Collaborative filtering-based group recommender system using sparse autoencoder

Musthafa Zaki Bahar [a,1], Zinke Abdurahman Baizal [a,2,*]

[a] School of Computing, Telkom University, Bandung, Indonesia
[1] mzakibahar@student.telkomuniversity.ac.id; [2] baizal@telkomuniversity.ac.id
* corresponding author

ARTICLE INFO

Article history
Received August 16, 2024
Revised October 17, 2025
Accepted October 26, 2025
Available online February 28, 2026

Keywords
Group recommender system
Collaborative filtering
Data sparsity
Sparse autoencoder

ABSTRACT

The development of technology makes the distribution of information easier and faster, but leads to information overload. A recommender system is one tool to overcome information overload, while the collaborative filtering (CF) paradigm is a widely used approach in recommender systems. The recommender system generally focuses on individual recommendations, but in real conditions, recommendations for a group are often needed, for example, when we want to listen to music with friends, or we plan a vacation with family. Many prior studies have used the CF paradigm with matrix factorization to build group recommender systems. Matrix factorization has been shown to alleviate the sparsity problem; however, it does not fully resolve it. Therefore, we propose an approach that uses a sparse autoencoder to address this sparsity issue. We chose the sparse autoencoder because it can effectively capture latent patterns in sparse data by learning a compressed representation while retaining important features crucial for accurate recommendations. We built a group recommender system with three different group sizes and aggregation approaches. For evaluation, we use the root-mean-square error (RMSE) and the mean absolute error (MAE). Test results indicate that the sparse autoencoder outperforms matrix factorization in terms of RMSE and MAE. This study improves group recommender systems by addressing data sparsity using a sparse autoencoder. The proposed approach enhances recommendation accuracy compared to traditional matrix factorization methods.

1. Introduction

Information overload is a condition in which a person or group has difficulty processing and managing information effectively [1]. This occurs due to the overwhelming amount of available information [2]. A recommender system is one key approach to mitigating this information overload [3]. A recommender system provides preferences by collecting information such as ratings, reviews, or opinions from previous users to be processed and given to users in the form of preferences [4].

To date, many recommender systems have been developed for personal user cases [5]. However, in daily life, there is often interaction in groups, such as listening to music with friends, watching movies with family, planning vacations with colleagues, etc. [6]. Therefore, a group recommender system has the same level of importance to be completed in addition to a personal recommender system [6], [7], [8]. A group recommender system can be built with two methods, i.e., aggregate model and aggregate prediction [9]. The aggregate model is a method that combines user profiles from each group member into a group profile before making a recommendation [9]. Meanwhile, aggregate prediction is a method that combines recommendation results for each user profile from a group [9]. Both methods share the same challenges, one of which is data sparsity [10].

Data sparsity is a condition where there are many items that have not been rated by users [11]. In previous research, the matrix factorization (MF) model, which is a paradigm of collaborative filtering (CF) is widely used to build a group recommender system [6], [12], [13], [14]. The MF model has shown good performance, but this model has shortcomings in overcoming data sparsity, which can reduce the accuracy of prediction [15], [16], [17]. MF is proven to be able to solve the sparsity problem; however, MF does not completely solve this problem [17].

Lately, deep learning has received significant attention in the development of machine learning, one of which is in the development of recommender system [18], [19]. Deep learning is applied in recommender systems to provide alternative solutions to accuracy, cold-start, and data sparsity problems that exist in previous recommender systems [20]. Autoencoder (AE) is among the deep learning techniques employed in recommender systems to overcome the scalability and data sparsity issue [20], [21]. Sparse AE is a variant of AE that adds sparse regularization to the model [22]. In previous research, sparse AE has been used in personal recommender systems to address sparsity [23]. This model shows better performance compared to several base models tested using datasets with different levels of sparsity [23].

Therefore, we propose an approach by utilizing sparse AE to address the sparsity issue. In this study, we apply the sparse AE model to a group recommender system. Indeed, many studies have been conducted on group recommender systems to address the sparsity problem. MF has indeed been shown to overcome this sparsity problem, but it cannot fully do so. Meanwhile, sparse AE in some previous studies has shown better ability to overcome the sparsity problem. From that, we hypothesize that applying sparse AE to the group recommender system can improve its performance. We use several aggregation methods and various group formation size scenarios to build a group recommender system. The evaluation is done by comparing our proposed model with the MF model in the same scenario. In the evaluation, we use root mean square error (RMSE) and mean absolute error (MAE).

The rest of this paper is organized as follows. Section 2 outlines the research method used to develop and evaluate a group recommendation system that employs a sparse AE. Section 3 presents the implementation and evaluation results, demonstrating the performance of the sparse AE in the group recommender system. Finally, Section 4 gives the conclusion.

2. Method

2.1. Dataset

The system design of our research is presented in Fig. 1. In this study, we used MovieLens 100k and MovieLens 1M datasets. MovieLens 100k has 100 thousand ratings in the rating range of 1 to 5 from 943 users with 1682 movies. MovieLens 1M has 1 million ratings in the range of 1 to 5 from 6040 users with 2000 movies. We use three columns, i.e, “userId”, “movieId”, and “rating”, to build the group recommender system. The two datasets have sparsity levels of 94% and 96%, respectively.

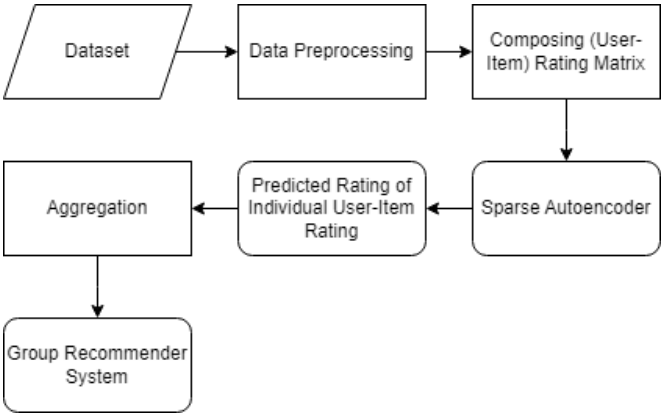

Fig. 1. System design

The system design starts by processing the dataset, which contains interactions between users and items, into training data and testing data. This data is then transformed into a rating matrix. Next, the data in a rating matrix is modeled using a sparse AE. This sparse AE model is used to predict the rating. The resulting rating predictions are aggregated, and the aggregated results are used to generate item recommendations for a group.

2.2. Group Recommender System

A group recommender system is a system that makes a recommendation about an item to a group of users [6], [24]. There are two approaches in group recommender systems: aggregate prediction and aggregate model [10]. In this research, we focus on building a group recommender system using an aggregate prediction approach. Fig. 2 illustrates the building process of a group recommender system using this approach with n users in one group. The first step is to provide recommendations for each user. Afterward, the items from each user's recommendations are combined. Then, the items are sorted to produce a list of recommended items for the group.

