

S-VAERec: A Hybrid Music Recommendation Strategy Based on Generative Modelling and Music Similarity

Junzhe Zhao

School of Statistics, Renmin University of China, Beijing, China
zhaojunzhe950824@ruc.edu.cn

Abstract: In the era of big data, music platforms require advanced algorithms to meet growing recommendation demands. This paper proposes a hybrid music recommendation framework, S-VAERec (Similarity-Variational Autoencoder Recommender), leveraging the intrinsic characteristics of music. The framework combines content-based and social-based similarity metrics to calculate item-user scores and employs a Variational Autoencoder (VAE) to extract latent features from historical data, generating additional predictive scores. These scores are linearly weighted to produce a combined rating prediction, enhancing recommendation accuracy. Experiments on the Last.fm dataset demonstrate fast model convergence and improved performance: S-VAERec reduces prediction error (RMSE) compared to similarity-based algorithms and increases Recall compared to generative model-based approaches. By integrating the strengths of both methods, S-VAERec achieves high accuracy and strong generalisation, offering a robust solution for personalised music recommendations.

Keywords: recommendation algorithm, Variational Autoencoder (VAE), Deep learning, Similarity calculation

1. Introduction

The widespread use of social media and the continuous emergence of social software have made recommender systems a key focus. As these platforms become a primary information source, delivering high-quality, personalised content is crucial. Recommender systems, which filter and screen information to reduce overload, have evolved and are now applied across various fields, significantly impacting related industries. The need to process complex data has driven advancements in deep learning, which excels in analysing large, high-dimensional datasets and is widely used in recommender systems [1].

Within deep learning, recommendation algorithms can be classified into various types, including generative algorithms. These algorithms use existing data to approximate data distributions and generate new data for predictions [2]. In the context of the recommendation task, it is challenging to accurately describe and measure the user's perception and choice of information about each item with a conventional model. This perception is not independent, but rather a combination of different product information for selection. Consequently, a single discriminative model applied to a single item is difficult to accurately describe and complete the recommendation task. In contrast, the generative model has a unique advantage in the context of big data recommendation tasks based on its own characteristics.

In consideration of the aforementioned background, this paper focuses on music platforms and proposes a hybrid music recommendation algorithm that combines traditional methods with generative deep learning to address the limitations of single-method approaches and achieve personalised recommendations. The method integrates generative algorithms with content filtering based on similarity, tailored for complex data structures and fine-grained recommendations.

2. Related works

2.1. Recommendation algorithms

The research results pertaining to recommender systems that have been obtained today can be divided into two main categories: traditional recommender systems and recommender systems based on deep learning algorithms. In this section, we mainly concentrate on algorithms based on DL.

2.1.1. Discriminative model

Deep learning (DL), an advanced machine learning algorithm, is widely applied in recommendation systems. Discriminative deep learning-based recommendation systems can be categorised into four types: those based on deep neural networks (DNNs), recurrent neural networks (RNNs), convolutional neural networks (CNNs), and graph neural networks (GNNs).

DNNs, with additional hidden layers, extract user and item information layer by layer, identifying intrinsic data patterns during training [3]. This enhances the model's generalisation ability, improving predictions for uninteracted items. RNNs, particularly Long Short-Term Memory (LSTM), retain long-term and short-term sequences, integrating past observations for future predictions [4]. CNNs, with convolutional and pooling layers, extract features from raw data, reducing dimensionality and transforming unstructured data into structured formats to capture user characteristics [5]. GNNs, combining ideas from RNNs and CNNs, model graph-structured data, leveraging interaction information between nodes for high-quality recommendations. Research shows GNN-based models effectively exploit node interactions in graphs, demonstrating their potential in recommendation tasks [6].

2.1.2. Generative model

In recent years, a number of generative LLMs have attracted significant attention and been applied in various fields. Some researchers have utilised various generative deep learning models in the domain of recommender systems. Notably, the variational autoencoder (VAE)-based commodity recommender system, developed by Zhang [7], introduces a review-aware recommendation model grounded in a heterogeneous VAE framework. The study by X. Lu et al. employs generative adversarial networks to enhance the learning of data distribution, thereby facilitating a more effective alignment of user interests with corresponding items in the recommender system [8]. Wu integrates collaborative filtering and generative adversarial networks to enhance the original recommendation model in terms of physical resolvability and logical expressibility [9]. The continuous emergence of new generative algorithms and the enrichment of related research on fusion models have led to the constant updating and refinement of recommender systems based on generative algorithms. These related studies provide significant ideas and methods for the research in this paper.

