2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Research on Movie Service Recommendation Algorithms Incorporating Film Attributes and Multimodal Information

Chuanxi Liu[1], Jing Li[1], Chengfan Jiao[1], Ming Zhu[1] (✉), and Wenxuan Liu[1]

[1] School of Computer Science and Technology, Shandong University of Technology, Zibo 255000, China
zhu_ming@sdut.edu.cn

Abstract. As a personalized recommendation technology, the recommendation system aims to predict users' preferences for items and provide recommendation services for users. Movie recommendation technology can help users quickly find their preferred movies and thus meet their viewing needs. Traditional context-based movie recommendation models only use text data, obtaining limited information from single-modal data and failing to fully address the problem of data sparsity. This paper proposes a multimodal movie recommendation model (Layered Multi-head Attention Dynamic Graph, LMADG) that integrates text and image data, aiming to capture the dynamic changes in user interests and the graph structure information of user-movie interactions. By combining Graph Convolutional Network (GCN) and temporal attention mechanisms, LMADG can effectively extract the temporal features of users and movies and generate personalized recommendation results. Finally, comparative experiments are conducted on the Movielens-1M, TMDB, and Netflix Prize datasets, verifying that the proposed model has better recommendation quality.

Keywords: Multi-modal, Graph convolutional network, Movie recommendation, Temporal attention mechanism

1 Introduction

In the Internet era, in order to effectively mine useful information for users, recommender systems have been widely used in many fields [1-4], and and have also become a research hotspot for alleviating the problem of information overload. Due to data sparsity, the performance of traditional recommendation algorithms is greatly limited [5]. Therefore, solving the problem of rating data sparsity is of great significance to improve the performance of the recommendation system. Traditional recommendation algorithms usually only consider ratings, and if the rating matrix is sparse, the performance of the algorithm will be negatively affected [6]. To enhance the performance of recommendation algorithms, some studies use auxiliary information such as movie attributes and movie reviews in the recommendation model of recommendation systems [7,8]. However, single-modal text data contains limited information and cannot effectively deal with the problem caused by data sparsity. In fact, image information has a huge influence on user preferences and plays a crucial role in improving the performance of the recommendation system. The LMADG proposed in this paper fully integrates text and image features, which brings significant results for the improvement of the accuracy of the recommendation system.

2 Related theories

2.1 Recommender systems

With the rapid advancement of deep learning technologies, deep learning algorithms are capable of effectively extracting the latent representations of auxiliary information, thereby enhancing the accuracy of recommendation rating predictions, so they are widely used in recommendation systems [9]. At present, there are three main types of recommendation system models: one is content-based recommendation model; The second is the recommendation model based on collaborative filtering. The third is a hybrid recommendation model. The user-based collaborative filtering algorithm (UserCF) employs a clustering-based approach to compute similarity metrics. [10]. T The core concept of the user-based collaborative filtering algorithm is to recommend items to the target user that have been favored by other users but with which the target user has not yet interacted, and the target user has similar interests and hobbies with other users, as shown in Fig. 1. The item-based collaborative filtering algorithm (ItemCF) relies on the common rating value of users for items to calculate the similarity [11]. However, Due to the low overlap in items purchased by different users, it becomes challenging for the algorithm to identify users with similar preferences, and there is a data sparsity problem. In addition, the required storage space keeps increasing as the number of users increases. In order to find similar users quickly, UserCF needs to spend a lot of storage overhead to maintain the user similarity matrix. The idea of item-based collaborative filtering algorithm is to recommend similar items to users based on their favorite items, and calculate the similarity between items using the user's historical preference data, as shown in Fig. 2. However, collaborative filtering algorithm is difficult to apply the similarity of items to other items, resulting in weak generalization ability. To solve this problem, Matrix Factorization was proposed in 2006, which uses more dense latent vectors to represent users and items, and mines hidden features based on collaborative filtering co-occurrence matrix, which can alleviate the problem of data sparsity to some extent.

