

A Construction Method for Personalized Movie Recommendation System Based on Embedding

Wenwei Zeng

*School of Data Science & Engineering, South China Normal University, Shanwei, China
952133592@qq.com*

Abstract. Personalized recommendation systems are very important. They can improve user experience and help platforms make more money. This study introduces a movie recommendation model based on deep learning and embedding techniques. We train this model using the MovieLens-25M dataset. Different from traditional collaborative filtering methods, our system changes users and movies into a low-dimensional latent space. It uses a multi-layer perceptron to catch complex nonlinear interactions. To solve the problems of data sparsity and cold-start, we use negative sampling and leave-one-out evaluation. The experimental results show that our model works well. The Hit Ratio @10 is 0.85. This proves that embeddings are useful for learning user preferences. Although our model gives good recommendations, it still has limitations. It does not use multimodal data and does not model time changes. In future work, we will use Transformer models, large language models, and multimodal data. These methods can make recommendations more personalized, easier to understand, and more adaptable in dynamic environments.

Keywords: Movie Recommendation System, MovieLens, Deep Learning, Embedding Techniques

1. Introduction

Now, internet technologies are developing very fast. We have entered the big data age. Every day, users see too much different information [1]. It is hard to filter this information quickly and correctly. It is difficult to find content that users really like. This is a big problem for big internet platforms. So recommendation systems become very important. They help solve information overload. They improve user experience and user activity. They also help companies find hidden user needs. This helps companies make more money.

Recommendation systems are used in many areas. They are used in online shopping, video websites, news apps and music apps. People call them the "second search engine" of the internet. Movie recommendation is a very important and competitive area. The film industry is growing fast. There are more and more movies. Users have too many choices and feel confused. A good movie recommendation system can find what users like. It can make users happier and help platforms make more money. It is very useful and valuable to build this kind of system. But now users want more personalized services. Traditional recommendation methods cannot meet these needs. Traditional collaborative filtering (CF) was useful before, but it has many problems. These methods need user scores and interaction data. When there is not enough data, the user-item matrix is very sparse. Then the recommendation results become less correct and less complete. Also, CF cannot learn deep user

interests. It cannot understand the meaning of content well. Its "personalization" is not real. It cannot give truly personalized recommendations. Most importantly, CF cannot solve the cold-start problem. When there are new users or new movies, there is no past data. Then the recommendation results become much worse.

In recent years, deep learning has developed quickly. It brings new chances for recommendation systems. Embedding is an important part of deep learning. It has strong ability to show information. It helps us build better recommendation models. Embedding changes users and movies into low-dimensional vector space. Users who like similar things and movies with similar features will stay close in this space. These vectors can catch complex non-linear relationships between users and movies. Neural networks can learn hidden features by themselves. This makes the model work better and give better recommendations. Besides, embedding can use unstructured data like text, pictures and social information. We can build multimodal recommendation systems. This makes recommendations more personalized.

Based on these technologies, we design and build a personalized movie recommendation system using embedding techniques. This system works very well. It understands user preferences better. It gives more correct and more different recommendations. Experiments show that this system is useful and practical. It provides a good solution for movie recommendation.

2. Previous works

With the fast development of technology and changing user needs, recommendation systems have gone through many stages. From early rule-based methods to today's intelligent models using deep learning, the main goal is still the same: to give users highly personalized and correct content recommendations.

Among traditional recommendation methods, collaborative filtering (CF) is one of the most widely used [2]. This method mainly uses past user-item interaction data. User-based CF finds users who are similar to a target user and then guesses the target user's preferences. It is simple, easy to understand, and does not need item content. However, when user behavior data is not enough, similarity between users becomes unreliable, and recommendation accuracy becomes lower. Also, when more users join, calculating similarity between every pair of users needs a lot of time and computer space. In contrast, item-based CF finds similarity between items. It recommends movies similar to those the user has watched or liked before. This way is usually more stable because item features do not change as much as user behavior, so calculations can be reused. But when there are too many items, calculating the item similarity matrix still needs a lot of computing. In addition, item-based CF still has cold-start problems, especially for new movies. To solve these problems, hybrid recommendation systems have appeared. These systems combine user-based CF and item-based CF, or mix collaborative filtering with content-based methods. They try to balance accuracy, diversity, and coverage. For example, some hybrid systems use collaborative filtering for users with much behavior data, and use content features when user data is few, so cold-start problems become less serious. Although hybrid models usually improve recommendation performance, they have complex structures, need hard work to adjust parameters, and depend a lot on feature engineering and parameter settings. Besides CF, content-based recommendation is another important kind of traditional methods [3]. This method matches static movie features such as genre, director, actors, and story descriptions with items the user liked before to make recommendations. Because it does not need other users' data, it partly solves data sparsity and cold-start problems. Also, it is easier to explain, since we can clearly find why a recommendation is given. However, content-based methods strongly depend on manually extracted features and cannot easily find hidden user interests. So they often create the "information cocoon" effect, which limits recommendation diversity and exploration. Some improved similarity-based methods have also been made in traditional