2.2. Music recommendation strategies

In music recommendation, researchers have developed personalised algorithms by combining traditional and model-based approaches with the unique attributes of music. Chen et al. proposed an audio embedding method, calculating similarity between track and user embeddings to identify

suitable user groups, demonstrating strong generalisation [10]. Zeng designed the Look-like and K-means algorithms, improving personalised recommendation accuracy [11]. Wang and Zhang fused collaborative filtering with the XGBoost model, using feature extraction and engineering to enhance recommendation list accuracy [12].

Deep learning integration is evident in systems like Fan et al.'s MFCC and LSTM-based model [13], and Zhang's KGIFE [14], which combines Knowledge Graphs with Graph Neural Networks. Current research highlights key trends: (1) Music's personalised and social nature makes content-based similarity algorithms crucial [15]. (2) Deep learning models are widely used to uncover hidden patterns in recommendation data, leading to hybrid strategies like ResNet and 'song genes' [16]. (3) While discriminative models dominate, generative models are underutilised and require further exploration. There is a need for algorithms that balance music features and historical data structures, combining music similarity computation with generative deep learning models.

3. Methods

In order to further creatively accomplish the music recommendation task based on previous research, this paper proposes a hybrid music recommendation strategy, S-VAERec (Similarity-Variational Autoencoder Recommender), which combines the method of similarity calculation and the generative model, VAE, to give the recommendation results.

3.1. Similarity calculation

A review of the extant literature on music recommendation systems reveals that the conventional recommendation algorithm, predicated on content similarity, occupies a significant role in the development of such systems. The proposed methodology in this paper involves two primary steps. Firstly, the similarity between each item (artist) is calculated based on historical playback data. Secondly, the user's similarity is integrated to provide a recommendation score. This score is calculated by predicting the item's similarity (s_1) and the user's similarity (s_2).

The predicted score s_1 is then computed based on item similarity, necessitating the utilisation of tag (label) information for the music. Assuming p tag divisions of music styles on music platforms, for a given item (artist) i , certain users categorise it with tags. The total number of tags assigned by all participating users is counted, and the frequency with which item i was categorised under each tag is obtained, resulting in a p -dimensional vector $\mathbf{u}_i = (n_1, n_2, \dots, n_p)^T \in \mathbf{R}^{p \times 1}$, is referred to as the style vector of item i , thereby implying the content attributes of item i itself. Subsequent to obtaining the style vectors of all items accordingly, a measure of item similarity is obtained by calculating the Euclidean distance between the style vectors of two items. For instance, the measure of similarity between items i and j is given by

$$Sim_{ij} = \|\mathbf{u}_i - \mathbf{u}_j\|_2^2 \quad (1)$$

The smaller the value of Sim_{ij} , the higher the similarity between items i and j . In the recommendation process, for the item i in the history playlist of a certain user to be recommended, we arrange the items to be recommended by Sim_{ij} , ($j \neq i$). The items are arranged in descending order of similarity, and a predictive score of 0-5 is assigned according to this similarity level. The predictive score is then calculated based on the item similarity s_1 by averaging the predictive scores based on all the items in the user's historical playlists.

To compute the predicted score s_2 based on user similarity, we obtain the set B_i of items in the history playlists of all users who are friends with user i based on the user's friend relationship matrix. The items recommended for a user are assigned the highest ranked predictive score (5), while items

not recommended are assigned the lowest ranked predictive score (0). This results in the user similarity-based predictive score s_2 .

3.2. Variational autoencoder (VAE)

On the other hand, to comprehensively explore the profound information contained within the recommendation list, a recommendation algorithm is devised. This algorithm is founded upon a generative model, namely the Variational Autoencoder (VAE), which furnishes a model-based predictive score, designated as s_3 . The Variational Autoencoder is constructed upon the foundation of the Autoencoder (AE).