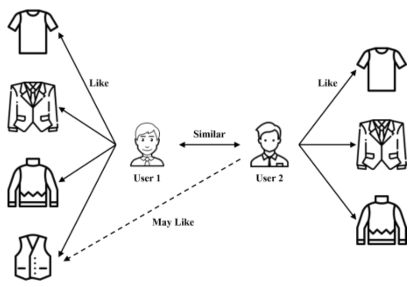

Fig. 1. User-based collaborative filtering algorithm

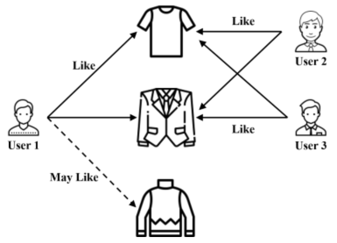

Fig. 2. Item-based collaborative filtering algorithm

There are some limitations in purely using collaborative filtering methods, which is because the recommendation technology of collaborative filtering has the cold start problem. One of the current solutions is to combine the advantages of various recommendation algorithms and use hybrid model recommendation method. Hybrid recommendation can be combined with real-time recommendation system by using weighted hybrid method [12], and the recommendation result of short-term preference is obtained by calculating, and it is assigned the highest weight and placed at the top of the recommendation list. In recent years, complex models such as deep learning have been widely used in industry, and the integration of multiple data into a model or framework has also played a significant role in recommendation systems. The hybrid recommendation method can also address the user cold-start problem through a hybrid strategy [13].

2.2 Word Embedding method

Word Embedding [14] is a technique that maps words from a high-dimensional space to a low-dimensional space. Word embedding methods represent words as real-valued vectors, and enable similar words to possess similar vector representations in the embedding space by capturing their semantic and syntactic relationships. Popular word embedding methods include distributed word vector encodings (Word2Vec, GloVe) and pre-trained models (BERT) [15].

BERT (Bidirectional Encoder Representations from Transformers) [16], a transformer-based pre-trained language model, was proposed by Google in 2018. Different from the traditional word embedding methods (Word2Vec, GloVe), BERT can capture the semantic information of words in different contexts. The Transformer model in it uses the Self-Attention Mechanism to capture the relationship between different positions in the input sequence. An encoder-decoder structure is employed to address the sequence-to-sequence mapping. Fig. 3 shows the structure of the BERT pre-trained model.

