Movie Recommendation System Based on Traditional Recommendation Algorithm and CNN Model

Haitao He [1,*,†], Zhifu Shang [2,†], Mingjie Wu [3,†], Yuling Zhang [4,†]

[1]Department of Electrical Engineering and Computer Science, NanFang College of Sun Yat-Sen University, Guangzhou, China

[2]College of Chemistry, Chemical Engineering and Resource Utilization, Northeast Forestry University, Harbin, China

[3]School of Information Science and Engineering, Ocean University of China, Qingdao, China

[4]School of Information Technology, Beijing Normal University Zhuhai, Zhuhai, Guangdong, China

* Corresponding Author Email: heht177128@stu.nfu.edu.cn

[†]These authors contributed equally.

Abstract. As streaming services have expanded in recent years, an excellent recommendation algorithm, as one of the core technologies in movie service, brings huge benefits. Although the current application research of recommendation algorithms is mature, most systems or products usually rely on only one main algorithm. This makes it difficult for the system to overcome the shortcomings of various algorithms and cannot benefit from the combination of multiple recommendation algorithms. In addition, the widely used recommendation models are found to be unable to extract the finer features of users and movies, and the calculation time is very long. Meanwhile, the recommendation results are inaccurate, and they are not user-friendly. In this research, we design and construct a system to recommend movies which is consisted by a model based on a convolutional neural network consisting. By extracting the features of users and movies, we can calculate the direct similarity of different users' movies and then predict movie ratings to give recommendations. When extracting features of users and movies, we refer to traditional algorithms based on content and content. Through the "cosine similarity" and "user movie score matrix", the Item-Based Collaborative Filtering and User-Based Collaborative Filtering can be well implemented. To sum up, our movie recommendation system is based on Convolutional Neural Network (CNN) model and the performance of the system is improved by adopting multiple recommendation algorithms such as content-based recommendation and system filtering. We efficiently trained the neural network and finally built a movie recommendation system with a faster operation speed, although some parts are not perfect.

Keywords: CNN, Movie Recommendation System, Deep Learning.

1. Introduction

Since the COVID-19 pandemic, as movie theaters were forced to suspend operations, major movie markets of the world have seen a severe drop in the box office which have ranged from 45.6% to 82.6%. Compared with 2019, the global box office approximately fell by 70%. On the contrary, streaming movies boomed whose box office had repeatedly hit new highs in 2020, due to both the pandemic and improved technology. A movie recommendation system is a core technology of streaming media services, which utilizes the viewing data of platform users to provide personalized movie recommendations to each user. An excellent recommendation algorithm not only improves user experience but also brings huge benefits to streaming companies. For instance, Netflix in the United States, the recommendation algorithm they designed saves them $1 billion a year [1]. It can be seen that training an effective movie recommendation model through user data to improve user experience are essential to the development of the streaming media industry.

Currently, the most widely used recommendation algorithms on the market include content-based recommendation (CB) and collaborative filtering (CF). Prior to this, the research on the application of recommendation algorithms has been very mature, both in terms of accuracy and speed. However,

the majority of systems or products, on the other hand, often only use one recommendation algorithm as the main body, and the rest of the algorithms generally serve the main algorithm. It will lead to a disadvantage that the system cannot benefit from the combination of multiple recommendation algorithms. In fact, the combined model may overcome the shortcomings of various algorithms and further improve the performance of the system. For example, CB recommendation algorithms can solve the "new item" problem of collaborative filtering, and CF can reduce the degree of "overfitting" of CB algorithms [2]. CB recommendation is particularly dependent on historical data and cannot mine users' potential interests, so it will cause overfitting. The openness and ability to learn from others' experiences in the CF results can help users discover new interests and pastimes. An article only uses a CB algorithm for recommendation, which has obvious shortcomings, and it fails to help users discover other content that may be of interest. Similarly, since the core of CF is based on historical data, new users and new items have a "cold start" problem, while there isn't a "cold start" problem in CB recommendation for items, because the tags and content of items will not be received the impact of user data [3]. In addition, the implementation of most systems is based on simple machine learning methods, and some deep neural networks in deep learning are not used, while this research uses a convolutional neural network (CNN) model to help the system further improve performance.