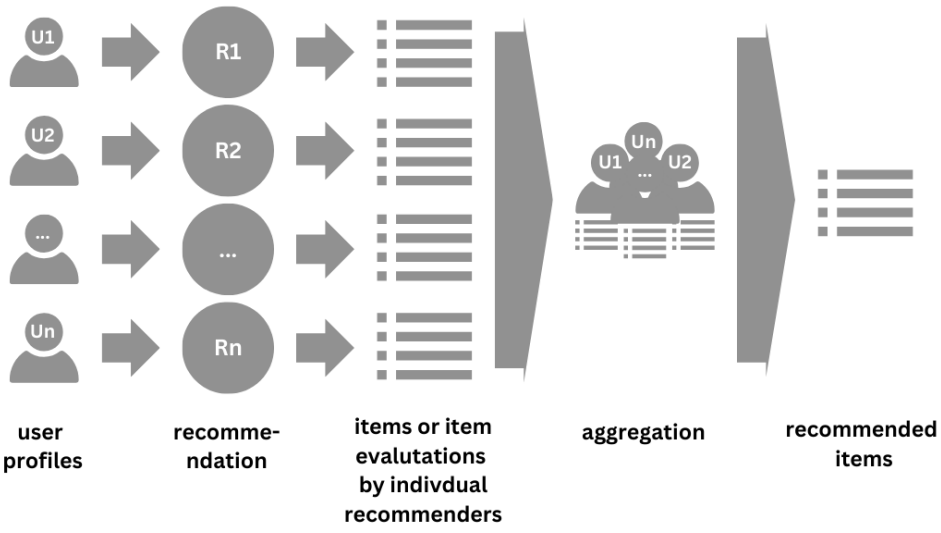

Fig. 2. The visualization of aggregate prediction

2.3. Collaborative Filtering

CF is the common paradigm used in recommender systems [25], [26]. This paradigm focuses on using information from a group of users to make recommendations to other users [27], [28]. CF utilizes the preferences of each active user to provide recommendations [16]. Fig. 3 shows the form of interaction of a group of users with each item.

Item / User	Item 1	Item 2	Item 3	Item 4	Item 5
U1		2	2	3	2
U2	4	1	3		2
U3		4	2	5	
U4	3	2	3	1	3

Fig. 3. The example of rating matrix

A rating is a preference expressed by active users towards an item [29]. Fig. 3 shows how the user interacts with the item: each user gives a rating to the item they have seen, and preferences can differ, as seen in the ratings given. Items that have never been rated by users pose a challenge for CF to predict ratings; this is called the data sparsity problem [30]. CF's main advantage is that it can provide recommendations without requiring detailed information about the items [31]. Recommendations are generated based on the preference patterns of other users who have similar preference patterns [31]. Fig. 4 shows how CF generates item recommendations for a user.

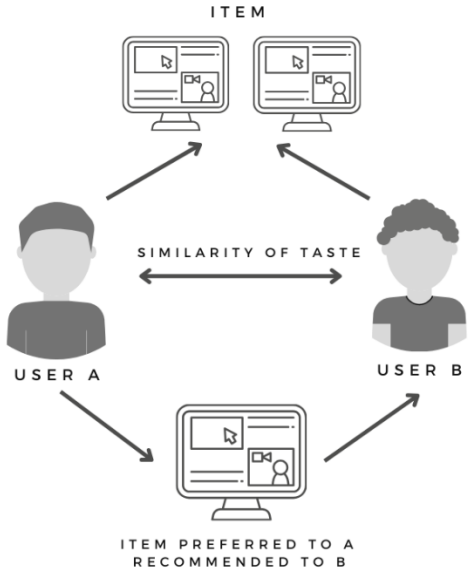

Fig. 4. The example interaction of collaborative filtering

Fig. 4 shows two users, A and B, who share similar preferences for the same item. Therefore, the CF paradigm can provide item recommendations from user A's preferences to user B. However, CF also has disadvantages, such as the cold start problem, which makes the system difficult to provide recommendations for new users because it does not have preference data from the new user [31], [32].

2.4. Sparse Autoencoder in Group Recommendation System

AE is one of the neural network architectures used to solve unsupervised learning problems [33]. AE can compress data, decode, and reconstruct the output [34]. AE has been used in various fields, such as image processing, speech recognition, and information retrieval [21]. This deep learning model has the advantage of overcoming data sparsity problems [20]. Basic AE has three layers, i.e., input layer, hidden layer, and output layer [34]. AE is divided into encoder and decoder [21]. The encoder consists of an input layer and a hidden layer, while the decoder consists of a hidden layer and an output layer [21]. Sparse AE is a variant of AE that adds sparsity regularization to the hidden layer [23], [35]. The addition of sparsity regularization to the traditional autoencoder can be seen in the following formula [36],

$$A_{sparse}(B,a) = A(B,a) + \beta \sum_{i=1}^{m} KL(\rho||\hat{\rho}_i) \quad (1)$$

where the matrix B represents the weight matrix that connects the input layer to the hidden layer in the autoencoder network, while a denotes the bias vector used to adjust neuron activations. The function $A(B,a)$ represents the autoencoder mapping function, which learns a compressed representation of the input data using the parameters B and a. The parameter β used to control the regularization value with a value range of [0, 1]. Meanwhile, the term $\sum_{(i=1)}^{m} KL(\rho\|\hat{\rho}_i)$ represents the sparsity regularization component added to the loss function, where the Kullback–Leibler (KL) divergence measures the difference between the desired average activation ρ and the actual average activation $\hat{\rho}_i$ of hidden units, encouraging the network to learn sparse feature representations.

Sparse AE creates information bottlenecks without reducing the number of nodes in the hidden layer. By adding regularization, the number of active neurons is limited, so the resulting feature representation includes only features considered important [36]. Fig. 5 shows the position of using the sparse AE in our group recommender system.