frameworks. Methods using cosine similarity, Pearson correlation, or Jaccard index try to better measure user or item relationships [4]. Although these methods are based on neighborhood-based CF, their different similarity rules aim to improve accuracy and personalization. Even though these traditional methods were important in early recommendation systems, their weaknesses have become more and more clear. As user needs become more complex and large, these methods are no longer enough to model user interests well.

With great progress in deep learning, neural network models have been slowly used in recommendation systems to automatically find high-order features and hidden relations. Convolutional neural networks (CNNs) can use visual information such as movie posters and pictures well to improve recommendation quality [5]. Recurrent neural networks (RNNs) are good at showing time order and changes in user behavior sequences, so they can catch dynamic user interests more correctly. Compared with traditional methods, these deep models can deal with more complex nonlinear patterns and make much better recommendations. However, they also need large amounts of data and high computing resources, and they are usually hard to explain.

In this trend, embedding techniques—a key part of deep learning—have started to play an important role in recommendation systems. By turning users and movies into dense, low-dimensional vector spaces, embeddings can efficiently catch hidden features and nonlinear relationships between users and items. Compared with traditional collaborative filtering, embedding-based methods work better when data is sparse and in cold-start situations. Also, they can mix multimodal information such as text and images, so they can give more correct and personalized recommendations.

3. Dataset and preprocessing

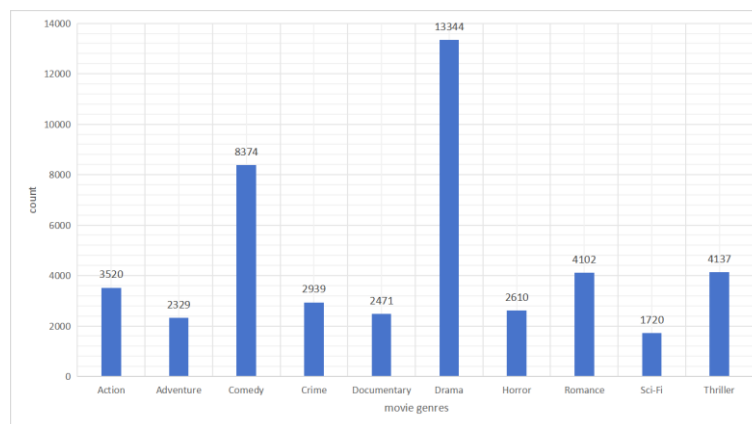


Figure 1. Distribution of movie genres in the dataset

This study uses the MovieLens-25M dataset. GroupLens Research released this dataset to the public [6]. The dataset has 25,000,095 five-star rating records. These records show how 162,541 users interacted with 62,423 movies. Each record has a user ID, a movie ID, a rating score (from 0.5 to 5.0, increasing by 0.5 each time), and a timestamp. Besides, the dataset gives movie titles, genre labels, and external IDs connected to IMDB and TMDB. These can help make the metadata richer. Figure 1 shows the distribution of the 10 most common genres in the MovieLens-25M movie list. Figure 2 shows how users rated the movies in the dataset.