To solve the limitation mentioned above, this research discusses the advantages and disadvantages of recommendation algorithms based on CB and CF algorithms, and gives optimization suggestions simultaneously. Furthermore, we synthesized the ideas of the above traditional recommendation algorithms constructed a recommendation model of movies which is based on the CNN deep learning algorithm, and applied it to the simple movie system we designed. The model adopts the calculation method of cosine similarity, tests it through the Movielens/ml-1m data set to calculate its loss and accuracy, and provides the results in the end. In addition, a scheme that can alleviate the cold-start problem of users, items and systems is also presented [4]. Despite the efficient training of our model, the accuracy of forecasting user ratings is not high, which may be due to the loss function's irrational design [5].

The rest parts of the chapter are sequenced as follows: Section 2 provides our method including specific structure and the experiment details. The experiment result data and discussion for these results will be given in Section 3. At the last, Section 4 discusses the conclusions of this work and future plans.

2. Method

2.1. Dataset description and preprocessing

In this project, we used the MovieLens 1M Dataset that is provided for free by grouplens [6], a research lab at the University of Minnesota. This dataset recorded 6040 users' ratings of 3883 movies, with a total of 1 million rating records, rating from 1.0 to 5.0. It includes three files, namely: users.dat, ratings.dat, and movies.dat. Table 1, Table 2 and Table 3 provide the sample data of 'movies.dat' and 'ratings.dat' and 'users.dat', respectively.

Table 1. Part of the movies.dat (MovieID::Title::Genres, 3883 records in total)

MovieID	Title	Genres
1	Toy Story (1995)	Animation|Children's|Comedy
782	Hunchback of Notre Dame, The (1996)	Animation|Children's|Musical
1565	Head Above Water (1996)	Comedy|Thriller
2348	Sid and Nancy (1986)	Drama
3131	Broadway Damage (1997)	Comedy

The title is composed of the film's name and release year. Different types of films are separated by pipe symbols, including animation, documentary, comedy, crime, etc, 18 genres in total [7].

Table 2. Part of the ratings.dat (UserID::MovieID::Ratings::Timestamp, 100209 records in total)

UserID	MovieID	Ratings	Timestamp
1	1193	5	978300760
1208	3948	4	974842609
2416	3864	3	974245048
4832	3082	3	962898636
6040	1094	5	956704887

For Table 2, user IDs range between 1 and 6040, 6040 users in total; Although the maximum of movie IDs is 3952, there are just 3883 unique movie IDs in the dataset; Ratings are scored from 1.0 to 5.0, with a maximum of 5 and a minimum of 1; Timestamp is Unix seconds since 1970/01/01 UTC [8]; 20 ratings are required for each user at least.

Table 3. Part of the users.dat (UserID::Gender::Age::Occupation::Zip-code, 6040 records in total)

UserID	Gender	Age	Occupation	Zip-code
1	F	1	10	48067
1208	M	25	0	12590
2416	M	45	7	01531
4832	M	25	14	17857
6040	M	25	6	11106

As to Table 3, use ‘M’ and ‘F’ for male and female, respectively; The number in the age field represents an age range, by way of example, 1 means younger than 18, and 18 means between 18 and 24 and so forth, 7 ranges in total; Apart from that, different occupations correspond to different figures, for instance, 0 indicates other or not specified, 1 indicates academic/educator, 2 indicates artist, etc, 21 choices in total[9, 10]; Zip-code was not used in this project.

The preprocessing is consisted of two parts. Firstly, the gender of a string type in the user data needs to be converted to a numeric type so that it can be passed into the model for training. To be more specific, the string type ‘F’ and ‘M’ are replaced with the number 1 and 0 respectively.

Secondly, the preprocessing for movie data is also considered. The first step is to store the movie ID information in the dictionary and get the maximum value of the movie ID. Then count the words in the movie name and assign a number to each word. After that count the words in the movie category and assign a number to each word. Finally, the movie category and the movie name are filled with a fixed length, and all the movie data is saved to the dictionary. Table 4 presents an example based on the comparison between pre-preprocessing and post-preprocessing data.