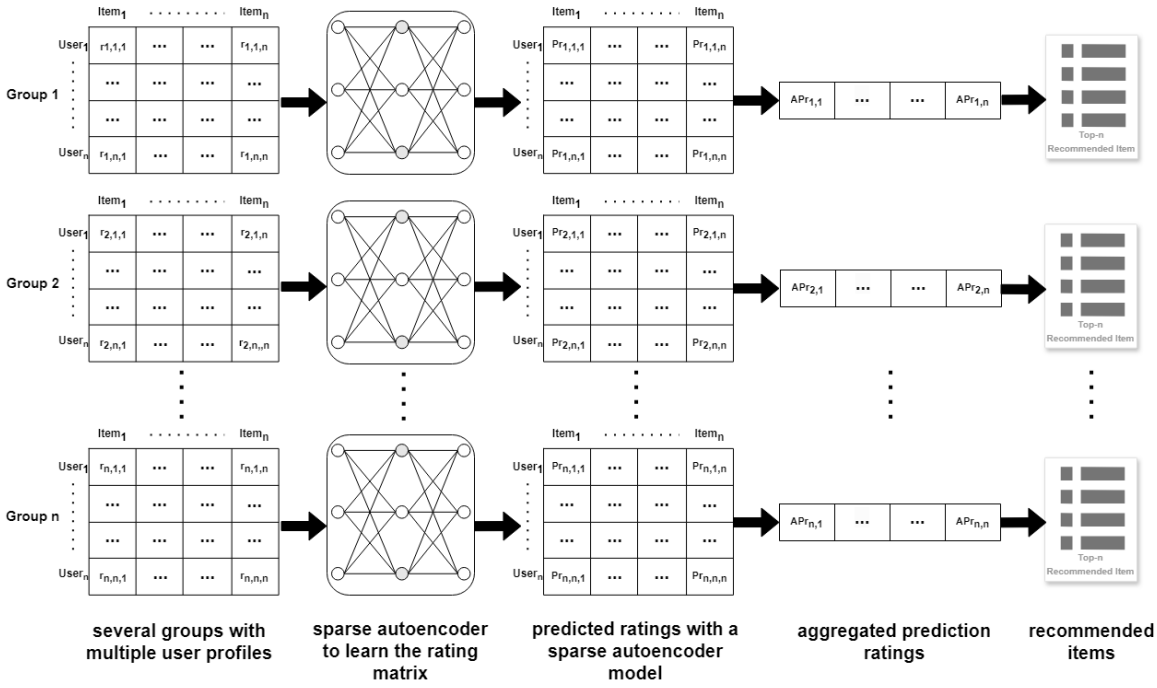

Fig. 5. Sparse autoencoder in group recommender system using aggregate prediction

where $r_{g,i,j}$ is the rating given by user i in group g for item j, $Pr_{g,i,j}$ is the predicted rating for user i in group g for item j, and $APr_{g,i,j}$ is the aggregated prediction rating for group g for item j.

We use a sparse AE to learn the rating matrix. The sparse AE is designed to capture key features of user-item interactions by applying sparsity regularization to the hidden layer. The resulting hidden patterns help predict the output-layer ratings, enabling the system to provide more accurate recommendations. Fig. 6 shows the sparse AE architecture that we used to build the group recommender system.

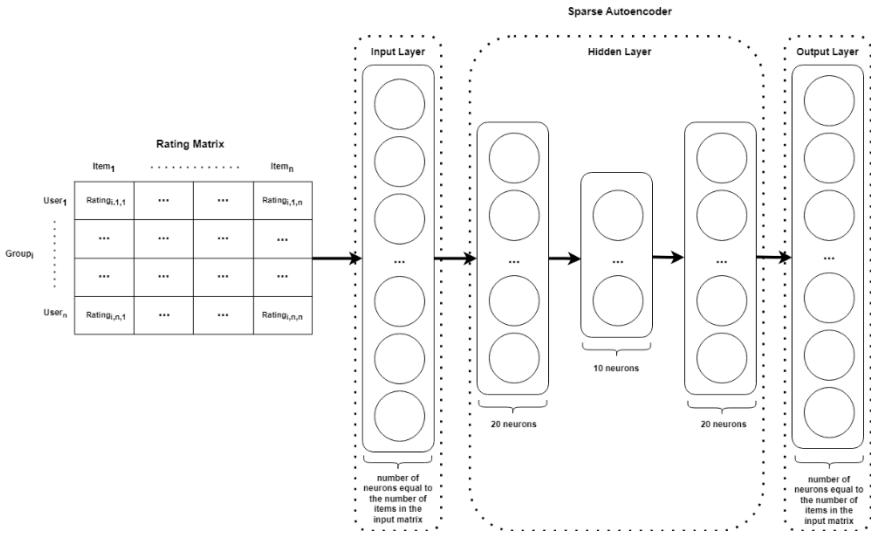

Fig. 6. The architecture of sparse autoencoder in system

Fig. 6 shows the rating matrix of a $Group_i$, which is then passed to our sparse AE architecture for modeling. Our sparse AE architecture begins with the input layer, where the number of neurons equals the number of items in the rating matrix. From the input layer, it goes to the hidden layer, which has three hidden layers. The first layer has 20 neurons, the second layer has 10 neurons, and the third layer has 20 neurons. For the sparse regularization value, we experimented with four values: 0.1, 0.3, 0.5, 0.7, and 0.9. The best sparse regularization value from the experimental results is used to compare with the base model. The sparse AE architecture in Fig. 6 is then used to train the training data. The training process is shown in Algorithm 1 (Fig. 7).

Algorithm 1 trainSparseAE(data_train, epoch, sae, criterion, optimizer)
// Train the sparse AE model

Input : data_train is training data where each entry represents a group of users, epoch is number of epochs for training, sae is the sparse AE model defined in Figure 6, criterion is loss function to measure prediction errors, optimizer is optimization algorithm

```
1:  for each epoch from 1 to epoch do
2:      train_loss ← 0
3:      batch_count ← 0
4:      for each group in data_train do
5:          prepare input and target for the group
6:          if there are non-zero values in target then
7:              output ← sae(input)
8:              loss ← criterion(output, target)
9:              adjust loss relative to non-zero target values
10:             perform backward pass to compute gradients
11:             update model parameters using optimizer
12:             accumulate train_loss
13:             batch_count ← batch_count + 1
14:         end if
15:     end for
16: end for
```

Fig. 7. Sparese AE training algorithm

Algorithm 1 shows the process of training a sparse AE model with training data (Fig. 7). The model training process generally includes initializing model parameters and loss functions, followed by a forward pass in which the input data is processed to generate predictions. The loss is then calculated to measure how accurate the prediction is to the original target value in line 8. In line 10, backpropagation is used to calculate a gradient that determines changes to the model parameters to reduce the loss. The parameters are then updated using an optimization algorithm in line 11. These steps are repeated for each epoch until training is complete.