AE (Figure 1) is a neural network capable of reconstructing input data, comprising an encoder and a decoder. The encoder's function is to convert an input n -dimensional vector X into a h -dimensional vector Z through multiple layers of neural networks, utilising Z as the hidden feature of X . The primary purpose of the encoder is compression. The decoder, conversely, is tasked with the reverse process, converting the h -dimensional compressed vector Z back into the n -dimensional reconstructed vector X^* via a second neural network. The primary function of the decoder is to reconstruct the X^* . The objective of constructing a self-encoder is to extract the hidden features embedded in the input data, and it is hoped that the extracted Z can reflect the information of X to the greatest extent possible, while the hidden features embedded in Z are expressed by the X^* reconstructed by the decoder. Therefore, the objective of training the self-encoder is to ensure that X and X^* are as similar as possible, i.e. to make the input equal to the output as far as possible. On the basis of this property, autoencoders have been extensively utilised in image recognition and other domains that necessitate the management of high-dimensional, unstructured data.

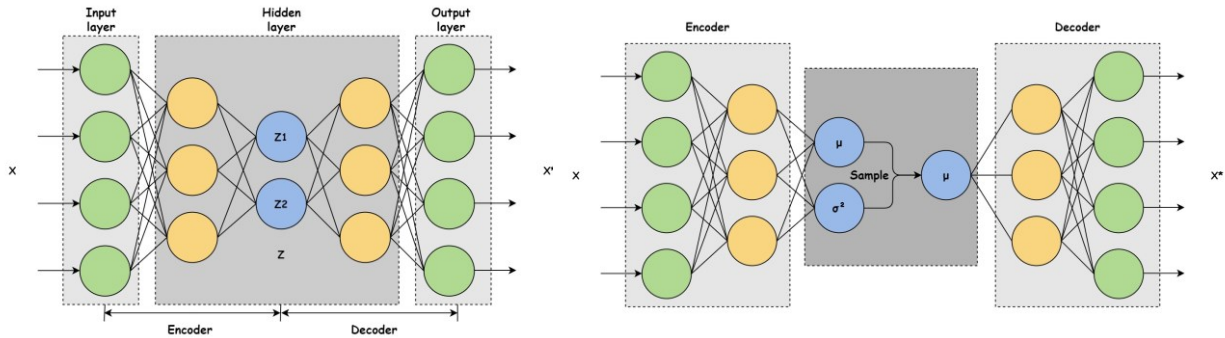


Figure 1: Autoencoder (AE) and variational autoencoder (VAE) model architecture

VAE (Figure 1) can be regarded as a generalisation of the AE, insofar as it employs an encoder to model the distribution of the hidden variable z and samples from a specific distribution to obtain the hidden variable z . The final step in the VAE process involves reducing z to the reconstructed data. In the structure of the VAE, the encoder is also termed the inference network, denoted as $q_{\theta_l}(z|x)$. In this case, the original data is introduced into the neural network to compute the conditional distributions obeyed by its corresponding hidden variables. It is assumed that the hidden variables obey a multivariate normal distribution with independence of each component. In this section, we will be examining the hypothesis that the hidden variables are distributed as an independent multivariate normal distribution, i.e. $N(\mu, \sigma^2 I)$. Here, I is the n -order unitary matrix, and n is the dimensionality of the hidden variables. The outputs of the inferred network are the mean and the variance of the normal distribution. In the following step, the hidden variable z is obtained by sampling from the aforementioned distribution. The final step involves transforming the hidden

variable z obtained from the sampling process into the reconstructed data x^* using a decoder, i.e., the generative network $p_{\theta_2}(x^*|z)$.

As demonstrated above, the objective of parameter optimisation for this model is twofold: Firstly, to ensure that the input x and the output x^* are sufficiently close. Secondly, given the assumption that the hidden variable z follows a specific distribution, it may be advantageous to impose the requirement that this distribution be multivariate standard normal. That is to say, the objective is to ensure that $N(\mu, \sigma^2 \mathbf{I})$ is as close as possible to $N(0, \mathbf{I})$.