Table 4. Comparison before and after data processing

Data source	Before	After
User data	UserID::Gender::Age::Occupation 1::F::1::10	{‘usr_id’: 1, ‘gender’: 1, ‘age’: 1, ‘job’: 10}
Movie data	MovieID::Title::Genres 2::Jumanji (1995)::Adventure|Children’s|Fantasy	{‘mov_id’: 2, ‘title’: [3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], ‘category’: [4, 2, 5, 0, 0, 0]}
Rating data	UserID::MovieID::Rating 1::1193::5	{‘usr_id’: 1, ‘mov_id’: 1193, ‘score’: 5}

2.2. Proposed approach

In this project, we referred to traditional algorithms which are Content-Based Recommendation and Collaborative Filtering, and the algorithm is based on the CNN model. Taking into account the differences in ratings between different users, the project uses “cosine similarity” to calculate the

similarity of each content. The main drawback of cosine similarity is that it does not consider the magnitude of the vector, only its direction, whereas "zero-mean" is able to alleviate this issue. Through the "cosine similarity" and "user movie score matrix", the User-Based Collaborative Filtering and Item-Based Collaborative Filtering can be well implemented.

$$Pred(u,i) = \widehat{r_{ui}} = \frac{\sum_N Sim(i,j) * r_{uj}}{\sum_N |Sim(i,j)|} \quad (1)$$

For formula (1), it is to predict the user's score on the movie "i", and the "N" indicates a commodity similar to the film "i".

$$Pred(u,i) = \widehat{r_{ui}} = \frac{\sum_v Sim(u,v) * r_{vi}}{\sum_v |Sim(u,v)|} \quad (2)$$

For formula (2), it is to predict the user's rating of the movie "i", and "v" indicates a user similar to the user "u". In addition, through the "movie similarity matrix", Content-Based Recommendations can be easily realized. Figure 1 illustrates the overall process of Content-based recommendation.

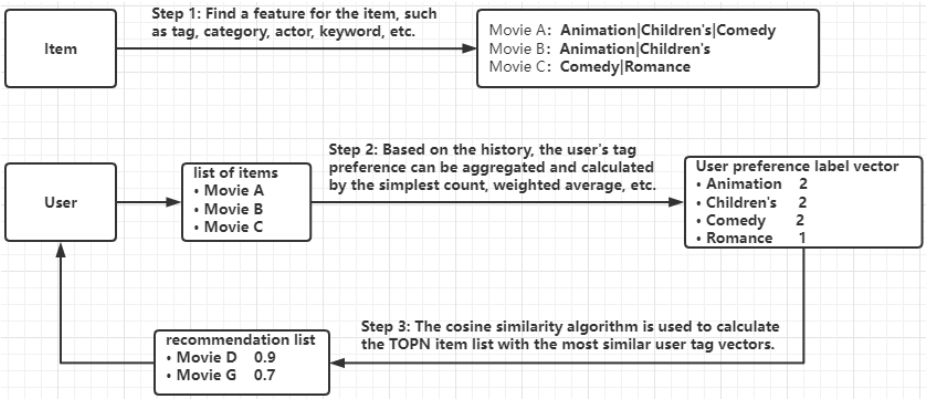

Figure 1. Steps for content-based recommendation

As to the CNN model, we design a simple movie recommendation neural network model, the design includes 4 procedures. Firstly, the feature data of user and movie are respectively converted into feature vectors. Secondly, for these feature vectors, a fully connected layer or convolutional layer is used to further extract features. Thirdly, the feature vectors of multiple user and movie data are fused into a vector representation, which is convenient for similarity calculation. Finally, calculate the similarity between features. Figure 2 provides our ideas of network designing.