2.5. Aggregation Strategy

Aggregation strategy in group recommender systems is used to combine several user preferences into a group [6]. Aggregation methods can be classified as borderline, majority-based, and consensus-based [9]. Each aggregation function has its own advantages and disadvantages, and in practice, it can be adapted to the needs of a particular domain [6]. Since the purpose of this research is not domain-specific, we use several aggregation methods. We chose domain-independent aggregation methods because they are widely used and applicable across contexts such as movies, music, and education [6]. This allows us to evaluate the model without relying on domain-specific information. However, different contexts may benefit from specific strategies; for example, MP works well for entertainment, while LM is often used in music to ensure all group members' preferences are considered [6]. The aggregation methods we use are [9]:

- Average (AVG): Aggregation is done by calculating the average of the ratings given. The recommended item is the one with the highest average. The formula for AVG is as follows,

$$AVG(i) = \frac{1}{|G|}\sum_{u \epsilon G} r_{ui} \quad (2)$$

where $AVG(i)$ is the average of item i, G is the user group, r_{ui} represents rating assigned to item i by user u, and $|G|$ is the amount of users in the group.

- Most Pleasure (MP): Aggregation is done by taking the maximum value of the rating given. The recommended item is the item that has the highest value, so the resulting recommendation is an item that is only liked by some users. The formula for MP is as follows,

$$MP(i) = \max_{u \in G} r_{ui} \quad (3)$$

where MP(i) is the highest rating given to item i by a group member.

- Least Misery (LM): Aggregation is done by calculating the minimum value of the rating given. The recommended item is the one with the highest minimum value in the aggregation results, so the recommendation is the item least disliked by some users. The formulation for LM is as follows,

$$LM(i) = \min_{u \in G} r_{ui} \quad (4)$$

where LM(i) is the least rating given to item i by the group members.

- Additive Utilitarian (AU): Aggregation is done by summing up the ratings given. The recommended item is the one with the highest value in the summation result. The formulation for AU is as follows,

$$AU(i) = \sum_{u \epsilon G} r_{ui} \quad (5)$$

where AU(i) is the total rating given to item i by all group members.

- Average Without Misery (AWM): Aggregation is done by calculating the average of the ratings that are above a certain threshold. In this way, ratings that are very low or below the threshold value are not considered. The formula for AWM is as follows,

$$AWM(i) = \frac{\sum_{u \epsilon G} r_{ui} \cdot 1(r_{ui} > T)}{\sum_{u \epsilon G} 1(r_{ui} > T)} \quad (6)$$

where AWM(i) is the average rating only for ratings that exceed the misery threshold T.

We use these five aggregation methods to aggregate the predicted ratings. Our aggregation process can be seen in Algorithm 2 (Fig. 8).

Algorithm 2 aggregatePredictions(r_pred, G, aggregation_strategy) ⟶ aggregated prediction
// Aggregates individual predictions into a group prediction.

Input : r_pred is matrix of individual predictions where r_pred[u, i] is the prediction for user u on item i, G is set of users in the group, aggregation_strategy is the method used for aggregation

```
1:  result ← [ ]
2:  for each item i in r_pred do
3:      values ← collect r_pred[u, i] for all users u in G
4:      aggregated_value ← aggregate values using aggregation_strategy
5:      result[i] ← aggregated_value
6:  end for
7:  return result
```

Fig. 8. Generate aggregate rating prediction algorithm

Algorithm 2 describes the aggregation process for generating an aggregate rating prediction (Fig. 8). First, an initialization step is performed to store the aggregation results. In lines 2 and 3, for each item in r_pred, the algorithm collects predictions from all users in the group for that item. Next, in line 4,

these prediction values are combined using a predefined aggregation method. The aggregation result for each item is stored in line 5, and this process is repeated for all items.

Algorithm 3 describes the process for generating top-N recommended items from aggregated prediction results (Fig. 9). The process starts by initiating an empty structure, top_n_items, to store the top recommendation items. In line 2, the items in aggregated_pred are sorted by prediction value in descending order. Next, the item with the highest value is selected and added to top_n_items until it reaches N. Finally, the algorithm returns the top_n_items list, which contains the N best items based on the aggregated results, to recommend to a group.

Algorithm 3 topNRecommendations(aggregated_pred, N) ⟶ top N items
// Generates top-N recommendations from aggregated predictions.

Input : aggregated_pred is a list of items with their aggregated prediction values, N is the number of top items to recommend

```
1:  top_n_items ← [ ]
2:  sorted_items ← sort aggregated_pred by descending
3:  for each item in sorted items up to N do
4:      top_n_items.append(item)
5:  end for
6:  return top_n_items
```

Fig. 9. Generating top-N recommended items algorithm

3. Results and Discussion

3.1. Evaluation Metrics

We use MAE and RMSE evaluation metrics to evaluate the proposed sparse AE model in building a group recommender system and compare it with the MF model as the base model. MAE and RMSE are evaluation metrics that have been used in previous studies to evaluate group recommender systems [9], [10], [12]. The MAE and RMSE formulas are as follows [37],

$$\text{MAE} = \frac{\sum_i^k (y_i - r_i)}{k} \tag{7}$$

$$\text{PM}\Sigma\text{E} = \sqrt{\frac{\sum_i^k (y_i - r_i)^2}{k}} \tag{8}$$

where y_i is the predicted rating for item i, r_i is the original rating of item i, and k is the number of items.

3.2. Result and Analysis

We conducted experiments using the MovieLens 100k and MovieLens 1M datasets. These two datasets do not have information on how group formation occurs. Therefore, we created 100 groups randomly from these datasets. These groups were formed with three different sizes: a small group with three users, a medium group with five users, and a large group with eight users. These sizes were selected based on previous studies in group recommender systems, representing small, medium, and large groups commonly used in the literature [13], [38]. For each group size, five aggregation methods were used: AVG, MP, LM, AU, and AWM. We trained the sparse AE model using the Adam optimizer with a learning rate of 0.001 for 100 epochs. We divided the dataset into 80% for training and 20% for testing.

For the first experiment, we tested to find the best sparse regularization value. The sparse regularization values we tested were 0.1, 0.3, 0.5, 0.7, and 0.9. Fig. 10 and Fig. 11 show the test results with MAE and RMSE, respectively, using the MovieLens 100k dataset.

