2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI) | 978-1-7281-9638-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/MLBDBI51377.2020.00102

A Personalized Movie Recommendation System based on LSTM-CNN

Haili Wang*
Leeds College
Southwest Jiaotong University
Chengdu, 611756, China
* Corresponding author: 1061686594@qq.com

Nana Lou
Engineering Management Department
Peking University
Beijing, 100871, China

Zhenlin Chao
Computer Science Department
East China Normal University
Shanghai, 200241, China

***Abstract*—In the context of such an era where nearly everything is based on big data, personalized recommendation systems are becoming increasingly valuable for research. Deep learning has attained great achievements in numerous fields by virtue of its powerful computing power and extraordinary nonlinear transformation capabilities. Applying deep learning to a recommendation system that needs to mine and extract features from massive amounts of data will not only help the development of recommendation algorithms, but also improve the algorithm performance and thus improve the user experience. This project introduces a recommendation algorithm based on LSTM-CNN and applies it to the recommendation of movies by mining user behavior data and recommending movies with higher ratings to them. This article uses the data provided by the movie website MovieLens. It is testing set and training set that the data is divided into, and Top-N recommendation list is produced for the training set, while the algorithm is evaluated on the testing set. It is the features of the data that LSTM-CNN can effectively extract and complete the recommendation from the results.**

Keywords-LSTM, CNN, movie, recommendation system, personalization

I. INTRODUCTION

As information technology develops, at present information overload has already replaced information shortage. In current era, faced with exponentially increasing information, what really confuses people is no longer how to obtain information, but how to quickly extract useful information from massive information. With this dilemma, recommendation system came into being and became an important tool for connecting users and data. A successful recommendation system not only accurately predicts user behavior based on data, but also discovers users 'potential interests, helping them find things that are not easy to find but meet their preferences. In fact, in daily life, recommendation systems are already everywhere.

For example, our commonly used shopping platforms: Amazon, Tmall; social networks: Facebook, Twitter, WeChat. It is precisely because of the existence of the recommendation system that we can quickly find the items we need among countless products, and we can also surprise in the recommendation list to find the products that we are interested but unexpected. It is also because of the existence of the recommendation system that we can successfully find our favorite bloggers and browse the content we are interested in from time to time. There is no doubt that the recommendation system is becoming a very significant part of our daily lives and brings us great convenience.

In recent years, with its powerful data mining capabilities, deep learning has been well received in many fields[[1],[2]]. How to apply deep learning to recommend products that are of real interest to specific people more efficiently has also received attention. RecSys, an international conference, has also actively held seminars to encourage the research on this direction. Hence, it is undeniable that deep learning will be the new trend and future direction of recommendation systems.

II. RELATED WORK

A. Traditional Methods

There are three categories of recommendation algorithms: Collaborative filtering recommendation algorithm, content-based recommendation algorithm and hybrid recommendation algorithm.

Cold start problem can be solved by content-based recommendation algorithm well since it does not need to provide historical behavior data of users. This algorithm mainly recommends the content to users according to the feature data of user and item provided in advance. For instance, given the data of users who like Disney series movies, the system will recommend movies such as "Snow White" and "Frozen". The key of the algorithm is to model based on the characteristic data of users and items, so its performance depends on the completeness and comprehensiveness of the model.

Collaborative filtering algorithm, which is one of the most common recommendation algorithms at present, is mainly based on user behavior [[3]]. And it can achieve prediction depending on historical behavior data of existing users without pre-

978-1-7281-9638-1/20/$31.00 ©2020 IEEE
DOI 10.1109/MLBDBI51377.2020.00102

acquisition of the user or item feature data. User behavior data is divided into display feedback (mainly for user ratings) and implicit feedback (mainly includes purchase, browsing, and searching). It is user's true preferences that the explicit feedback can reflect accurately and directly, while it needs analysis and processing to reflect the user's preference for implicit feedback, but there is a lot of noise. There exist two kinds of collaborative filtering algorithms: user-based collaborative recommendation and item-based collaborative recommendation[[4],[5]]. User-based collaborative recommendation: first seek the users who have similar interests with the target user, and then recommend the items attract the users found to the target user. Item-based collaborative recommendation: based on user behavior data, recommend items that are similar to items they previously liked. Most users who like Item A also like Item B, which means that Item A is similar to Item B. The common application of Item-based collaborative recommendation is "The user who bought the item also bought ...".

