	0.7397	0.748	0.7486
KNNBasic(sim_options, k=20)	0.7416	0.7381	0.7371
KNNBasic(sim_options, k=10)	0.7509	0.7532	0.763
NMF algorithm	0.7476	0.7405	0.7405

Figure 12. MAE for the experimented 3 K-Folds using several methods at rating scale = (0.5 , 4)

5. Conclusion

One advantage of content-based filtering is that the machine learning model does not require any information about the user, only an idea of the user's interests is required. Therefore, content-based filtering model uses metadata, and does not have any problems with cold start, with possible restrictions. In this system, only movie overview, keywords, genre, and actors are used. Furthermore, there are many users who prefer higher-rated movies or movies produced in a particular year. However, more features have to be extracted to improve the quality and accuracy of users recommendations.

Researchers could work on a collaborative or mixed recommendation system that combines filtering Collaborative and content-based filtering, by focusing more on the user and not just the content in terms of watching the movie, the amount of time spent watching, or rating the movies. All this will help in better prediction and

recommendation as well as getting high accuracy in the recommendation process. In this work, the CountVectorizer() utilizes many features for the recommendation process, and the results in terms of the names of the proposed movie were better, but through the similarity that used in each algorithm, it can be noticed from the achieved results that the recommendations by using TF-IDF with - Cosine similarity (sigmoid kernel) are slightly better than the TF-IDF with - Cosine similarity (linear_kernel), and Cosine similarity with (CountVectorizer()).

In collaborative filtering technique evaluation, it is concluded that the accuracy values of the K-fold Cross validator accuracy measures RMSE and MAE are utilized for performance evaluation purposes. From the K-Fold cross validator mean accuracy results (Lower Values are better), the best RMSE and MAE scores are obtained using SVD where the RMSE was 90% and the MAE was 69%, followed by KNNBasic(sim_options, K= 20), NMF, KNNBasic (K=20), KNNBasic (K=10), KNNBasic(sim_options, K= 10).

References

[1] Goyani, Mahesh and Chaurasiya, (2020) Neha. A Review of Movie Recommendation System. *ELCVIA: electronic letters on computer vision and image analysis*, Vol. 19, Num. 3, pp. 18-37, https://raco.cat/index.php/ELCVIA/article/view/373942 [View: 6-05- 2022]

[2] Fayyaz Z, Ebrahimian M, Nawara D, Ibrahim A, Kashef R. (2020) Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities. *Applied Sciences*.Vol 10(21):7748.

[3] V. Kumar and B. Subba, (2020) A TfidfVectorizer and SVM based sentiment analysis framework for text data corpus. *National Conference on Communications (NCC)*, pp. 1-6.

[4] S. C. Eshan and M. S. Hasan, (2017) An application of machine learning to detect abusive Bengali text. *20th International Conference of Computer and Information Technology (ICCIT)*, pp. 1-6.

[5] Rahutomo, F., Kitasuka, T., and Aritsugi, M. (2012). Semantic cosine similarity. *In The 7th international student conference on advanced science and technology ICAST* (Vol. 4, No. 1, p. 1).

[6] R. Singla, S. Gupta, A. Gupta and D. K. Vishwakarma, (2020) FLEX: A Content Based Movie Recommender," *International Conference for Emerging Technology (INCET)*, pp. 1-4.

[7] Lee, Maria R., Tsung Teng Chen, and Ying Shun Cai. (2016) Amalgamating Social Media Data and Movie Recommendation." *Pacific Rim Knowledge Acquisition Workshop. Springer International Publishing*.

[8] T. Kenter and M. de Rijke, (2015) Short Text Similarity with Word Embeddings Categories and Subject Descriptors," *Proc. 24th ACM Int. Conf. Inf. Knowl. Manag. (CIKM 2015)*, pp. 1411– 1420.

[9] Reddy, S. R. S., Nalluri, S., Kunisetti, S., Ashok, S., & Venkatesh, B. (2019). Content-based movie recommendation system using genre correlation. *In Smart Intelligent Computing and Applications* (pp. 391-397). Springer, Singapore.