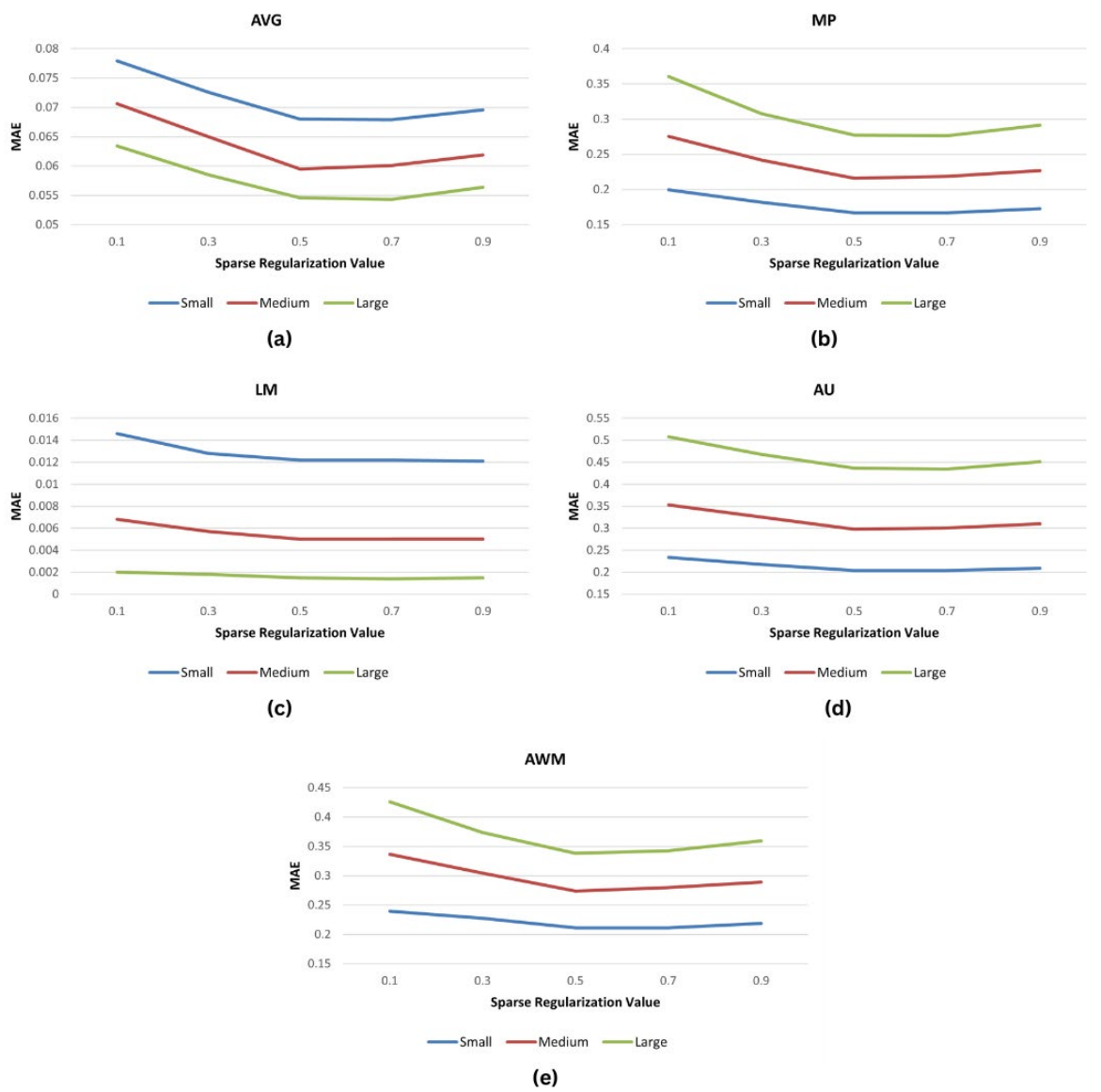

Fig. 10. MAE results from sparse autoencoder tuning experiments

Fig. 10 shows that the overall MAE tends to decrease as the sparse regularization value increases from 0.1 to 0.5. There is no significant change between sparse regularization values of 0.5 and 0.7. However, the MAE tends to increase again as the sparse regularization value increases, as shown in sparse regularization with a value of 0.9.

Fig. 11 shows the test results with RMSE. Overall, the test results are quite similar to those using MAE. Larger sparse regularization values tend to decrease the RMSE, as seen at values of 0.1 to 0.5. However, the RMSE tends to increase again at 0.9. From experiments on the sparse AE model with different sparse regularization values, it can be concluded that the optimal sparse regularization values lie in the range 0.5-0.7. This shows that higher sparse regularization values, which encourage the model to activate fewer neurons, can improve the performance of the group recommender system. However, if the sparse regularization value is too high, resulting in too few active neurons, the group recommender system's performance can be reduced.

From the test results, we found that the best sparse regularization values are 0.5 and 0.7. Therefore, we used the sparse regularization value of 0.5 in sparse AE to build the group recommender system. In the next step, we tested sparse AE against MF as the baseline. Table 1 and Table 2 show the MAE and

RMSE results from tests on the MovieLens 100k and MovieLens 1M datasets using the scenarios we described previously.