It is no denying that each recommendation algorithm mentioned above has its own advantages and disadvantages and has different effects in different applications. Content-based recommendation relies on feature data of users and items, but it can avoid cold start problems for new users; collaborative filtering recommendation does not require rigorous modeling or understand the description of items, and can share the experience of others to help users discover potential Preference, but this algorithm relies too much on historical data, so that the cold start problem is inevitable and the performance is affected by the amount and accuracy of user historical data directly. Therefore, in practical applications, there usually exist a variety of recommendation algorithms in one recommendation system to achieve a better recommendation effect[6]. However, when processing text, images and other complex information, these algorithms often cannot achieve very satisfying results.

B. Application of Deep Learning in Recommendation System

Hinton and his students first proposed the recommendation system based on deep learning. Later, many scholars initiate research on the application of deep learning in the recommendation system[7]. Deep learning is usually in combination with traditional recommendation algorithms to improve the performance of original algorithm.

Lei et al. used pre-trained word embedding technology to express the text of the review through deep cooperative neural networks, extracted semantic information from the review, and combined content-based recommendation algorithms to study the implicit relationship between the preferences of the users and the feature of the commodities. This model has achieved good results, proving that the semantics and emotional attitude of comments can improve the accuracy of score prediction. Therefore, combined with deep learning, the recommendation system can effectively analyze the implicit feedback, so that the model built with the content-based recommendation algorithm is more comprehensive and accurate resulting in the better performance of the system[8].

An improved automatic encoder is used to collaboratively filter implicit feedback data by Dawen et al. This model can generate polynomials with neural network parameterization to improve the traditional linear factor model. Bayesian inference is used for parameter estimation, which greatly enhances the modeling ability of this nonlinear probability model. Through comparative analysis, the results prove that this model has a significant improvement in the performance compared with the traditional ones[9].

Vaibhav et al. applied deep neural networks to news recommendation, focusing on whether users have read given articles and the reading order to model the potential characteristics of users and projects [[10]]. First, a two-way LSTM is used to obtain the user's interest over time from these hidden factors. The user's behavior data is used as input, the attention mechanism is used to obtain the interest level, then the similarity between the user and the project is captured, and finally the news recommendation is realized. This model has achieved very good results, and also solved the problem of cold start.

It can be seen from the above literature that with deep learning, recommendation system has more excellent ability to make prediction, and experiments have shown that the application of deep learning models can not only improve the accuracy but also can provide a better user experience. Therefore, in this project LSTM-CNN is applied in the recommendation of movies. Experimental results draw the conclusion that the combination of LSTM-CNN and collaborative filtering algorithm can make a personalized recommendation list for users and the performance of this system exceed the traditional algorithm.

III. THEORETICAL MODEL

A. CNN

Figure 1 shows the structure of CNN:

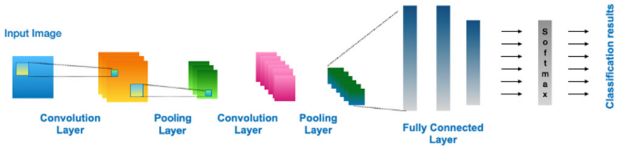

Figure 1. Structure of CNN.

Convolutional Neural Network (CNN) has an Excellent performance in large-scale image processing, and it has three concepts: Local Receptive Fields, Shared Weights and Pooling. Local Receptive Fields means that CNN connect each hidden node to a local area of the image, so as to reduce the number of parameter training. Shared Weights means that in the convolution layer of the CNN, the weights of the corresponding neurons are the same, so the amount of training parameters can be reduced. Pooling use the idea similar to image compression to convolute the image and adjust the size of the image.