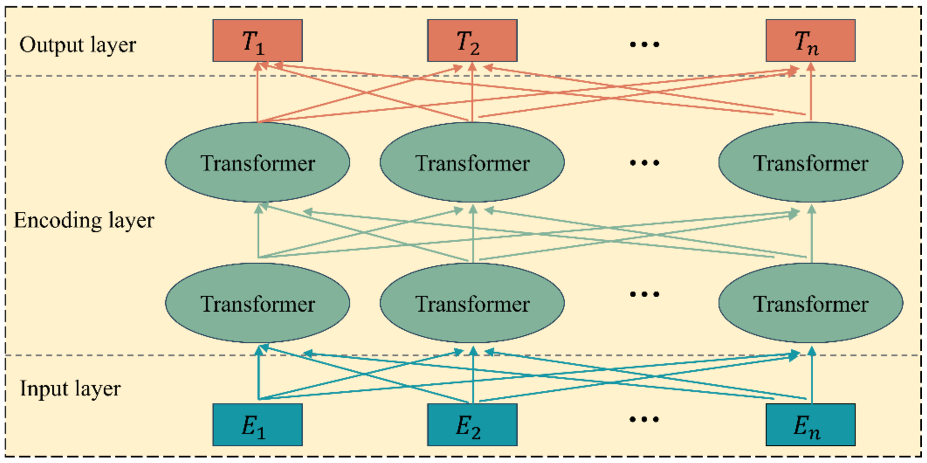

Fig. 3. BERT Model Architecture

In the pre-training phase of BERT, the model learns context-sensitive representations of words by unsupervised training on large-scale text corpus. The pre-training task included two stages: MLM (Masked Language Model) [17] and NSP (Next Sentence Prediction) [18]. In MLM, part of the words of the input sequence were randomly masked, and the masked words needed to be predicted by other words in the context. This task enables the model to understand the context information of words and learn the semantic relationship between words. By receiving two sentences as input, NSP needs to determine whether the two sentences are consecutive, and this task enables the model to understand the relationship between sentences. The BERT model comprises multiple layers of Transformer encoders. Since BERT is pre-trained, only the encoder part is usually used. BERT uses Multi-Head Self-Attention to capture relationships in the input sequence [19]. Also included are structures such as Feedforward Neural Network layer and Residual Connections.

2.3 Graph Neural Networks

Graph Neural Network (GNN) [20] is a type of deep learning model designed to process graph data, where a graph is a network structure composed of nodes and edges. Graph neural networks are able to learn node features on a graph and propagate and aggregate them based on the structure of the graph to generate new feature representations for nodes, edges, or the entire graph. Graph Convolutional Network (GCN) is a typical representative of GNN, which is a deep learning model for graph-structured data. It aims to learn the representation of nodes in the graph structure, so that these representations can capture the neighbor structure and global topology information of nodes in the graph. GCN utilizes the adjacency matrix of the graph to represent the connection relationships between nodes and leverages the node feature matrix to represent the features of each node. The representation of a node is updated by aggregating the features of a node and the features of its neighbor nodes.

Consider a batch of graph data, there are N nodes, each node contains its own features, these node features form an $N \times D$ dimensional matrix X, the relationship between the various nodes form an $N \times N$ dimensional adjacency matrix A, matrix X and adjacency matrix A are used as the input of the model. The propagation mode between the middle layer and layer of GCN is as follows.

$$H^{(l+1)} = \sigma\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{1}$$

where $\widetilde{D}$ denotes the degree matrix of A, that is, the degree of each node. A is the adjacency matrix of the graph. The adjacency matrix A is added to the identity matrix I to obtain $\tilde{A}$ to prevent information loss during node information propagation. $H^{(l)}$ represents the input features and $W^{(l)}$ represents the parameter matrix. $\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$ is used for normalization to avoid the information imbalance problem during information aggregation. Through the stacking of multi-layer GCN, each node can aggregate the information of its neighbor nodes, so as to extract deeper features to better obtain graph structure information.

2.4 Attention mechanism

Attention Mechanism [21] dynamically assigns weights to better focus on important parts of the input data. The basic idea is to dynamically calculate a weight vector in the input sequence to represent the importance of each input element. The weight vector is usually the same length as the input sequence and is normalized so that the sum of all weights is 1. These weights are then weighted and summed with the input sequence to obtain a weighted representation where important parts get higher weights and less important parts get lower weights.

Multi-head self-attention [22] contains three parts, which are the target word feature vector Q, the context feature vector K, and the original vector V. The weights obtained from the similarity calculation of Q and K are applied to reconstruct the original vector V, and the formula is given below.

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

$$Attention - output = Attention(Q, K, V) \tag{3}$$

3 Multimodal model design

3.1 Multimodal alignment and fusion

Film review websites contain rich text and image information, providing valuable references for users' browsing. In the existing multimodal representation learning methods, the embedding features of images and texts are respectively in their own spaces due to the large amount of image and text features, making it difficult for multimodal encoders to learn and model the interaction between images and texts. To address this challenge, this paper adopts a method of aligning images and texts before fusion and introduces a contrastive loss function (image-text contrastive loss) to align the representations of images with those of texts before their embedding fusion. The framework of multimodal alignment and fusion is shown in Fig. 4.