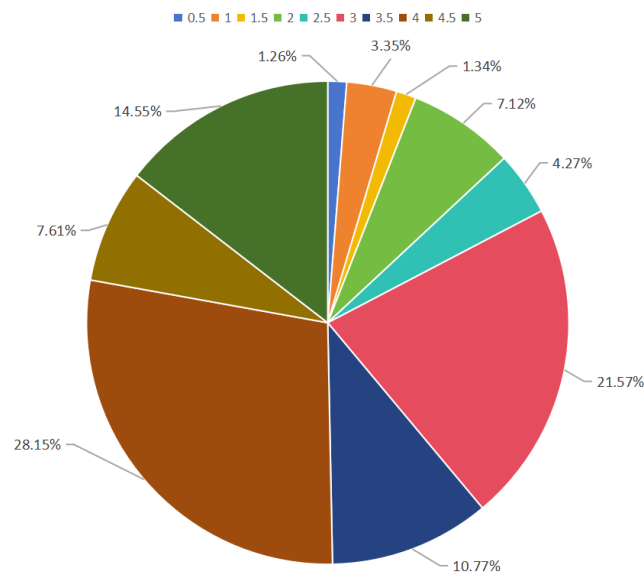


Figure 2. Distribution of rating scores in the dataset

When we preprocessed the data, we chose some users randomly from the original rating data. This was to keep the data size easy to handle and to remove useless redundant information. To make the training set and test set, we sorted each user's rating history by timestamp (in the order of time). We used the leave-one-out strategy: the latest rating record of each user was put into the test set, and the other interactions were put into the training set. This setting is like a real recommendation situation, where we predict what users will do in the future. In the training set, all user-movie interactions were seen as positive feedback (meaning users liked those movies). Recommendation systems often have too few positive feedbacks, so we added negative samples to better find out what users really like. Specifically, for each positive user-movie pair in the training set, we randomly chose many negative samples from the movies that the user had not interacted with. These negative samples and positive samples were mixed in a fixed ratio (for example, 4 negative samples to 1 positive sample). This made a balanced training dataset with both positive and negative samples.

4. Model and results

In this study, we made a personalized movie recommendation model using deep learning. The model uses embedding techniques to find hidden connections between users and movies. It uses neural networks to model nonlinear relationships well. Unlike traditional collaborative filtering methods, this model does not directly use the sparse user-movie rating matrix. Instead, it changes users and movies into low-dimensional dense vectors called embeddings. It calculates their similarity in a continuous vector space [7]. This design reduces data sparsity. It also helps the model learn deep user interests more naturally in the embedding space.

Table 1. Architecture of the proposed recommendation model

	Name	Type	Params
0	User input	Embedding	1 M
1	Movie input	Embedding	1 M
2	Neural Network	Linear	1 K
3	Neural Network	Linear	2 K
4	Output	Linear	33

As shown in Table 1, the whole model has two embedding layers and three fully connected layers. User IDs and movie IDs are first turned into 8-dimensional dense vectors by their own embedding layers. This changes discrete user and movie IDs into a continuous latent feature space. It gives efficient input for later feature learning. Then the user and movie embeddings are combined and sent into a multi-layer perceptron (MLP). Two nonlinear fully connected layers with 1K and 2K parameters learn higher-order feature interactions step by step. Nonlinear activation functions like ReLU help the model learn complex relationships between user interests and movie features. The last layer is a linear output layer. It gives a score that shows how much a user may like a movie.

During training, real user–movie interactions are used as positive samples. Negative samples are movies that the user has not watched. We make a training set with 1 positive sample and 4 negative samples. The model uses cross-entropy loss for training. We update parameters in embedding layers and fully connected layers using gradient descent. This helps the model tell positive and negative samples apart better. To test the model, we use Hit Ratio @10 as the main measure. It checks whether the top 10 recommended movies include the one the user really watched. Experiments show that our model gets a Hit Ratio @10 of 0.85 on the MovieLens dataset. This means the model can find real user preferences in most top-10 recommendations. It shows strong recommendation quality and real practical value.

5. Conclusion

This study looks at personalized movie recommendation. We propose a deep learning recommendation model based on embedding techniques using the MovieLens dataset. The model changes users and movies into low-dimensional dense vectors. It uses a multi-layer perceptron (MLP) to find hidden interaction patterns. This helps the model learn user preferences well. Experimental results show that the model gets a Hit Ratio @10 of 0.85. This proves that embedding techniques work well for movie recommendation.

Although the model works well, this study has several limitations. First, we only test the model on the MovieLens dataset. MovieLens is a common benchmark in recommendation research. But it is smaller and less diverse than real commercial platforms that have millions or hundreds of millions of user interactions. Because of this small scale, we cannot fully test how strong and general the model is in large and complex environments. We also cannot test it well on long-tail or cold-start users. Second, the model mainly uses user past rating data to build features. It does not use multimodal information such as movie descriptions, user reviews, or movie posters. These unstructured data have rich meaning and feelings. They are important to understand user interests and movie features more fully. Without these extra data, the model cannot find deep user preferences and detailed content features. This may make recommendations less diverse and harder to explain. Besides, the current model has a simple structure. It uses embeddings and an MLP to model feature interactions. Although this can catch some nonlinear relationships, it cannot model complex behavior sequences and context relationships well. The model does not think about how user interests change over time. It also does not use advanced structures such as attention mechanisms or graph neural networks. This may limit its performance in highly personalized recommendation tasks.