CNN contains eight layers: input layer, Convolution layer, Pooling layer, Convolution layer, Pooling layer, Convolution layer, Full connection layer and output layer. In the Convolution layers, we extract features by moving the convolution kernel, multiply the pixels and the corresponding weights of convolution kernel, and finally all the products are added to get

an output. In the Pooling layers, we reduce the dimension of data. We form a new picture by summing the data in the selection box, averaging the data, then multiplying a weight and adding a bias value. Since the sampling weight and bias value of each feature plane are the same, the sampling layer corresponding to each feature plane only has two parameters to be trained.

B. LSTM-CNN

In this project, the LSTM-CNN as shown in figure 2 is applied to extract features from the data. Its structure is as follows. Simply put, an LSTM layer is inserted between the embedding layer and the CNN layer. The principle is that for the vocabulary feature vector in the sentence, Encode through the LSTM layer first, so that the output of each time step includes not only the feature vector information of the current vocabulary, but also the feature information of all vocabularies before the vocabulary, so that the feature vector of each time step can contain more For more information, then, take the output vector of each time step as the embedding layer in the text convolution network.

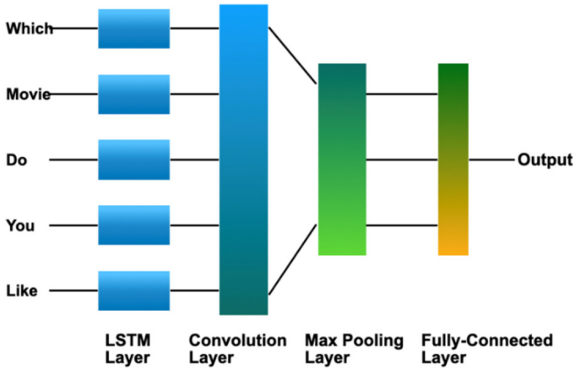

Figure 2. Structure of LSTM-CNN.

C. Algorithm Design

The user's characteristic matrix is mainly generated by embedding user information into the network. When preprocessing data, UserId, MovieId, gender, age, occupation characteristics are all converted into number type, and then this number is used as index of embedding matrix. The embedding layer is used in the first layer of the network, so that the dimension of data input is kept at (n, 32) and (n, 16). Then the full connection layer is converted to the size of (n, 128), and then the full connection layer is converted to the size of (n, 200). In this way, the final output user feature dimension is relatively high, and it also ensures that the features of each user can be fully carried and expressed through features.

For the movie data set, LSTM - CNN is applied to process the movie name. The general process is to get the embedding vector of each word corresponding to the movie name from the embedding matrix. Because the movie name is a bit special, the name length is limited. Here, when using the filter size, the length of 2, 3, 4 and 5 is chosen. Then the text embedding layer is convoluted and pooled by using convolution kernels of sliding 2, 3, 4 and 5 word sizes, and then the dropout operation is used to output the full connection layer.

The feature vectors generated in the above steps are connected by two full connection layers. The first full connection layer is the full connection of movie ID features and movie type features, and then the full connection of LSTM-CNN generated movie name features to generate the final feature set.

IV. Experimental Implementation

A. Experimental Data Set

This project uses movielens 1m data set collated from movielens by the grouplens project group. The data set contains 100 million comments from 6000 users on nearly 4000 movies. The dataset is divided into three files: users.dat, movies.dat and ratings.dat.

Users data includes user ID, gender, age, occupation ID and Zip-code.

"M" represents male while "F" represents female. And age is divided into 7 age groups, occupation is divided into 20 categories and Zip-code is not used. The original user data is shown in Table 1:

TABLE I. Examples of original Users Data

UserID	Gender	Age	OccupationID	Zip-code
1	F	1	10	48067
2	M	56	16	70072
3	M	25	15	55117
4	M	45	7	02460
5	M	25	20	55455

Movies Data includes movie ID, movie name and movie style fields. Here are some examples of Movies Data as shown in Table 2:

TABLE II. Examples of original Movies Data

MovieID	Title	Genres
1	Toy Story (1995)	Animation|Children's|Comedy
2	Jumanji (1995)	Adventure|Children's|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama
5	Father of the Bride Part 2 (1995)	Comedy

There are user ID, movie ID, rating and timestamp fields respectively in Ratings data. Here are some examples of Ratings Data as shown in Table 3:

TABLE III. Examples of original Ratings Data

UserID	MovieID	Rating	Timestamps
1	1193	5	978300760
2	661	3	978302109
3	914	3	978301968

4	3408	4	978300275
5	2355	5	978824291

B. Experimental Environment

The relevant experimental environment is in Table 4:

TABLE IV. EXPERIMENTAL ENVIRONMENT.