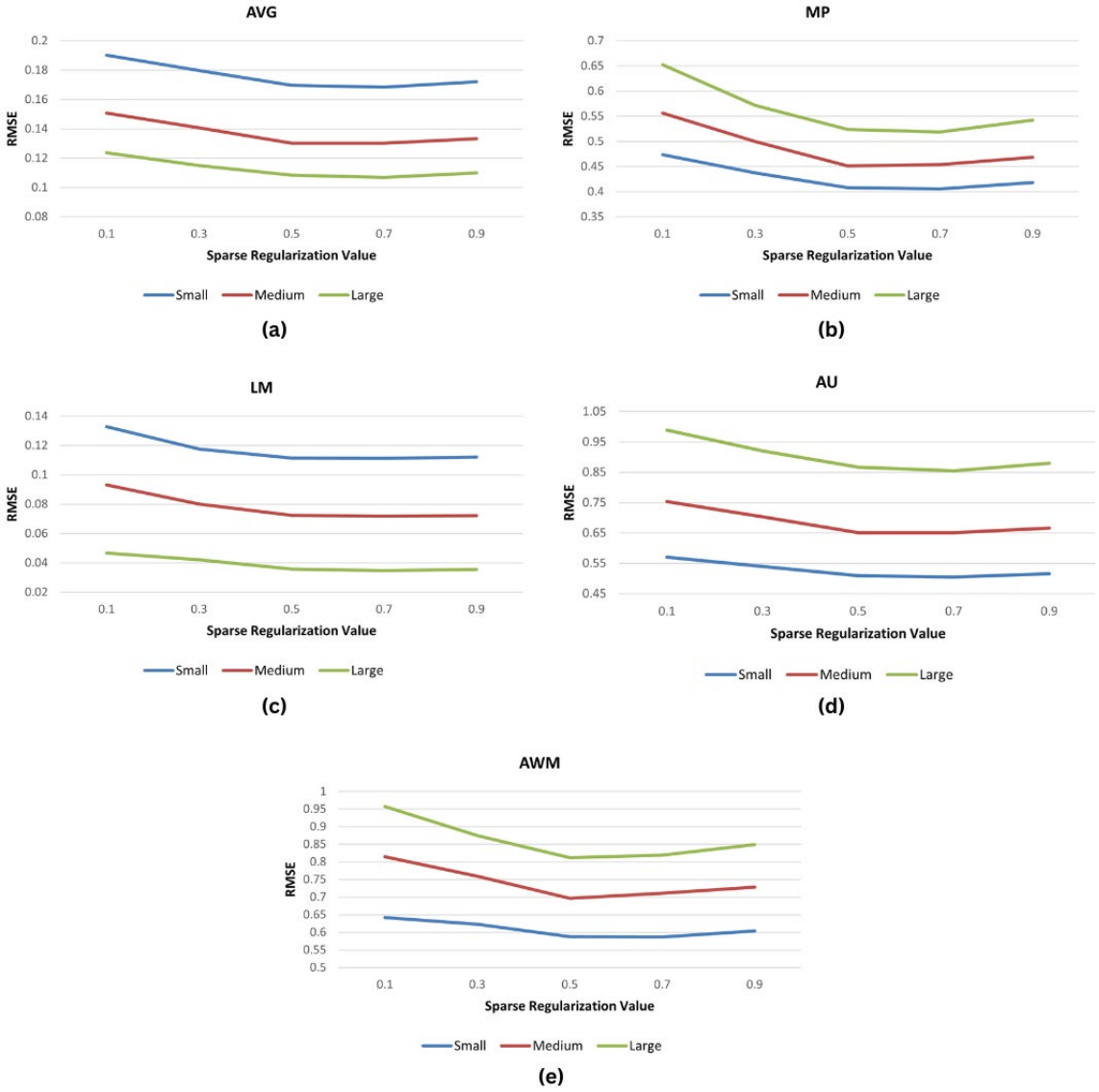

Fig. 11. RMSE results from sparse autoencoder tuning experiments

The first test, shown in Table 1 using the MovieLens 100k dataset, shows that the sparse AE model is overall better than the MF model in terms of MAE and RMSE across various group sizes and aggregation methods. Sparse AE shows significant differences, especially in AVG, MP, AU, and AWM aggregation methods. While in the LM aggregation method, the difference is not significant, but sparse AE is still better.

The second test, shown in Table 2, uses the MovieLens 1M dataset, where the MovieLens 1M dataset has a greater sparsity level than the dataset in the first test. We found that the sparse AE model significantly outperforms the AVG, MP, AU, and AWM aggregation methods based on the MAE and RMSE evaluation metrics. However, in LM aggregation method for small size sparse AE is better although the difference is not significant, but in medium and large size MF is able to maintain its advantage as evidenced by lower MAE and RMSE. This happens because LM takes the lowest rating in the group. In medium and large groups, the chance of having a very low rating is higher, so the result is more influenced by that value. Because there is a greater likelihood of receiving a very low rating in medium and large groups, that value has a greater impact on the outcome. While sparse AE occasionally

produces more varied predictions that may not be as appropriate when LM uses the lowest rating, MF typically produces smoother predictions, making it more stable in LM. The sparse AE learns hidden patterns from user ratings and captures the shared interests among group members. When the aggregation combines or averages individual ratings, sparse AE can use what it has learned to generate more accurate and balanced predictions. In contrast, LM emphasizes the lowest rating, so a single low score can dominate the result and reduce the sparse AE advantage

Table 1. MovieLens 100k result

Experiment		MF		Sparse AE	
		MAE	RMSE	MAE	RMSE
AVG	Small	0.1115	0.2766	0.0641	0.1622
	Medium	0.1060	0.2359	0.0593	0.1305
	Large	0.1036	0.2100	0.0562	0.1104
MP	Small	0.2662	0.6321	0.1596	0.3957
	Medium	0.3497	0.7178	0.2144	0.4529
	Large	0.4573	0.8206	0.2872	0.5333
LM	Small	0.0158	0.1411	0.0114	0.1059
	Medium	0.0063	0.0863	0.0052	0.0709
	Large	0.0019	0.0430	0.0017	0.0380
AU	Small	0.3344	0.8298	0.1923	0.4865
	Medium	0.5299	1.1795	0.2965	0.6523
	Large	0.8286	1.6801	0.4500	0.8831
AWM	Small	0.3476	0.8890	0.2031	0.5761
	Medium	0.4608	1.0570	0.2765	0.7134
	Large	0.5834	1.2075	0.3563	0.8418

Across two tests on two datasets, using several aggregation methods and group sizes, the sparse AE model was overall better than MF. This is evident from the lower MAE and RMSE accuracy metrics. Although in some specific conditions, the sparse AE model is not always better than MF. Based on the evaluation results, we observe that sparse AE, which can overcome sparsity issues, can improve performance in building a group recommender system.

Table 2. MovieLens 1M result

Experiment		MF		Sparse AE	
		MAE	RMSE	MAE	RMSE
AVG	Small	0.0920	0.2485	0.0621	0.1641
	Medium	0.0907	0.2151	0.0575	0.1293
	Large	0.0856	0.1901	0.0509	0.1043
MP	Small	0.2258	0.5793	0.1608	0.4169
	Medium	0.3191	0.6923	0.2294	0.4971
	Large	0.3979	0.7682	0.2966	0.5697
LM	Small	0.0098	0.1100	0.0085	0.0961
	Medium	0.0022	0.0470	0.0021	0.0473
	Large	0.0007	0.0237	0.0008	0.0254
AU	Small	0.2761	0.7455	0.1863	0.4924
	Medium	0.4536	1.0755	0.2874	0.6467
	Large	0.6845	1.5207	0.4072	0.8344
AWM	Small	0.3063	0.8424	0.1964	0.5808
	Medium	0.4407	1.0572	0.2864	0.7474
	Large	0.5346	1.1794	0.3553	0.8593

Compared with previous deep learning approaches in group recommendation, such as trust-aware GRS and hybrid deep MF methods, our work follows a different path. Trust-aware GRS uses virtual coordinators based on social trust among group members, while hybrid deep MF combines latent factors with neural networks to capture complex interactions. Our method relies only on user–item rating data and addresses sparsity with a sparse autoencoder, making it simpler and applicable even when trust or

side information is unavailable, while still showing competitive improvements across most aggregation strategies.

In addition to improving accuracy, sparse AE's gains in MAE and RMSE also show that the recommendations are easier to understand and apply. Lower MAE and RMSE indicate predicted ratings that are more in line with group members' actual preferences, which is significant in real-world group decision scenarios such as watching a movie or organizing a trip. Sparse AE can capture small changes in group members' preferences and make recommendations that satisfy more members than MF, which usually produces smoother but less flexible predictions. To further contextualize the results, we compared our work with several previous group recommender system studies, as summarized in Table 3.