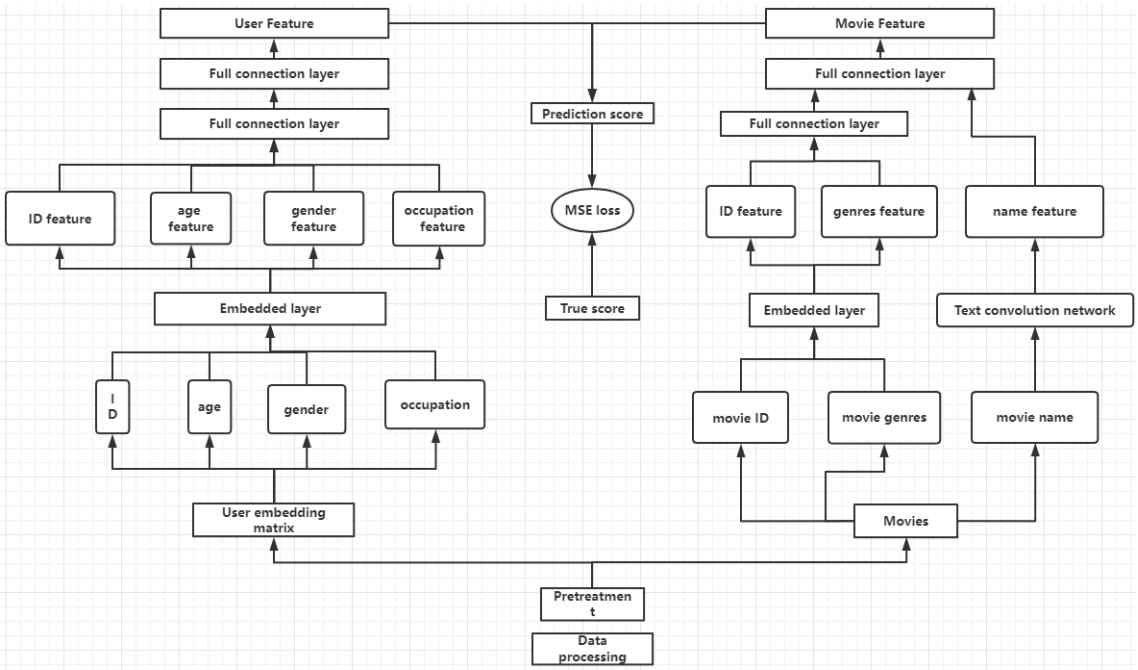

Figure 2. Design of network structure

2.3. Function of movie recommendation and overall process in our application

The function of movie recommendation is when a user enters the system, the system will query popular movies and recently released movies according to the rules and return them to the user. Meanwhile, the system will then predict user preferences and return recommended movies. Figure 3 shows the process of recommendation after a user enters the website.

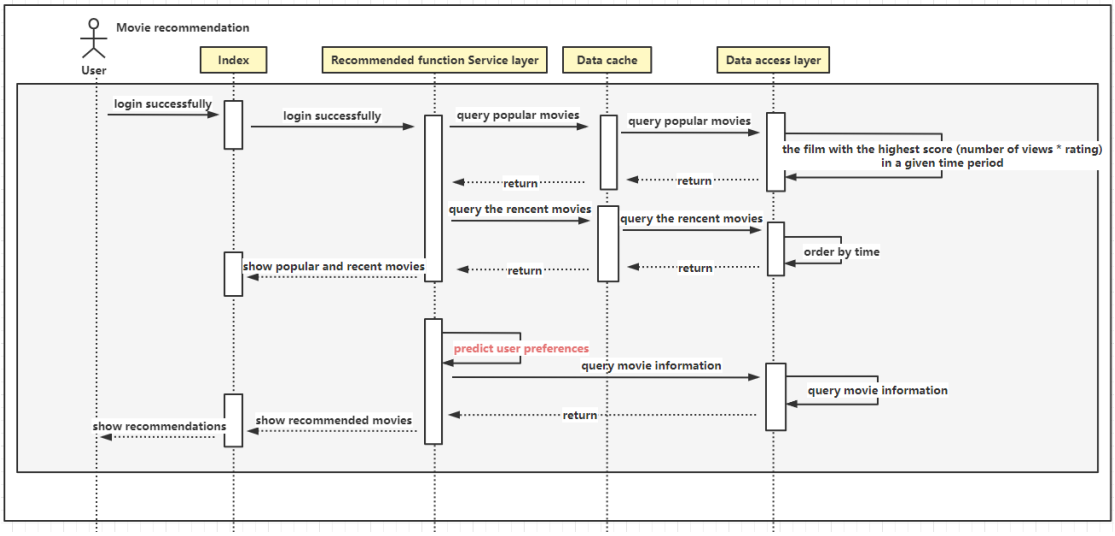

Figure 3. Recommended sequence diagram

What is more, the overall process from entering the website to exiting the website can be summarized as follows: 1. Check whether the user is logged in. If not, the user needs to register an account; 2. Recommend movies to users according to system rules. Figure 4 overall explains the whole operation process of the system.