Experimental Environment	Environment Configuration
Programming Language	Python 3.7
Deep Learning Framework	Tensorflow 2.1.0

C. Data Preprocessing

- There is nothing needs change for Userid, occupation and movieid.
- Gender: Here 'f' needs to be converted to 0 and 'm' need to be converted to 1.
- Age: The data needs to be transformed into 7 consecutive digits 0-6.
- Genres: The data needs to be converted into a number. First, the categories in genres are converted into strings into a number dictionary, and then the genres field of each movie is converted into a number list, since it is a combination of several genres that some movies are.
- Title: To begin with, a dictionary of text to numbers is produced and convert the description in title to a list of numbers. The year in the title is supposed to be removed.
- Genres and title need to be uniform in length so that they can be easily processed in neural networks. The blank part is filled with the number corresponding to '< pad >.

The data after preprocessing is shown in Table 5 and Table 6:

TABLE V. EXAMPLES OF USERS DATA AFTER PREPROCESSING

UserID	Gender	Age	JobID
1	0	0	10
2	1	5	16
3	1	6	15
4	1	2	7
5	1	6	920

TABLE VI. EXAMPLES OF MOVIES DATA AFTER PREPROCESSING.

MovieID	Title	Genres
1	[295, 1511, 1528, 1528, 1528…	[4, 13, 9, 1, 1, 1,1, 1…
2	[732, 1528, 1528, 1528, 1528…	[7, 13, 18, 1, 1, 1, 1…
3	[3844, 4131, 3737, 1528, 1528…	[9, 16, 1, 1, 1, 1, 1…
4	[595, 2968, 102, 1528, 1528…	[9, 1, 1, 1, 1, 1, 1, 1…
5	[4104, 601, 919, 3583, 3690…	978824291

D. Experimental parameters

In the training process, some super parameters are set for the algorithm in advance, and the final setting results are given in Table 7:

TABLE VII. INITIAL CODE PARAMETERS

Parameters	Values
Num epochs	5
Batch size	256
Dropout keep	0.5
Learning rate	0.0001
Show_every_n_batches	20

E. Experimental Process

Firstly, the data set is divided into training set and test set according to the ratio of 4:1. The following is the partition result of the final training set test set of the algorithm model:

Comparison of parameter adjustment process:

Through the previous 6 steps parameter comparison, in the case of the same learning rate and batch size, num > epochs has a greater impact on the training time; while in the case of the same learning rate and num < epochs, batch < size has a greater impact on loss, and batch < size is 512, and loss has a shaking situation. Finally, the following adjustment parameter is determined to be fixed in the following steps, with the learning rate decreasing, it is found that the loss decreases first, but increases when the learning rate = 0.00005, and finally selects the learning rate as the learning rate = 0.0001.

V. RESULTS AND DISCUSSION

In this project, the MSE function is used to optimize the loss, adjust the learning rate and model parameters, and the resulting loss curve is shown in Figure 3:

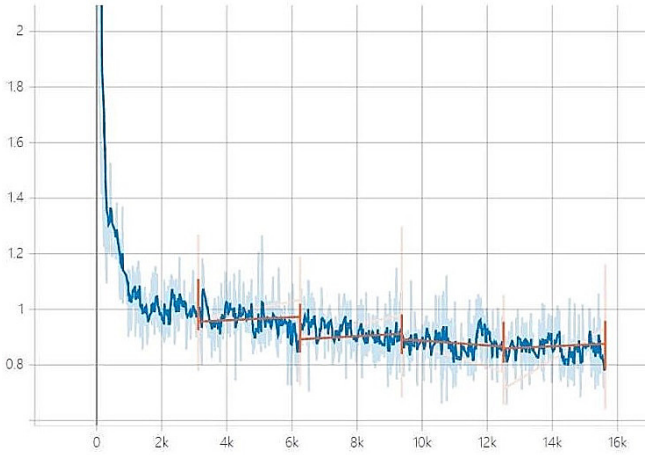

Figure 3. Loss Curve.