Table 3. Summary of Related Group Recommender System Studies

Author (Year)	Method	Main Contribution / Advantage
Shi et al. [12]	Latent Group Model	Proposed a latent group model that detects hidden user groups using latent factors and aggregates them for efficient and accurate group recommendations on the MovieLens dataset.
Ortega et al. [13]	MF	Proposed three MF-based approaches to map user groups into the latent factor space, demonstrating that MF outperforms KNN-based CF methods across different group sizes on MovieLens and Netflix datasets.
Ahmad et al. [39]	Memory-based CF	Developed a group recommender system using collaborative filtering, tested multiple aggregation methods (average, least misery, most happiness), and found that least misery achieved the best F1-score, especially when group neighborhood weighting was applied.
Vo & Soh [40]	Variational Autoencoder	Built a generative group recommender model using a Variational Autoencoder that jointly creates new items and identifies user groups likely to prefer them, combining rating and reconstruction losses for improved novelty and diversity.
Wang et al. [14]	MF with Trust	Introduced a trust-aware group recommender system (TruGRC) using a virtual coordinator that integrates profile and result aggregation, leveraging social trust to handle conflicting preferences and improve ranking accuracy.
Yannam et al. [10]	Slope One	Proposed a dynamic Slope One–based group recommender, improving efficiency and online prediction performance.
Proposed	Sparse AE	Developed a group recommender system using a sparse autoencoder to address the sparsity problem in collaborative filtering. The model learns compact latent representations from sparse rating data, improving prediction accuracy compared to MF across various aggregation methods and group sizes, as evidenced by lower RMSE and MAE values.

Table 3 summarizes prior studies on group recommender systems and their main contributions. Most approaches improve recommendation quality by integrating trust mechanisms, hybrid learning strategies, or generative models to more effectively capture user interactions. In contrast, our approach mitigates data sparsity using sparse autoencoders that learn compact user–item representations. This enables consistent and domain-independent performance across various aggregation strategies and group sizes.

4. Conclusion

We build a recommender system using the sparse AE model approach. The sparse AE model addresses the sparsity problem, a shortcoming of the MF model widely used in previous research to build group recommender systems. In building the group recommender system, we used several different aggregation strategies, that is, AVG, MP, LM, AU, and AWM, as well as several different group sizes. Evaluation is performed using MAE and RMSE metrics on the MovieLens 100k and MovieLens 1M datasets, which differ in sparsity. The test results show that the sparse AE model is overall better than the MF model, as evidenced by the lower MAE and RMSE values. From these results, applying the sparse AE model, which better handles sparsity than the MF model, can improve the performance of the group recommender system.

Declarations

Author contribution. All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.
Funding statement. None of the authors have received any funding or grants from any institution or funding body for the research.
Conflict of interest. The authors declare no conflict of interest.
Additional information. No additional information is available for this paper.

References

[1] Z. Fayyaz, M. Ebrahimian, D. Nawara, A. Ibrahim, and R. Kashef, "Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities," *Appl. Sci.*, vol. 10, no. 21, p. 7748, Nov. 2020, doi: 10.3390/app10217748.

[2] D. Bawden and L. Robinson, "Information Overload: An Introduction," in *Oxford Research Encyclopedia of Politics*, Oxford University Press, Jun. 2020. doi: 10.1093/acrefore/9780190228637.013.1360.

[3] H. Hanafi, N. Suryana, and A. S. H. Basari, "Dynamic convolutional neural network for eliminating item sparse data on recommender system," *Int. J. Adv. Intell. Informatics*, vol. 4, no. 3, p. 226, Nov. 2018, doi: 10.26555/ijain.v4i3.291.

[4] B. Alhijawi and Y. Kilani, "A collaborative filtering recommender system using genetic algorithm," *Inf. Process. Manag.*, vol. 57, no. 6, p. 102310, Nov. 2020, doi: 10.1016/j.ipm.2020.102310.

[5] Y. Li, K. Liu, R. Satapathy, S. Wang, and E. Cambria, "Recent Developments in Recommender Systems: A Survey," Jun. 2023, pp. 78–95. doi:10.48550/arXiv.2306.12680.

[6] S. Dara, C. R. Chowdary, and C. Kumar, "A survey on group recommender systems," *J. Intell. Inf. Syst.*, vol. 54, no. 2, pp. 271–295, Apr. 2020, doi: 10.1007/s10844-018-0542-3.

[7] Z. Huang, X. Xu, H. Zhu, and M. Zhou, "An Efficient Group Recommendation Model With Multiattention-Based Neural Networks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 31, no. 11, pp. 4461–4474, Nov. 2020, doi: 10.1109/TNNLS.2019.2955567.

[8] W. Wang, G. Zhang, and J. Lu, "Member contribution-based group recommender system," *Decis. Support Syst.*, vol. 87, no. July, pp. 80–93, Jul. 2016, doi: 10.1016/j.dss.2016.05.002.

[9] A. Felfernig, L. Boratto, M. Stettinger, and M. Tkalčič, "Algorithms for Group Recommendation," Springer, Cham, 2018, pp. 27–58. doi: 10.1007/978-3-319-75067-5_2.

[10] V. R. Yannam, J. Kumar, K. S. Babu, and B. K. Patra, "Enhancing the accuracy of group recommendation using slope one," *J. Supercomput.*, vol. 79, no. 1, pp. 499–540, Jan. 2023, doi: 10.1007/s11227-022-04664-4.

[11] M. I. Ardimansyah, A. F. Huda, and Z. K. A. Baizal, "Preprocessing matrix factorization for solving data sparsity on memory-based collaborative filtering," in *2017 3rd International Conference on Science in Information Technology (ICSITech)*, IEEE, Oct. 2017, pp. 521–525. doi: 10.1109/ICSITech.2017.8257168.

[12] J. Shi, B. Wu, and X. Lin, "A Latent Group Model for Group Recommendation," in *2015 IEEE International Conference on Mobile Services*, IEEE, Jun. 2015, pp. 233–238. doi: 10.1109/MobServ.2015.41.

[13] F. Ortega, A. Hernando, J. Bobadilla, and J. H. Kang, "Recommending items to group of users using Matrix Factorization based Collaborative Filtering," *Inf. Sci. (Ny).*, vol. 345, no. June, pp. 313–324, Jun. 2016, doi: 10.1016/j.ins.2016.01.083.

[14] X. Wang, Y. Liu, J. Lu, F. Xiong, and G. Zhang, "TruGRC: Trust-Aware Group Recommendation with Virtual Coordinators," *Futur. Gener. Comput. Syst.*, vol. 94, no. May, pp. 224–236, May 2019, doi: 10.1016/j.future.2018.11.030.

[15] X. Yuan, L. Han, S. Qian, G. Xu, and H. Yan, "Singular value decomposition based recommendation using imputed data," *Knowledge-Based Syst.*, vol. 163, no. January, pp. 485–494, Jan. 2019, doi: 10.1016/j.knosys.2018.09.011.

[16] R. Barathy and P. Chitra, "Applying Matrix Factorization In Collaborative Filtering Recommender Systems," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, IEEE, Mar. 2020, pp. 635–639. doi: 10.1109/ICACCS48705.2020.9074227.