[10] F. Maxwell Harper and Joseph A. Konstan. (2015) The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst. 5, 4, Article 19* (January 2016), 19 pages.

[11] J. Liu, M. Tang, Z. Zheng, X. F. Liu, and S. Lyu, (2016) Locationaware and personalized collaborative filtering for web service recommendation, *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 686–699, 2016

[12] L. Yao, Q. Z. Sheng, A. H. Ngu, J. Yu, and A. Segev, (2015) Unified collaborative and content-based web service recommendation, *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 453–466.

[13] H. Chen and J. Li, (2017) Learning multiple similarities of users and items in recommender systems, *in IEEE International Conference on Data Mining (ICDM). IEEE*, pp. 811–816

[14] W. Pan and L. Chen, (2013) Cofiset: Collaborative filtering via learning pairwise preferences over item-sets, *in Proceedings of the 2013 SIAM International Conference on Data Mining. SIAM*, pp. 180–188

[15] C. S. M. Wu, D. Garg, and U. Bhandary, (2018) Movie Recommendation System Using Collaborative Filtering, *In 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, pp. 11-15, IEEE.

[16] R. E. Nakhli, H. Moradi, and M. A. Sadeghi, (2019) Movie Recommender System Based on Percentage of View, *In 2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, pp. 656-660, IEEE

[17] Hawashin, B., Mansour, A., Fotouhi, F., AlZu'bi, S. and Kanan, T., (2021). A Novel Recommender System Using Interest Extracting Agents and User Feedback. *In 2021 International Conference on Information Technology (ICIT)* (pp. 674-678). IEEE.

[18] Lafi, M., Hawashin, B. and AlZu'bi, S., (2021). Eliciting Requirements from Stakeholders' Responses Using Natural Language Processing. *Computer Modeling in Engineering & Sciences*, 127(1), pp.99-116.

[19] Alsmadi, A., AlZu'bi, S., Al-Ayyoub, M. and Jararweh, Y., (2020). Predicting Helpfulness of Online Reviews. *arXiv preprint arXiv:2008.10129*.

[20] Hawashin, B., Alzubi, S., Mughaid, A., Fotouhi, F. and Abusukhon, A., (2020). An efficient cold start solution for recommender systems based on machine learning and user interests. *In 2020 Seventh International Conference on Software Defined Systems (SDS)* (pp. 220-225). IEEE.

[21] Lafi, M., Hawashin, B. and AlZu'bi, S., (2020). Maintenance requests labeling using machine learning classification. *In 2020 Seventh International Conference on Software Defined Systems (SDS)* (pp. 245-249). IEEE.

[22] Alzubi, S., Hawashin, B., Mughaid, A. and Jararweh, Y., (2020). Whats Trending? An Efficient Trending Research Topics Extractor and Recommender. *In 2020 11th International Conference on Information and Communication Systems (ICICS)* (pp. 191-196). IEEE.

[23] Hawashin, B., Aqel, D., Alzubi, S. and Elbes, M., (2020). Improving recommender systems using co-appearing and semantically correlated user interests. *Recent Advances in Computer Science and Communications (formerly: Recent Patents on Computer Science)*, 13(2), pp.240-247.

[24] Muhairat, M., AlZu'bi, S., Hawashin, B., Elbes, M.W. and Al-Ayyoub, M., (2020). An Intelligent Recommender System Based on Association Rule Analysis for Requirement Engineering. *J. Univers. Comput. Sci*., 26(1), pp.33-49.

[25] AlZu'bi, S., Alsmadiv, A., AlQatawneh, S., Al-Ayyoub, M., Hawashin, B. and Jararweh, Y., (2019). A brief analysis of amazon online reviews. In *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)* (pp. 555-560). IEEE.

[26] Hawashin, B., Aqel, D., AlZu'bi, S. and Jararweh, Y., (2019). Novel weighted interest similarity measurement for recommender systems using rating timestamp. *In 2019 Sixth International Conference on Software Defined Systems (SDS)* (pp. 166-170). IEEE.

[27] Hawashin, B., Alzubi, S., Kanan, T. and Mansour, A., (2019). An efficient semantic recommender method forarabic text. *The Electronic Library*.