By observing the Loss Curve, it can be found that the curve has quickly converged, which shows that the LSTM-CNN model is stable and effective.

In this project, two features are taken as input and are passed into the fully connected layer again till a value is output. The output value is then returned to the real score, and Mean Squared Error (MSE) is used to optimize the loss. Mean Absolute Error (MAE) is applied to evaluate the recommended model. The

smaller the value of the two indicators, the better the prediction result.

$$MSE = \frac{\sum_{(u,i\in E^U)}(|y_{u,i}-\hat{y}_{u,i}|)^2}{E^U} \quad (1)$$

$$MAE = \frac{\sum_{(u,i\in E^U)}(|y_{u,i}-\hat{y}_{u,i}|)}{E^U} \quad (2)$$

In formula (1) and (2) E^U is the size of the dataset used to predict the ratings.

A. Comparison between CNN Model and LSTM-CNN Model

The performance of the CNN-based recommendation algorithm and the LSTM-CNN-based recommendation algorithm are compared in this project, and the program results are shown in Figure 4 and Figure 5 respectively:

```
Step #30650     Epoch   4 Batch 2524/3125   Loss: 0.711231 mae: 0.685501 (41.981189092334576 steps/sec)
Step #30700     Epoch   4 Batch 2574/3125   Loss: 0.818561 mae: 0.698344 (44.553661813990879 steps/sec)
Step #30750     Epoch   4 Batch 2624/3125   Loss: 0.938533 mae: 0.685009 (45.393617691361679 steps/sec)
Step #30800     Epoch   4 Batch 2674/3125   Loss: 0.763609 mae: 0.683320 (44.486124911994881 steps/sec)
Step #30850     Epoch   4 Batch 2724/3125   Loss: 0.710348 mae: 0.675484 (42.856316679707091 steps/sec)
Step #30900     Epoch   4 Batch 2774/3125   Loss: 0.677977 mae: 0.679657 (41.121482132765 9 steps/sec)
Step #30950     Epoch   4 Batch 2824/3125   Loss: 0.774725 mae: 0.690237 (45.121798510163394 steps/sec)
Step #31000     Epoch   4 Batch 2874/3125   Loss: 0.787354 mae: 0.671726 (45.32555960149 25 steps/sec)
Step #31050     Epoch   4 Batch 2924/3125   Loss: 0.795461 mae: 0.681127 (43.847384526863095 steps/sec)
Step #31100     Epoch   4 Batch 2974/3125   Loss: 0.666899 mae: 0.683002 (42.71708207053 45 steps/sec)
Step #31150     Epoch   4 Batch 3024/3125   Loss: 0.689966 mae: 0.673126 (45.872887719753338 steps/sec)
Step #31200     Epoch   4 Batch 3074/3125   Loss: 0.734862 mae: 0.674310 (46.03299824090744 steps/sec)
Step #31250     Epoch   4 Batch 3124/3125   Loss: 0.783089 mae: 0.676481 (46.62266920223435 steps/sec)

Train time for epoch #5 (31250 total steps): 69.42882657051086
Model test set loss: 0.777218 mae: 0.695507
best loss = 0.777218222618103
```