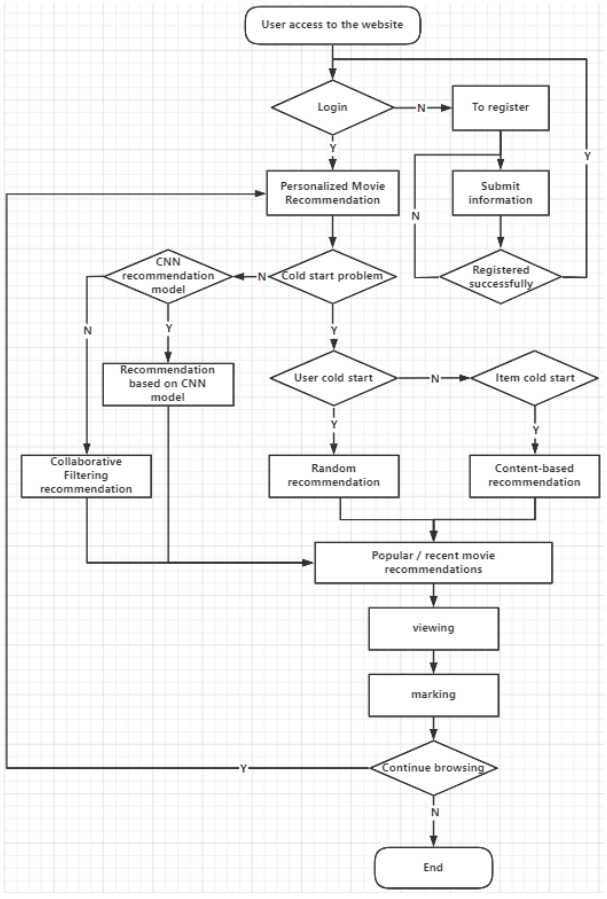

Figure 4. System flow chart

3. Results and discussion

3.1. Display of experimental results

In this CNN model, through continuously adjusting the hyper-parameters, ultimately, the hyper-parameters that are epoch, batch size and learning rate are set to 5, 256 and 0.00005 respectively. Table 5 illustrates the accuracy and loss value between the predicted score and the real score in each round of training results. Figure 5 shows the trend of loss value using above-mentioned hyper-parameters in each epoch. Figure 6 shows the information of five movies recommended by the system to user whose ID is 1 through this model.

Table 5. Accuracy rate and loss value in each round of training results

Epoch	ACC	MSE
1	0.207700	0.97014
2	0.210722	0.95414
3	0.208711	0.95074
4	0.207461	0.948718
5	0.206411	0.94746