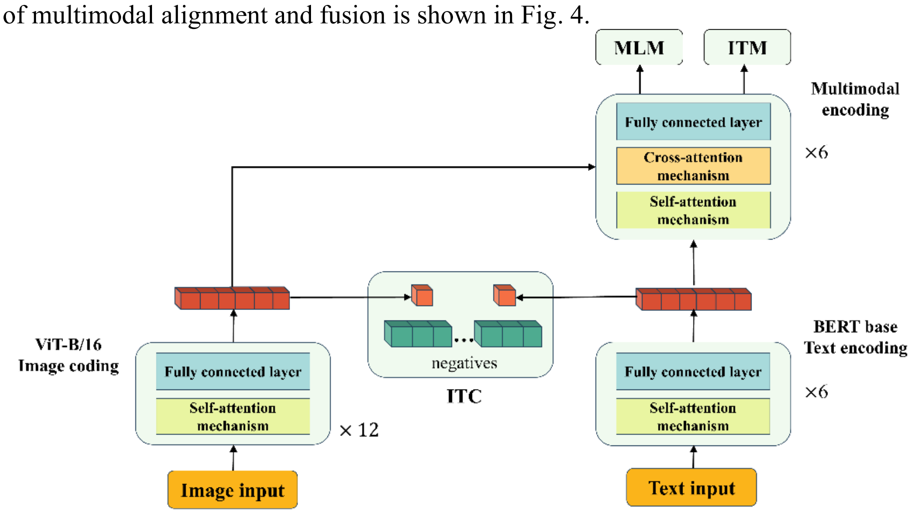

Fig. 4. Diagram of multimodal alignment and fusion framework

The multimodal alignment and fusion framework consists of three parts: image encoding, text encoding, and multimodal encoding. The image encoding uses a 12-layer ViT-B/16 model, the text encoding adopts the first 6 layers of the transformer in the BERT model, and the multimodal encoding uses the last 6 layers of the transformer. Firstly, the vectors of the image and text are obtained after passing through the fully connected layer and softmax. The classification tokens of the image and text are input into the same space for alignment. Then, in the multimodal transformer, the cross-attention mechanism in the multimodal encoding is used to fuse the tokens of the image and text to represent the features of the image-text fusion. Finally, the fully connected layer and softmax are connected to predict whether the image-text pair matches. The final objective function is trained by calculating three loss functions, and the introduction of each loss function is as follows:

ITC (Image-Text Contrastive Learning) is the image-text contrastive loss. The image is processed by ViT to obtain the image token v_{cls}, and the text is processed by BERT base to obtain the text token w_{cls}. The original tokens obtained from the image and text encodings are respectively projected from 1*768 dimensions to 1*256 dimensions through fully connected layers, and then normalized by softmax to obtain the vectors g_{v} and g_{w}. The formula for the similarity between the image and text is as follows:

$$s(I,T) = g_v(v_{cls})^{\mathrm{T}} g_w(w_{cls}) \tag{4}$$

$$s(T,I) = g_w(w_{cls})^{\mathrm{T}} g_v(v_{cls}) \tag{5}$$

Then, the similarity between the image and the text is compared by calculating the cosine similarity between the normalized projection vectors g_v and g_w. The goal of contrastive learning is to maximize the similarity scores between positive sample pairs while minimizing the similarity scores between negative sample pairs. Compared with other unmatched negative samples, the positive sample pairs of matching images and texts hope to obtain a higher similarity. The similarity calculation formulas from image to text and from text to image are as follows:

$$p_m^{\mathrm{i2t}}(I) = \frac{\exp\left(\frac{s(I,T_m)}{\tau}\right)}{\sum_{m=1}^{M} \exp\left(\frac{s(I,T_m)}{\tau}\right)} \tag{6}$$

$$p_m^{\mathrm{t2i}}(T) = \frac{\exp\left(\frac{s(T,I_m)}{\tau}\right)}{\sum_{m=1}^{M} \exp\left(\frac{s(T,I_m)}{\tau}\right)} \tag{7}$$

where τ represents the temperature coefficient, which is used to adjust the smoothness of the similarity score in the loss function. Let $y^{i2t}(I)$ and $y^{t2i}(T)$ be the corresponding GT (Ground Truth) labels, then the ITC objective function is the cross-entropy loss function between p and y, as shown in the following formula:

$$L_{itc} = \frac{1}{2} E_{(I,T)\sim D}\left[H\left(y^{i2t}(I), p^{i2t}(I)\right) + H\left(y^{t2i}(T), p^{t2i}(T)\right)\right] \tag{8}$$