[28] AlZu'bi, S., Hawashin, B., EIBes, M. and Al-Ayyoub, M., (2018). A novel recommender system based on apriori algorithm for requirements engineering. *In 2018 fifth international conference on social networks analysis, management and security (snams)* (pp. 323-327). IEEE.

[29] Alhadid, I., Khwaldeh, S., Al Rawajbeh, M., Abu-Taieh, E., Masa'deh, R.E. and Aljarah, I., (2021). An intelligent web service composition and resource-optimization method using K-means clustering and knapsack algorithms. *Mathematics*, 9(17), p.2023.

[30] Al Rawajbeh, M., (2017). LOW COST DESIGN AND IMPLEMENTATION FOR HAS USING MULTIFUNCTIONAL WI-FI. *International Journal of Computer Networks & Communications (IJCNC)*, 9(3), pp.105-116.

[31] Al Rawajbeh, M. and Haboush, A., (2011). Enhancing the egovernment functionality using knowledge management. *In World Academy of Science, Engineering and Technology*.

[32] AlZu'bi, S., Abu Zitar, R., Hawashin, B., Abu Shanab, S., Zraiqat, A., Mughaid, A., Almotairi, K.H. and Abualigah, L., (2022). A Novel Deep Learning Technique for Detecting Emotional Impact in Online Education. *Electronics*, 11(18), p.2964.

[33] Mughaid, A., Al-Zu'bi, S., Al Arjan, A., Al-Amrat, R., Alajmi, R., Zitar, R.A. and Abualigah, L., (2022). An intelligent cybersecurity system for detecting fake news in social media websites. *Soft Computing*, 26(12), pp.5577-5591.

[34] Kanan, T., Hawashin, B., Alzubi, S., Almaita, E., Alkhatib, A., Maria, K.A. and Elbes, M., (2022). Improving arabic text classification using p-stemmer. *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)*, 15(3), pp.404-411.

[35] Hattawi, W., Shaban, S., Al Shawabkah, A. and Alzu'bi, S., (2021). Recent Quality Models in BigData Applications. *In 2021 International Conference on Information Technology (ICIT)* (pp. 811-815). IEEE.

[36] Kanan, T., AbedAlghafer, A., Kanaan, G.G., AlShalabi, R., Elbes, M. and AlZubi, S., (2021). Arabic Text Categorization: A Comparison Survey. *In 2021 International Conference on Information Technology (ICIT)* (pp. 739-742). IEEE.

[37] Aharon, M., Elad, M., & Bruckstein, A. (2006). K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11), 4311-4322.

[38] Gaussier, E., & Goutte, C. (2005). Relation between PLSA and NMF and implications. *In Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 601-602).

[39] Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research*, 30(1), 79-82.

[40] Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. *Geoscientific model development*, 7(3), 1247-1250.

[41] Joachims, T. (1996). A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. *Carnegie-mellon univ pittsburgh pa dept of computer science.*

[42] Nguyen, H. V., & Bai, L. (2010). Cosine similarity metric learning for face verification. *In Asian conference on computer vision (pp. 709-720). Springer, Berlin, Heidelberg.*

[43] Irwin, J. R., & McClelland, G. H. (2003). Negative consequences of dichotomizing continuous predictor variables. *Journal of Marketing Research*, 40(3), 366-371.

Notes on contributors

Shadi AlZu'bi is Associate Professor at the Department of Computer Science, Al-Zaytoonah University of Jordan, Amman, Jordan. His main teaching and research interests include Ai, Image processing, Data Science and Intelligent Systems. He has published several research articles in international highly impacted journals.

Amjed Zraiqat is Professor at the Department of Mathematics, Al-Zaytoonah University of Jordan, Amman, Jordan. His main teaching and research interests include ordinary & partial differential equations, Operator Theory and Applied Mathematics. He has published several research articles in international journals of mathematics.

Samar Hendawi is a master student at the Department of Computer Science, Al-Zaytoonah University of Jordan, Amman, Jordan. Her research interest is Data Science and Intelligent Systems.