Firstly, for each user's rating vector $x = (x_{i1}, x_{i2}, \dots, x_{ip}) \in \mathbf{R}^{p \times 1}$, a distribution $N(\mu, \sigma^2 \mathbf{I})$ is obtained through an encoder $q_{\theta_1}(z|x)$, from which a hidden variable $z_{(h \times 1)}$ is sampled, followed by a generative network to obtain a reconstructed rating vector $x_{i(p \times 1)}^*$. The loss calculation of the model is divided into two parts, the first of which is used to represent the difference between x and x^* , i.e.

$$l(x_i, x_i^*) \quad (2)$$

The corresponding loss function of the model is given by l . The second part is used to represent the difference between the distribution of the hidden variable (denoted as $p(z)$) and the standard normal distribution (denoted as $\Phi(z)$). This, combined with the principle of variational inference, is measured by the Kullback-Leibler Divergence of the two distributions, i.e.

$$D_{KL}(p(z)||\Phi(z)) = \int p(z) \log \frac{p(z)}{\Phi(z)} dz \quad (3)$$

Therefore, the optimisation objective for the model parameters is:

$$\min_{\theta} \sum_{i=1}^N (l(x_i, x_i^*) + D_{KL-i}) \quad (4)$$

Algorithm 1: VAE Training

Input: Epoch B , Loss function l , Initialising model parameters θ_1, θ_2 , Learning rate lr , Score matrix X .

Output: Parameters of the trained model θ_1, θ_2 .

- 1: **For** b in 1: B **do**
 - 2: Forward propagation to calculate $q_{\theta_1}(z|x_i)$ and obtain the mean μ and variance σ^2 of the normal distribution to which the hidden variables are subjected.
 - 3: Sample ϵ_i from $N(0, \mathbf{I})$ and compute the hidden variable $z_i = \mu + \sigma \odot \epsilon_i$.
 - 4: Calculate the loss corresponding to the current parameter $L(\theta) = \sum_{i=1}^N (l(x_i, x_i^*) + D_{KL-i})$.
 - 5: Calculate the gradient corresponding to the parameter $\frac{\partial L}{\partial \theta_1}, \frac{\partial L}{\partial \theta_2}$.
 - 6: Update parameters $\theta_1 \leftarrow \theta_1 - \frac{\partial L}{\partial \theta_1} \cdot lr, \theta_2 \leftarrow \theta_2 - \frac{\partial L}{\partial \theta_2} \cdot lr$.
 - 7: **End until conversion**
-

3.3. Hybrid strategy

Once the model has been constructed, a vector of the user's historical ratings is input, and the reconstructed output, a vector containing the ratings of the uninteracted items, is the model's prediction of the score s_3 . Combining the s_1 and s_2 obtained in the previous section, the final predicted score *Score* is obtained using linear weighting, with the following expression:

$$Score = \omega_1 \cdot s_1 + \omega_2 \cdot s_2 + \omega_3 \cdot s_3 \quad (5)$$

where $\omega_1 + \omega_2 + \omega_3 = 1$. Our hybrid strategy is shown in figure 2.

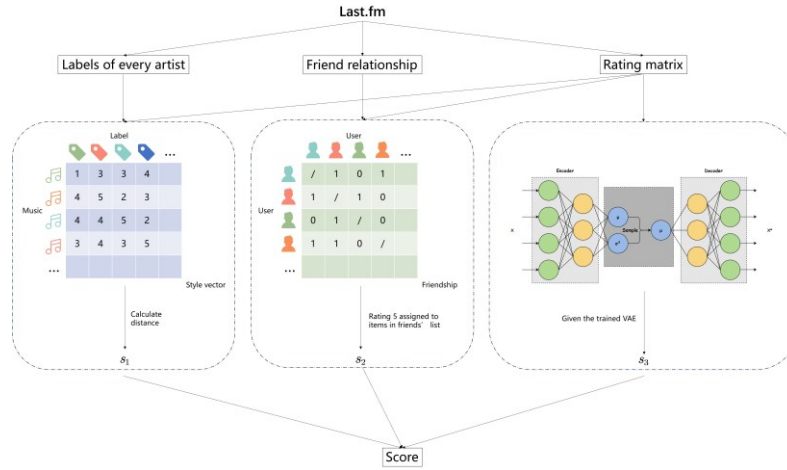


Figure 2: Hybrid strategy: S-VAERec

4. Materials and experiments

4.1. Data and pre-processing

The study uses the Last.fm music recommendation dataset, which captures user listening habits and artist works, released during RecSys 2011 and HetRec 2011. Key data files include ‘user_artists.dat’, providing user-artist listening counts; ‘user_taggedartists.dat’, containing user-assigned tags for calculating style vectors; and ‘user_friends.dat’, detailing user friendships for predicting similarity-based ratings. Table 1 summarises the dataset's basic information.