MLM (Masked Language Modeling) is a pre-training task of BERT. It predicts the masked words by using images and context texts. With a 15% probability, the tokens of the input text are processed. Among them, 10% are randomly replaced with other tokens, 80% are replaced with [MASK] tokens, and the remaining 10% remain unchanged. Finally, the processed sequence $\hat{T}$ is obtained. Then, the cross-entropy loss function is used for learning, as shown in the following formula:

$$L_{mlm} = E_{(I,\hat{T})\sim D} H\left(y_{msk}, p_{msk}(I,\hat{T})\right) \tag{9}$$

ITM (Image-Text Matching) is a loss function for image-text matching, used to predict whether an image and text are matched. The first CLS token of the final output sequence is input into a fully connected layer to determine whether the input image and text are matched. Cross-entropy loss function is adopted for training, and the formula is as follows:

$$L_{itm} = E_{(I,T)\sim D} H\left(y^{itm}, p^{itm}(I,T)\right) \tag{10}$$

The final pre-training objective function is as follows:

$$L = L_{itc} + L_{mlm} + L_{itm} \tag{11}$$

4 Experiment and Analysis

4.1 Experimental data

To verify the effectiveness of the movie recommendation method based on knowledge graph and graph attention network proposed in this paper, multiple datasets were selected for experimental evaluation, including the ml-20m sub-dataset of the MovieLens1M dataset, the TMDB dataset and the NetflixPrize dataset. Among them, the ml-20m dataset is often used in movie recommendation services, containing 25,000,095 ratings and 1,093,360 tags from 162,541 users from 1995 to 2019, with a total of 62,423 movies. Each user has rated at least 20 movies. In this paper, 26,376 movies and 31,043 user ratings were selected as the interaction data between users and movies. The TMDB + NetflixPrize dataset includes 4,374 movie titles, 20 movie genres, 1,616 director names, 3,728 movie keywords, 9,253 users and 29,671 user ratings. The initialization of each parameter adopts the Xavier initialization method, and the Adam optimizer is used to optimize the parameters in the model, with a learning rate of 0.0001. In the Adam optimizer, the values of β1 and β2 are set to the recommended parameters 0.9 and 0.999 in the original Adam paper. The size of the Embedding vector is set to 64. The number of epochs is set to 20. The TMDB + Netflix Prize dataset is shown in the table 1.

Table 1. TMDB + Netflix Prize Dataset

Dataset	Number of movies	Number of genres	Number of directors	Number of keywords	Number of users	Number of ratings
TMDB+Netflix Prize	4374	20	1616	3728	9253	29671

4.2 Experimental setup

In terms of hardware, model training and inference default to using a GPU, and the CUDA device is specified through ‘args.device’; if a GPU is unavailable, the code automatically switches to running on a CPU. Regarding the software environment, the code is implemented based on PyTorch, leveraging its efficient tensor computations and automatic differentiation capabilities. It also relies on numpy and scipy for scientific computing and uses sklearn to calculate evaluation metrics.

4.3 Evaluation indicators

This paper adopts two evaluation metrics: $NDCG@K$ and $Recall@K$. Where k represents the assessment of the top k predicted results and can be set to different values such as 20, 40, 60, 80, and 100. In this paper, k is set to 20. The specific calculation methods and meanings of the evaluation metrics are as follows:

NDCG (Normalized Discounted Cumulative Gain) represents a ranking metric of List-Wise, and its calculation formula is as follows:

$$NDCG@K = \frac{DCG@K}{IDCG@K} \tag{12}$$

Since only the top k results in the recommendation list are needed, the symbol @k is used to represent the metric of the top k results. The calculation formula of $DCG@K$ is as follows:

$$DCG@K = \sum_{i=1}^{K} \frac{2^{rel_i - 1}}{log_2(i+1)} \tag{13}$$

$IDCG@K$ is the ideal discounted cumulative gain, which is the DCG under the optimal ranking. Since $CG@K$ does not take into account the order relationship of each item and items ranked higher have a greater impact on users, it is necessary to introduce $DSG@K$, dividing the score at each position by the discount value, thereby causing items ranked lower to have a greater discount during the calculation process. To make the scores of items ranked higher have a greater influence on the overall score, it is necessary to consider the weight of each item position.
The results of $DCG@K$ are normalized to compare the effects of different recommendation methods. The $IDCG@K$ is introduced as the maximum DCG value under the ideal situation, and its calculation formula is as follows.

$$\text{ID}CG@K = \sum_{i=1}^{K} \frac{2^{rel_i - 1}}{log_2(i+1)} \tag{14}$$

Recall@K is used to evaluate the recall rate of a model, indicating the proportion of predicted positive examples in the test samples among the actual positive examples. '@k' indicates that only the first k results are considered in the metric. The confusion matrix is composed of FN, FP, TN, and TP. Among them, TP(True Positive) represents the positive examples that are correctly predicted, meaning the model judges the current sample as a positive sample and the current sample is actually a positive sample. TN (True Negative) represents the negative examples that are correctly predicted, that is, the model judges the current sample as a negative sample and the current sample is actually a negative sample. FP(False Positive) represents the positive examples that are wrongly predicted, meaning the model judges the current sample as a positive sample but the current sample is actually a negative sample. FN(False Negative) represents the negative examples that are wrongly predicted, that is, the model judges the current sample as a negative sample but the current sample is actually a positive sample. The calculation formula of Recall@k is as follows:

$$\text{Recall}@K = \frac{DCG@K}{IDCG@K} \tag{14}$$

5 Experimental Results and Analysis

5.1 Overall results

The data in the table shows that on the MovieLens1M dataset and the TMDB+Netflix Prize dataset, the model proposed in this paper outperforms all baseline methods in both Recall@20 and NDCG@20 metrics. Specifically, compared with the other five methods, the proposed method in this paper has increased the recall@20 on the MovieLens1M dataset by 29.62%, 21.13%, 14.25%, 19.54%, and 3.21% respectively, and on the TMDB+Netflix Prize dataset by 16.69%, 8.70%, 6.23%, 5.83%, and 4.90% respectively; the proposed method has increased the ndcg@20 on the MovieLens1M dataset by 26.34%, 11.84%, 11.35%, 21.24%, and 6.62% respectively, and on the TMDB+Netflix Prize dataset by 29.93%, 24.22%, 4.65%, 5.65%, and 3.34% respectively.

Table 2. Comprehensive data comparison results

Model	MovieLens1M		TMDB+Netflix Prize	
	Recall@20	NDCG@20	Recall@20	NDCG@20
NFM	0.1614	0.1211	0.2163	0.1784
CKE	0.1727	0.1368	0.2322	0.1866
MKR	0.1831	0.1374	0.2376	0.2215
KGAT	0.1750	0.1262	0.2385	0.2194
MMGCN	0.2027	0.1435	0.2406	0.2243
Our model	0.2092	0.1530	0.2524	0.2318
Improve	3.21%	6.62%	4.90%	3.34%

Model performance evaluation is conducted by calculating metrics such as Recall and NDCG on the validation set and test set to comprehensively measure the performance of the recommendation system. Experimental results demonstrate that LMADG has significant advantages in capturing the dynamic changes of user interests and generating high-quality recommendation lists, and its performance is superior to that of static GNN and traditional methods. The model in this paper outperforms the MKR model and other models because the MKR model does not consider the attention mechanism and is easily disturbed by noise, resulting in performance inferior to that of the model in this paper. These evaluation results provide important references for the further optimization and application of the model.