Future work can be improved in several ways. First, using the Transformer structure is a good direction. Transformers work well in sequential modeling. They use multi-head self-attention to catch global relationships and detailed patterns in behavior sequences. Compared with traditional RNN or LSTM, Transformers can compute faster and represent sequences better. They help understand how user interests change over time more accurately [8]. Adding Transformers to recommendation systems should improve context modeling of user history and give more personalized recommendations. Second, combining large language models (LLMs) with multimodal information can greatly improve the model [9]. Movie data include not only structured ratings but also rich text

descriptions, reviews, and visual data such as posters and pictures. LLMs can deeply understand the meaning in reviews and story summaries. They help model user interests and movie features accurately. At the same time, multimodal learning that mixes text, image, and structured data can fix the problems of single-modality models. It can show user preferences and movie features more completely. It also improves recommendation diversity and explainability. Third, exploring contrastive learning [10] and transfer learning [11] can help make the model more general. Contrastive learning uses similar and different sample pairs to learn strong representations. It is especially useful when data are sparse or labels are weak. Transfer learning uses knowledge from other areas or tasks for movie recommendation. It reduces the need for large labeled datasets. It also improves performance in cold-start and cross-domain situations. Using these two methods together lets recommendation models keep high accuracy and stability when data are limited or environments change. Finally, studying better negative sampling methods and dynamic user modeling is also important. Current negative sampling uses random selection. It may not show real user preference boundaries well. Future methods can use hard negative sampling or adversarial generation to make more useful and difficult negative samples. This improves the model's ability to tell differences. At the same time, dynamic user modeling can better catch how interests change over time. It gives more personalized and timely recommendations.

In summary, this study shows that embedding techniques work well in personalized movie recommendation. It provides a practical framework to build efficient recommendation systems. With the continuous development of deep learning and multimodal modeling, future recommendation systems will understand users better and become more intelligent. They will give more accurate and richer recommendation experiences.

References

- [1] A. Da'u and N. Salim, "Recommendation system based on deep learning methods: A systematic review and new directions," *Artif. Intell. Rev.*, vol. 53, no. 4, pp. 2709–2748, Apr. 2020, doi: 10.1007/s10462-019-09744-1.
- [2] 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, June 2020, doi: 10.3390/info11060317.
- [3] S. A. Thorat, G. Ashwini, and M. Seema, "Survey on collaborative and content-based recommendation systems," in *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Jan. 2023, pp. 1541–1548. doi: 10.1109/ICSSIT55814.2023.10061072.
- [4] S. C. Mana and T. Sasipraba, "Research on cosine similarity and pearson correlation based recommendation models," *J. Phys. Conf. Ser.*, vol. 1770, no. 1, p. 012014, Mar. 2021, doi: 10.1088/1742-6596/1770/1/012014.
- [5] H. Daneshvar and R. Ravanmehr, "A social hybrid recommendation system using LSTM and CNN," *Concurr. Comput. Pract. Exp.*, vol. 34, no. 18, p. e7015, 2022, doi: 10.1002/cpe.7015.
- [6] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans Interact Intell Syst*, vol. 5, no. 4, p. 19:1–19:19, Dec. 2015, doi: 10.1145/2827872.
- [7] S. Li et al., "Embedding compression in recommender systems: A survey," *ACM Comput Surv*, vol. 56, no. 5, p. 130:1–130:21, Jan. 2024, doi: 10.1145/3637841.
- [8] H. I. Pohan, H. L. H. S. Warnars, B. Soewito, and F. L. Gaol, "Recommender system using transformer model: A systematic literature review," in *2022 1st International Conference on Information System & Information Technology (ICISIT)*, July 2022, pp. 376–381. doi: 10.1109/ICISIT54091.2022.9873070.
- [9] Z. Zhao et al., "Recommender systems in the era of large language models (LLMs)," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 6889–6907, Nov. 2024, doi: 10.1109/TKDE.2024.3392335.
- [10] Z. Liu, Y. Ma, Y. Ouyang, and Z. Xiong, "Contrastive learning for recommender system," Jan. 05, 2021, arXiv: arXiv:2101.01317. doi: 10.48550/arXiv.2101.01317.
- [11] J. Fu et al., "Exploring adapter-based transfer learning for recommender systems: Empirical studies and practical insights," in *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, in *WSDM '24*. New York, NY, USA: Association for Computing Machinery, Mar. 2024, pp. 208–217. doi: 10.1145/3616855.3635805.