Table 1: Last.fm music recommendation dataset basics

Number of Users	Number of Item	Number of Interactions	Sparsity
1892	17632	92834	99.7%

The 'weight' value in the data, which represents the number of listens, has a wide range of values (0~352698), which precludes the measurement of user ratings of items. Consequently, the 'weight' value is transformed into a rating scale (1-5) for the interaction. The 20%, 40%, 60%, and 80% quartiles of the 'weight' variable are 83, 188, 358, and 764, respectively. The variable 'rating' is constructed according to the classification criteria outlined in Table 2, with the aim of representing the rating of the interaction.

Table 2: Grade scoring criteria

Weight	Rating
$0 \leq weight < 83$	1
$83 \leq weight < 188$	2
$188 \leq weight < 358$	3
$358 \leq weight < 764$	4
$weight \geq 764$	5

In this study, the recommendation process of the recommendation system is transformed into a process of predicting rating.

The dataset is then partitioned, with 80% of the interaction data being randomly selected as the training set to train the model, while the remaining 20% of the data is used as the test set to validate and test the model. The user-item score matrix is constructed based on these data.

The s_1 and s_2 of each user are then calculated according to the method outlined in Section 3.1, with the data from 'user_taggedartists.dat' and 'user_friends.dat' being utilised for this purpose. The following weights are assigned to them: The weights of linear weighting are temporarily set to (0.8,0.1,0.1) to obtain the model outputs on the test set, which are used to validate the model performance during the training process.

4.2. Experiment design

In the following section, the implementation of VAE is presented. The Pytorch framework has been utilised for this purpose, and the experimental environment is illustrated in Table 3.

Table 3: Experimental environment

Environments	Setup
CPU	Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz
Memory	30G
GPU	NVIDIA RTX A2000
Video Memory	12G
Python	3.7
Pytorch	1.8.1

The specific structure of the VAE model with the relevant parameters is shown in Figure 3.

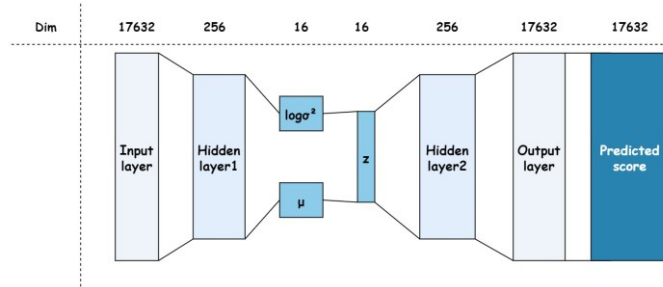


Figure 3: Relevant model parameters of VAE

Training is performed using Adam as optimizer at a learning rate of 0.0001, with 200 epochs.

4.3. Evaluation metrics

After training, the results of the training need to be evaluated on the test set. To this end, we propose the following evaluation metrics (where x_i, x_i^* represent the true and predicted ratings of user i , respectively):

(1) Squared Error (MSE)

$$MSE_i = \frac{1}{N_i} \sum_{j=1, j \in R_i}^p (x_{ij} - x_{ij}^*)^2 \quad (6)$$

It should be noted that only the rating differences on the interacting items are considered in calculating this value, and N_i is the number of interacting items.

(2) Root Mean Square Error (RMSE)

$$RMSE_i = \sqrt{\frac{1}{N_i} \sum_{j=1, j \in R_i}^p (x_{ij} - x_{ij}^*)^2} \quad (7)$$

RMSE is the arithmetic square root of MSE.

(3) Mean Absolute Error (MAE)

$$MAE_i = \frac{1}{N_i} \sum_{j=1, j \in R_i}^p |x_{ij} - x_{ij}^*| \quad (8)$$

(4) *Hit@k*

Top-k hit rate is calculated as follows: If at least one of the first k items $E_i^{(k)}$ recommended by the recommender system for the user $i (i = 1, 2, \dots, N)$ is among the first k items $D_i^{(k)}$ which is the user's real favourite, then the *Hit@k* is 1 for that user, otherwise it is 0. The formula is as follows:

$$Hit_i@k = I[E_i^{(k)} \cap D_i^{(k)} \neq \emptyset] \quad (9)$$

where I represents the indicative function. This metric measures whether at least one of the items in the recommendation list is actually preferred by the user.