5.2 Melting experiment

To verify the performance of the model proposed in this paper under different circumstances, this subsection conducts ablation experiments by respectively adopting different modalities, different model depths, and different embedding methods of combination ways to examine their impacts on the model's effectiveness.

To study the impact of different model depths on recommendation performance, this paper compares models with different depths of attention embedding propagation layers on the MovieLens1M and TMDB + NetflixPrize datasets, as shown in the Table 3.

The two evaluation metrics, Recall@20 and NDCG@20, increase as the model depth increases, indicating that integrating multi-hop information into the recommendation system model can enhance the recommendation effect. However, when the model depth exceeds 2 layers, the evaluation metrics Recall@20 and NDCG@20 start to show a downward trend. This might be due to overfitting when the number of layers increases to a certain extent, caused by the relatively sparse multi-hop information in the model; or it could be due to the over-smoothing problem resulting from the deepening of layers in the graph neural network.

Table 3. Experimental results at different depths

Model	MovieLens1M		TMDB+Netflix Prize	
	Recall@20	NDCG@20	Recall@20	NDCG@20
One layer	0.2058	0.1498	0.2431	0.2263
Two layer	0.2092	0.1530	0.2524	0.2318
Three layer	0.2067	0.1504	0.2509	0.2284

To study the impact of different combination methods on the effect of movie recommendation, this paper adopts three different combination methods, namely additive combination method (ADD), concatenation combination method (CONCAT), and bidirectional interaction combination method (BI), to learn their embedding vectors under the MovieLens1M and TMDB+NetflixPrize datasets, as shown in Table 4. The concatenation combination method is superior to both the additive combination method and the bidirectional interaction combination method. The additive combination method is better than the bidirectional interaction combination method. This might be because the bidirectional interaction combination method focuses on the interaction between entities but neglects the features or important information of the entities themselves, resulting in the inability to integrate entity information well and weak information transmission between entities. In contrast, the additive combination method integrates entity information more directly, thus performing better in handling entity information. The concatenation combination method is superior to both the additive combination method and the bidirectional interaction combination method. This might be because the information of general entities and the multimodal information of neighbor node entities are not in the same semantic space. The concatenation combination method can expand the features of different dimensions in the semantic space and is more suitable for fusing information in different semantic spaces.

Table 4. Experimental results with different combination methods

Combination method	MovieLens1M		TMDB+Netflix Prize	
	Recall@20	NDCG@20	Recall@20	NDCG@20
ADD	0.2078	0.1506	0.2511	0.2293
BI	0.2092	0.1494	0.2486	0.2274
CONCAT	0.2092	0.1530	0.2524	0.2318

6 Experimental Results and Analysis

6.1 Research Summary

This paper addresses the issue that traditional context-based movie recommendation models only utilize text data, from which limited information can be obtained due to the single modality, and thus cannot fully address the problem of data sparsity. To this end, a multimodal movie recommendation model (Layered Multi-head Attention Dynamic Graph, LMADG) is proposed, aiming to capture the dynamic changes in user interests and the graph structure information of user-movie interactions. By integrating Graph Convolutional Network (GCN) and temporal attention mechanism, LMADG can effectively extract the temporal features of users and movies and generate personalized recommendation results. Finally, comparative experiments are conducted on the Movielens-1M, TMDB, and Netflix Prize datasets, and the experimental results verify that the proposed model has better recommendation quality.

6.2 Future research directions

Although the effectiveness of the method proposed in this paper has been demonstrated on multiple datasets, in the actual application scenario of movie recommendation, user data and movie information are more complex, and more user-item interaction information needs to be taken into account. Therefore, in the future, more detailed user and movie information can be considered to be introduced into the movie recommendation method to improve the effect of movie recommendation.