Figure 4. Results of CNN.

```
Step #30650     Epoch   4 Batch 2524/3125   Loss: 0.697417 mae: 0.680071 (50.780072602734442 steps/sec)
Step #30700     Epoch   4 Batch 2574/3125   Loss: 0.780885 mae: 0.684599 (45.196773826477 66 steps/sec)
Step #30750     Epoch   4 Batch 2624/3125   Loss: 0.936506 mae: 0.681362 (49.27240945059489 steps/sec)
Step #30800     Epoch   4 Batch 2674/3125   Loss: 0.757528 mae: 0.678925 (50.186985019422 22 steps/sec)
Step #30850     Epoch   4 Batch 2724/3125   Loss: 0.715252 mae: 0.671493 (46.651646815442 85 steps/sec)
Step #30900     Epoch   4 Batch 2774/3125   Loss: 0.673024 mae: 0.675006 (46.265199730062015 steps/sec)
Step #30950     Epoch   4 Batch 2824/3125   Loss: 0.759148 mae: 0.684148 (49.7582970751265 steps/sec)
Step #31000     Epoch   4 Batch 2874/3125   Loss: 0.779240 mae: 0.667545 (50.885592632241995 steps/sec)
Step #31050     Epoch   4 Batch 2924/3125   Loss: 0.778995 mae: 0.678056 (51.208313849782 74 steps/sec)
Step #31100     Epoch   4 Batch 2974/3125   Loss: 0.662422 mae: 0.679352 (51.183018134364 11 steps/sec)
Step #31150     Epoch   4 Batch 3024/3125   Loss: 0.688300 mae: 0.668558 (50.754657618455255 steps/sec)
Step #31200     Epoch   4 Batch 3074/3125   Loss: 0.732081 mae: 0.670788 (50.618712794583 16 steps/sec)
Step #31250     Epoch   4 Batch 3124/3125   Loss: 0.801664 mae: 0.674337 (44.985557186291 92 steps/sec)

Train time for epoch #5 (31250 total steps): 64.25476789474487
Model test set loss: 0.772415 mae: 0.691739
best loss = 0.7724153995513916
```

Figure 5. Results of LSTM-CNN.

It can be seen from the calculation results that the MSE of the CNN model is 0.7772, the MAE is 0.695507, and the running time is 69.4288. The MSE of the LSTM-CNN model is 0.7724, the MAE is 0.691739, and the running time is 64.2548. The values of MSE and MAE of LSTM-CNN are smaller, and the running time is shorter.

In the above, after completing the model training verification, we actually recommend movies. Here, the produced user feature matrix and movie feature matrix is used to recommend movies, mainly in three ways.

B. Recommend Movies of the Same Type

Based on the result of the cosine similarity between the current movie feature vector and the whole movie feature matrix, the Top-k recommendation list is obtained with the largest similarity. Here, add some random selections to ensure that each recommendation is slightly different. Figure 6 displays the result of this kind of situation:

```
INFO:tensorflow:Restoring parameters from ./save
The movies you watch are:[1401 'Ghosts of Mississippi (1996)' 'Drama']
Here are the recommendations for you:
3680
[3749 'Time Regained (Le Temps Retrouv) (1999)' 'Drama']
2891
[2960 'Beefcake (1999)' 'Drama']
2157
[2226 'Ring, The (1927)' 'Drama']
274
[277 'Miracle on 34th Street (1994)' 'Drama']
949
[961 'Little Lord Fauntleroy (1936)' 'Drama']
INFO:tensorflow:Restoring parameters from ./save
Here are the recommendations for you:
1162
[1178 'Paths of Glory (1957)' 'Drama|War']
523
[527 "Schindler's List (1993)" 'Drama|War']
49
[50 'Usual Suspects, The (1995)' 'Crime|Thriller']
1950
[2019
 'Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954)'
 'Action|Drama']
735
[745 'Close Shave, A (1995)' 'Animation|Comedy|Thriller']
```

Figure 6. The result of way1.

C. Recommend Your Favorite Movies

Based on the user feature vector and the movie feature matrix, the scores of all movies are obtained to select the top-K movies with the highest scores. Random selection parts are also added. The result is shown in Figure 7:

```
INFO:tensorflow:Restoring parameters from ./save
Here are the recommendations for you:
1162
[1178 'Paths of Glory (1957)' 'Drama|War']
523
[527 "Schindler's List (1993)" 'Drama|War']
49
[50 'Usual Suspects, The (1995)' 'Crime|Thriller']
1950
[2019
 'Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954)'
 'Action|Drama']
735
[745 'Close Shave, A (1995)' 'Animation|Comedy|Thriller']
```

Figure 7. The result of way2.