(5) *Recall@k*.

Recall@k represents the relative amount of items in $E^{(k)}$ that are in the top k items that the user really likes, and is calculated as:

$$Recall_i@k = \frac{card(E_i^{(k)} \cap D_i^{(k)})}{k} \quad (10)$$

where *card* represents the number of elements in the corresponding set. This metric measures the goodness of the recommended results in more detail than the hit rate.

(6) Normalised Discounted Cumulative Gains (NDCG)

Relevant literature points to the Normalised Discounted Cumulative Gains (NDCG) method as a way of evaluating the quality of recommendation ranking. The definition of NDCG in this paper is the same as that in the paper by Jarvelin et al [17].

5. Results and discussion

In order to visualise the training process of our S-VAERec model, we plotted a line graph of the training error versus the testing error as a function of the number of training rounds, as shown in Figure 4. (only the first 100 rounds are shown). From the figure, it can be concluded that both from the training set and the test set, the error decreases with the training process, and both of them have reached stability before 25 rounds, indicating that the model has converged, and the speed of reaching convergence is relatively fast.

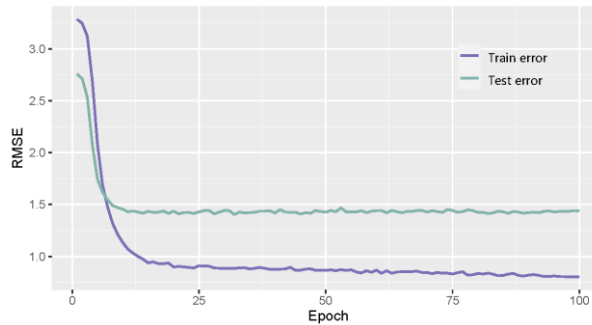


Figure 4: Model training process

As mentioned in 3.3, our final predictive score is obtained by linearly weighting the item similarity-based predictive score s_1 , the user similarity-based predictive score s_2 and the generative model's predictive score s_3 , i.e.

$$Score = \omega_1 \cdot s_1 + \omega_2 \cdot s_2 + \omega_3 \cdot s_3 \quad (11)$$

where the linearly weighted weights are a hyperparameter of interest. To explore the effect of the weighting hyperparameter on the prediction results, we adjust the value of ω_3 by adjusting the value of the other two weights and setting the value of the other two weights to be

$$\omega_1 = \omega_2 = 0.5 \times (1 - \omega_3) \quad (12)$$

Based on several sets of parameters obtained by this method, we calculate their corresponding prediction scores separately and give the recommendation results. In order to examine its recommendation effect, we calculate the mean value of each evaluation index on the test set under different weight settings respectively, and the results are shown in Figure 5.

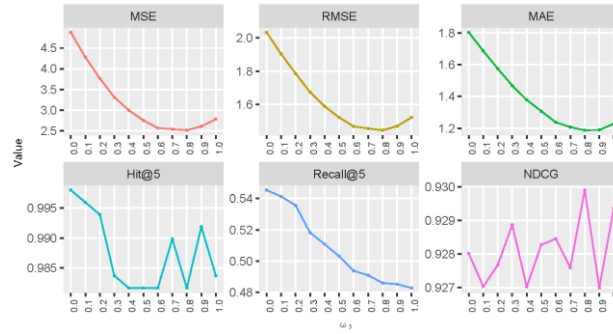


Figure 5: Values of each evaluation metric on the test set with different weight settings

It is evident that all three metrics (MSE, RMSE, MAE) employed to assess the prediction accuracy of ratings demonstrate decreasing and then increasing trend as ω_3 changes from small to large. This finding suggests that utilising similarity calculation exclusively for recommendation purposes results in elevated prediction error for ratings. On the contrary, incorporating the outcomes of generative models enhances prediction accuracy. However, if the weight assigned to the results of generative models is excessively high, it leads to an escalation in error. In this context, the optimal value of the parameter, designated as ω_3 , is estimated to be approximately 0.8.

Conversely, an augmentation in the weights of similarity calculations results in a marginal enhancement in the value of Recall, while no discernible pattern emerges in the alterations of Hit and NDCG. The value of 0.8 for the parameter w_3 was determined through empirical analysis, and the resulting values for each evaluation metric are presented in Table 4. This analysis substantiates that S-VAERec model generates more precise ratings and furnishes recommendation lists of superior quality.