References

1. Fan W. Recommender systems in the era of large language models (llms)[J]. IEEE Transactions on Knowledge and Data Engineering, 2024: 1-20.
2. Ge Y, Liu S, Fu Z, et al. A survey on trustworthy recommender systems[J]. ACM Transactions on Recommender Systems, 2024, 3(2): 1-68.
3. Gao C, Zheng Y, Li N, et al. A survey of graph neural networks for recommender systems: Challenges, methods, and directions[J]. ACM Transactions on Recommender Systems, 2023, 1(1): 1-51.
4. Zhao Z, Fan W, Li J, et al. Recommender systems in the era of large language models (llms)[J]. IEEE Transactions on Knowledge and Data Engineering, 2024.)
5. Arnold M, Goldschmitt M, Rigotti T. Dealing with information overload: a comprehensive review[J]. Frontiers in psychology, 2023, 14: 1122200.
6. Gao Y. MOOCs video recommendation using low-rank and sparse matrix factorization with inter-entity relations and intra-entity affinity information[J]. Information Processing & Management, 2024, 61(6): 103861.
7. Wang J, Huang P, Zhao H, et al. Billion-scale commodity embedding for e-commerce recommendation in alibaba[C]//Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 2018: 839-848.

8. Sun Y, Hu W, Zhou D, et al. Alleviating data sparsity problems in estimated time of arrival via auxiliary metric learning[J]. IEEE Transactions on Intelligent Transportation Systems, 2022, 23(12): 23231-23243.
9. Acharya A, Singh B, Onoe N. Llm based generation of item-description for recommendation system[C]//Proceedings of the 17th ACM conference on recommender systems. 2023: 1204-1207.
10. Permana K E. Comparison of User Based and Item Based Collaborative Filtering in Restaurant Recommendation System[J]. Mathematical Modelling of Engineering Problems, 2024, 11(7).
11. Dwivedi P, Islam B. An item-based collaborative filtering approach for movie recommendation system[C]//2023 10th International Conference on Computing for Sustainable Global Development (INDIACom). IEEE, 2023: 153-158.
12. Chaudhari A, Alhussian H, Sarlan A, et al. A Hybrid Recommendation System: A Review[J]. IEEE Access, 2024.
13. Elahi M, Kholgh D K, Kiarostami M S, et al. Hybrid recommendation by incorporating the sentiment of product reviews[J]. Information Sciences, 2023, 625: 738-756.
14. Johnson S J, Murty M R, Navakanth I. A detailed review on word embedding techniques with emphasis on word2vec[J]. Multimedia Tools and Applications, 2024, 83(13): 37979-38007.
15. Zhou C, Li Q, Li C, et al. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt[J]. International Journal of Machine Learning and Cybernetics, 2024: 1-65.
16. Johnson S J, Murty M R, Navakanth I. A detailed review on word embedding techniques with emphasis on word2vec[J]. Multimedia Tools and Applications, 2024, 83(13): 37979-38007.
17. Meisenbacher S, Chevli M, Vladika J, et al. DP-MLM: Differentially private text rewriting using masked language models[J]. arXiv preprint arXiv:2407.00637, 2024.
18. Yu S, Gu C, Huang K, et al. Predicting the next sentence (not word) in large language models: What model-brain alignment tells us about discourse comprehension[J]. Science advances, 2024, 10(21): eadn7744.
19. Wang Y, Yang G, Li S, et al. Arrhythmia classification algorithm based on multi-head self-attention mechanism[J]. Biomedical Signal Processing and Control, 2023, 79: 104206.
20. Ye F, Pu S, Zhong Q, et al. Dynamic gcn: Context-enriched topology learning for skeleton-based action recognition[C]//Proceedings of the 28th ACM international conference on multimedia. 2020: 55-63.
21. Shi H, Miao K, Ren X. Short-term load forecasting based on CNN-BiLSTM with Bayesian optimization and attention mechanism[J]. Concurrency and Computation: Practice and Experience, 2023, 35(17): e6676.
22. Wang Y, Yang G, Li S, et al. Arrhythmia classification algorithm based on multi-head self-attention mechanism[J]. Biomedical Signal Processing and Control, 2023, 79: 104206.