[17] G. Ye and X. Zhao, "Improved SVD algorithm based on Slope One," in *2018 Chinese Control And Decision Conference (CCDC)*, IEEE, Jun. 2018, pp. 1002–1006. doi: 10.1109/CCDC.2018.8407276.

[18] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep Learning Based Recommender System," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–38, Jan. 2020, doi: 10.1145/3285029.

[19] R. Kusumaningrum, I. Z. Nisa, R. P. Nawangsari, and A. Wibowo, "Sentiment analysis of Indonesian hotel reviews: from classical machine learning to deep learning," *Int. J. Adv. Intell. Informatics*, vol. 7, no. 3, p. 292, Nov. 2021, doi: 10.26555/ijain.v7i3.737.

[20] Z. Batmaz, A. Yurekli, A. Bilge, and C. Kaleli, "A review on deep learning for recommender systems: challenges and remedies," *Artif. Intell. Rev.*, vol. 52, no. 1, pp. 1–37, Jun. 2019, doi: 10.1007/s10462-018-9654-y.

[21] G. Zhang, Y. Liu, and X. Jin, "A survey of autoencoder-based recommender systems," *Front. Comput. Sci.*, vol. 14, no. 2, pp. 430–450, Apr. 2020, doi: 10.1007/s11704-018-8052-6.

[22] D. Arpit, Y. Zhou, H. Q. Ngo, and V. Govindaraju, "Why Regularized Auto-Encoders learn Sparse Representation?," in *Proceedings of Machine Learning Research*, PMLR, Jun. 2016, pp. 136–144. Accessed: Mar. 01, 2026. [Online]. Available at: https://proceedings.mlr.press/v48/arpita16.html.

[23] Y. Li, J. Ren, J. Liu, and Y. Chang, "Deep sparse autoencoder prediction model based on adversarial learning for cross-domain recommendations," *Knowledge-Based Syst.*, vol. 220, no. May, p. 106948, May 2021, doi: 10.1016/j.knosys.2021.106948.

[24] R. Abolghasemi, E. H. Viedma, P. Engelstad, Y. Djenouri, and A. Yazidi, "A graph neural approach for group recommendation system based on pairwise preferences," *Inf. Fusion*, vol. 107, no. 17, p. 102343, Jul. 2024, doi: 10.1016/j.inffus.2024.102343.

[25] M. Srifi, A. Oussous, A. Ait Lahcen, and S. Mouline, "Recommender Systems Based on Collaborative Filtering Using Review Texts—A Survey," *Information*, vol. 11, no. 6, p. 317, Jun. 2020, doi: 10.3390/info11060317.

[26] H. Ko, S. Lee, Y. Park, and A. Choi, "A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields," *Electronics*, vol. 11, no. 1, p. 141, Jan. 2022, doi: 10.3390/electronics11010141.

[27] G. B. Martins, J. P. Papa, and H. Adeli, "Deep learning techniques for recommender systems based on collaborative filtering," *Expert Syst.*, vol. 37, no. 6, p. 12647, Dec. 2020, doi: 10.1111/exsy.12647.

[28] J. Du, L. Li, P. Gu, and Q. Xie, "A Group Recommendation Approach Based on Neural Network Collaborative Filtering," in *2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW)*, IEEE, Apr. 2019, pp. 148–154. doi: 10.1109/ICDEW.2019.00-18.

[29] F. M. Harper and J. A. Konstan, "The MovieLens Datasets," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Jan. 2016, doi: 10.1145/2827872.

[30] N. Idrissi and A. Zellou, "A systematic literature review of sparsity issues in recommender systems," *Soc. Netw. Anal. Min.*, vol. 10, no. 1, p. 15, Dec. 2020, doi: 10.1007/s13278-020-0626-2.

[31] Y. Koren, S. Rendle, and R. Bell, "Advances in Collaborative Filtering," in *Recommender Systems Handbook*, New York, NY: Springer US, Jan. 2022, pp. 91–142. doi: 10.1007/978-1-0716-2197-4_3.

[32] M. Bazargani, S. H.Alizadeh, and B. Masoumi, "Group deep neural network approach in semantic recommendation system for movie recommendation in online networks," *Electron. Commer. Res.*, vol. 26, no. 1, pp. 521–560, Feb. 2026, doi: 10.1007/s10660-024-09897-4.

[33] P. Li, Y. Pei, and J. Li, "A comprehensive survey on design and application of autoencoder in deep learning," *Appl. Soft Comput.*, vol. 138, no. May, p. 110176, May 2023, doi: 10.1016/j.asoc.2023.110176.

[34] G. Bathla, H. Aggarwal, and R. Rani, "AutoTrustRec: Recommender System with Social Trust and Deep Learning using AutoEncoder," *Multimed. Tools Appl.*, vol. 79, no. 29–30, pp. 20845–20860, Aug. 2020, doi: 10.1007/s11042-020-08932-4.

[35] K. Narayana Rao, K. Venkata Rao, and P. R. Prasad, "A hybrid Intrusion Detection System based on Sparse autoencoder and Deep Neural Network," *Comput. Commun.*, vol. 180, pp. 77–88, Dec. 2021, doi: 10.1016/j.comcom.2021.08.026.

[36] Y. Zhang, C. Zhao, M. Chen, and M. Yuan, "Integrating Stacked Sparse Auto-Encoder Into Matrix Factorization for Rating Prediction," *IEEE Access*, vol. 9, pp. 17641–17648, 2021, doi: 10.1109/ACCESS.2021.3053291.

[37] D. Bokde, S. Girase, and D. Mukhopadhyay, "Matrix Factorization Model in Collaborative Filtering Algorithms: A Survey," *Procedia Comput. Sci.*, vol. 49, no. 1, pp. 136–146, Jan. 2015, doi: 10.1016/j.procs.2015.04.237.

[38] M. N. M. Saputro and Z. K. A. Baizal, "Group Recommender System using Matrix Factorization Technique for Book Domain," *J. MEDIA Inform. BUDIDARMA*, vol. 7, no. 3, p. 1247, Jul. 2023, doi: 10.30865/mib.v7i3.6435.

[39] H. S. Ahmad, D. Nurjanah, and R. Rismala, "A combination of individual model on memory-based group recommender system to the books domain," in *2017 5th International Conference on Information and Communication Technology (ICoIC7)*, IEEE, May 2017, pp. 1–6. doi: 10.1109/ICoICT.2017.8074655.

[40] T. V. Vo and H. Soh, "Generation meets recommendation," in *Proceedings of the 12th ACM Conference on Recommender Systems*, New York, NY, USA: ACM, Sep. 2018, pp. 145–153. doi: 10.1145/3240323.3240357.