Table 4: Values of evaluation metrics with $w_3 = 0.8$

Metrics	Values
MSE	2.51
RMSE	1.45
MAE	1.19
Hit@5	0.98
Recall@5	0.49
NDCG	0.93

6. Conclusion

In this paper, we propose a hybrid music recommendation strategy, S-VAERec, which is based on a generative model and music similarity. The strategy generates the final recommendation list by performing a linear weighted calculation of the similarity and prediction scores of the VAE. The generative model enhances the intrinsic regularity of the data and improves the prediction accuracy, while the similarity calculation boosts the recall. The strategy aims to balance the advantages of both to provide higher quality recommendations. At present, S-VAERec utilises a single generative model; however, future research will explore multi-model fusion to adapt to complex recommendation scenarios and enhance recommendation accuracy and efficiency. Concurrently, we will monitor the development of new deep learning technologies and integrate them into recommendation algorithms to optimise recommendation methods under big data analysis.

References

- [1] Zhou, J., & Lu, W. (2023). *Data Science and Big Data Technology Series: Deep Learning Based on PyTorch Implementation*.
- [2] Dong, Y., Fan, S., Zhang, Y., et al. (2023). Development and challenges of generative adversarial networks. *Signal Processing*, 154-175.
- [3] Nguyen, T.-D. (2024). An approach to improve the accuracy of rating prediction for recommender systems. *Automatika: Journal for Control, Measurement, Electronics, Computing & Communications*, 65(1), 58–72.
- [4] Huang, J., Zhao, W. X., Dou, H., et al. (2018). Improving sequential recommendation with knowledge-enhanced memory networks, 505-514.
- [5] Chen, H., Fu, J., Zhang, L., et al. (2019). Deformable convolutional matrix factorization for document context-aware recommendation in social network. *IEEE Access*, 7, 66347-66357.
- [6] Zhang, J. (2024). *Research on recommendation system based on graph neural network [Doctoral dissertation, Zhengzhou Light Industry University]*.
- [7] Zhang, J. (2022). *Research and Implementation of Commodity Recommendation System Based on Variational Self-Encoder [Master's thesis, Zhongnan University of Economics and Law]*.
- [8] Lu, X., Liu, J., Gan, S., Li, T., Xiao, Y., & Liu, Y. (2022). Recommendation model based on dynamic interest group identification and data compensation. *IEEE Transactions on Network and Service Management*, 19(1), 89–99. <https://doi.org/10.1109/TNSM.2021.3112702>
- [9] Wu, J. (2022). *Research on Algorithm Improvement and Recommendation System Design Based on Collaborative Filtering and Generative Adversarial Network [Master's thesis, Guangdong University of Technology]*.
- [10] Chen, K., Liang, B., Ma, X., et al. (2021). Learning audio embeddings with user listening data for content-based music recommendation. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 3015-3019). IEEE.
- [11] Zeng, D. (2023). Design of a music recommendation system based on look-alike and k-means algorithms. In *2023 International Conference on Evolutionary Algorithms and Soft Computing Techniques (EASCT)* (pp. 1-5). IEEE.
- [12] Wang, F., & Zhang, G. (2024). An application of XGBoost with collaborative filtering on music push. *Science and Technology Innovation and Application*, 49-52.
- [13] Fan, K., & Hu, Y. (2024). Research on music recommendation system based on LSTM model. *Electroacoustic Technology*, 136-138.
- [14] Zhang, B. (2023). *Music Recommendation System Based on Knowledge Graph Enhanced Graph Neural Network and Sentiment Analysis [Master's thesis, Nanjing University of Posts and Telecommunications]*.
- [15] Reddy, S. V., Krishna, V. R., Sapkal, R. J., Dhanke, J., Waghmare, S. P., & Kumar, K. (2024). A cutting-edge artificial intelligence paradigm for entertainment-infused music recommendations. *Entertainment Computing*, 51. <https://doi.org/10.1016/j.entcom.2024.100717>
- [16] Rao, W. (2024). *Design and Implementation of Music Recommendation System Based on Hybrid Strategy [Master's thesis, Nanchang University]*.
- [17] Jarvelin, K., & Kekalainen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4).