D. What other movies did people watch (like) after watching this movie

First of all, select the top guy who likes a movie and get their user feature vector; then calculate their ratings for all movies; select the movie with the highest rating for each person as the recommendation; also add random selection. The result is shown in Figure 8:

```
INFO:tensorflow:Restoring parameters from ./save
The movies you watch are:[1401 'Ghosts of Mississippi (1996)' 'Drama']
The people who like to watch this movie are:[[4399 'F' 18 4]
 [1585 'M' 56 20]
 [713 'M' 35 7]
 [5458 'F' 18 2]
 [596 'F' 25 1]
 [1855 'M' 18 4]
 [342 'M' 18 12]
 [2338 'M' 45 17]
 [4754 'F' 18 0]
 [3031 'M' 18 4]
 [5533 'F' 50 2]
 [2294 'M' 56 13]
 [394 'M' 18 0]
 [371 'M' 18 4]
 [4800 'M' 18 4]
 [3038 'F' 35 2]
 [5861 'F' 50 1]
 [3901 'M' 18 14]
 [4903 'M' 35 12]
 [4085 'F' 25 6]]
People who like to watch this movie also like to watch it:
3366
[3435 'Double Indemnity (1944)' 'Crime|Film-Noir']
1162
[1178 'Paths of Glory (1957)' 'Drama|War']
523
[527 "Schindler's List (1993)" 'Drama|War']
1180
[1198 'Raiders of the Lost Ark (1981)' 'Action|Adventure']
1950
[2019
 'Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954)'
 'Action|Drama']
```

Figure 8. The result of way3.

VI. CONCLUSION

Through the experiment and given the results, there is a conclusion that LSTM-CNN improves the performance of the algorithm compared with CNN-based recommendation algorithm. Therefore, the experimental results prove that LSTM-CNN is very powerful for the feature extraction of text information, and its effect on the recommendation algorithm is also satisfactory.

However, since the neural network is a black box process, specific details in the process of back propagation and the details of the features extracted by each convolutional layer are unkonwn, and this algorithm lacks certain ability to be interpreted.

Generally, in the industrial world, the amount of user data is massive, and the convolutional neural network consumes a lot of computing resources, so cluster computing is very important. And due to the limited experimental environment of this course, it is still running on a single machine, running a full amount of data on the server cluster later may result in more accurate results.

This algorithm is still inadequate in solving the problems of cold start and sparsity, so future research will focus on solving these two problems.

VII. ACKNOWLEDGEMENTS

Haili wang and Nana Lou contributed equally to this work.

Thanks Professor David Woodruff for patient guidance and generous help, Reyna for answering our questions in time, and thanks the CIS research project for giving us the opportunity and conditions to complete the project together.

REFERENCES

[1] Y. Lecun, Y. Bengio, G. Hinton, 2001. Deep Learning. Nature. 521, pp.436.

[2] Z. J. Sun, L. Xue, Y. M. Xu, 2012. Summary of Deep Learning Research. Application Research of Computers. 29(8), pp.2806-2810.

[3] C. Wang, Z. G. Zhu, Y. X. Zhang, 2016. Recommendation efficiency and personalized improvement of user-based collaborative filtering algorithm. Small Microcomputer System. 37(3), pp.428-432.

[4] A. Albadvi, M. Shahbazi, 2009. A hybrid recommendation technique based on product category attributes. Expert Systems with Applications. 36(9), pp. 11480-11488.

[5] N. N. Chen, 2019. Research on Recommendation System Based on Dep Learning. Guilin University of Electronic Technology.

[6] B. M. Asrwar, G. Karypis, J. A. Konstan, 2001. Item-based colaborative filtering recommendation algorithms. International Conference on World Wide Web. 1, pp. 285-295.

[7] S. Ruslan, M. Andriy, H. Geoffrey, 2007. Restricted Boltzmann Machines for Collaborative Filtering. Proceedings of the 24th International Conference on Machine Learning. 24, pp.791-798.

[8] Z. Lei, N. Vahid, S. Philip, Joint. 2017. Deep Modeling of Users and Items Using Reviews for Recommendation. Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. Pp. 425-434.

[9] L. Dawen, G. K. Rahul, D. H. Matthew, 2018. Variational Autoencoders for Collaborative Filtering. Proceedings of the 2018 World Wide Web Conference. Pp. 689-698.

[10] K. Vaibhav, K. Dhruv, G. Shashank, 2017. Deep Neural Architecture for News Recommendation. in Working Notes of the 8th International Conference of the CLEF Initiative. Pp.142-162.