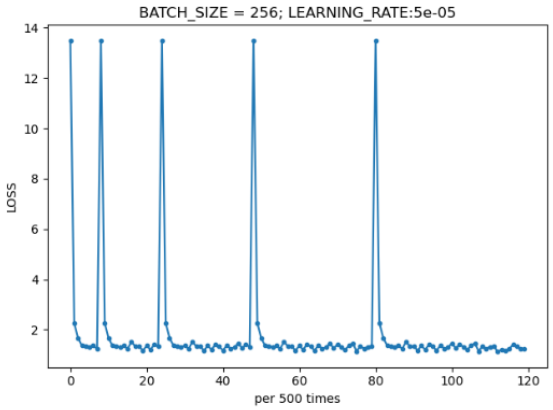

Figure 5. Loss value per round of training

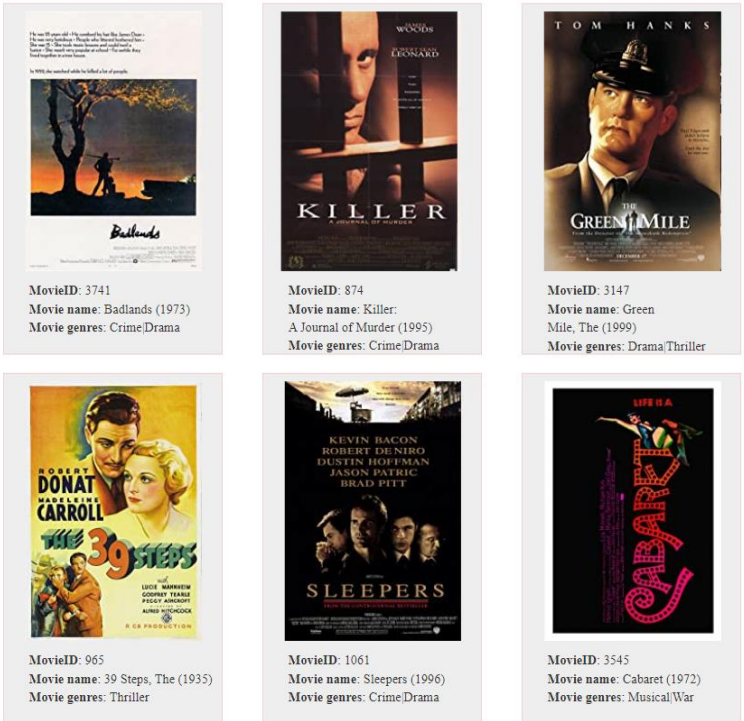

Figure 6. Movies recommended for user whose ID is 1

3.2. Discussion of experimental results

ACC represents the accuracy rate, MAE represents the average absolute value error, the higher the ACC, the better, and the lower the MAE, the better. Obviously, the effect of the model we finally realized is not ideal, which indicates that the system is not accurate in scoring prediction. Whereas, the training results indicate that the prediction score is not accurate, which does not mean whether the film should not be recommended.

The convergence of the system loss value indicates that the training of the network is effective, however the accuracy of predicting user scores is not high. The possible reasons why the effect not good are that first the structure of the neural network is not reasonable, for example, the design of the embedding layer leads to sparse matrix; Secondly, in this project, we utilize the MSE to calculate the difference between the real value and the predicted value. It may also be caused by unreasonable design of loss function; What is more, probably because of the optimizer.

4. Conclusions

In this work, a movie recommendation system is completed. We use a deep learning CNN model that ameliorate the property of the system by employing various recommendation algorithms such as content-based recommendation and system filtering. Compared with other systems, our system is excellent in terms of computing speed, comprehensive consideration of users' potential interests, and effective combination of algorithms, but it is slightly inferior in recommendation accuracy and MSE. We analyze that there may be a problem with the optimizer, or there may be a problem with the data set. Compared to datasets from other systems, they are more standardized and larger than the datasets we use. However, the low accuracy of the predicted score does not mean that it should not be recommended, because the actual score of the user will not be the same as our predicted score, but a higher or lower probability. In the future, we will shift the focus from a specific score value to a manageable range, focusing more on whether users are interested, rather than whether the predicted score is more accurate.

References

[1] Siles I, Espinoza-Rojas J, Naranjo A, et al. The mutual domestication of users and algorithmic recommendations on Netflix [J]. Communication, Culture & Critique, 2019, 12(4): 499-518.

[2] Elahi M, Ricci F, Rubens N. A survey of active learning in collaborative filtering recommender systems [J]. Computer Science Review, 2016, 20: 29-50.

[3] Salakhutdinov R, Mnih A, Hinton G. Restricted Boltzmann machines for collaborative filtering [C]//Proceedings of the 24th international conference on Machine learning. 2007: 791-798.

[4] Lika B, Kolomvatsos K, Hadjiefthymiades S. Facing the cold start problem in recommender systems [J]. Expert systems with applications, 2014, 41(4): 2065-2073.

[5] Pereira A L V, Hruschka E R. Simultaneous co-clustering and learning to address the cold start problem in recommender systems [J]. Knowledge-Based Systems, 2015, 82: 11-19.

[6] Grouplens. Ml-1m [R]. https://files.grouplens.org/datasets/movielens/ml-1m.zip

[7] Beukman E. Improving collaborative filtering with fuzzy clustering [D]. Stellenbosch: Stellenbosch University, 2021.

[8] Lokesh A. A Comparative Study of Recommendation Systems [J]. 2019: 28-32.

[9] Zaccone G, Karim M R, Menshawy A. Deep learning with TensorFlow [M]. Packt Publishing Ltd, 2017: 388-392.

[10] Grouplens. Ml-100k [R]. https://files.grouplens.org/datasets/movielens/ml-100k-README.txt