Deep Learning Methods for Movie Recommendation System

SONG,Di 20733378   LOU,Yuqi 20745448   YUAN,Yanzhe 20728555   JIANG,Yuxin 20710998   WEN,Ze 20711552
HUANG,Yitian 20718885

1 Writing Motivation

Nowadays, the applications of recommendation systems are becoming more and more common in daily life. As for the film industry, movies of different types and different themes appear on the big screen. Therefore, movie recommendation systems are gradually becoming an increasing demand. In fact, there are quite a few movie recommendation systems based on collaborative filtering technology, but most of them suffer the problems of cold start and data sparsity.

Deep learning has gradually become a hotspot in the field of artificial intelligence in recent years. The recommendation system based on deep learning methods can better abstract the features of users and movies, and solve the problems of cold start and data sparsity to some extent. In this project, we plan to explore deep learning methods and apply them into the movie recommendation system to achieve a better quality of recommendation.

2 Data Introduction

As for datasets, we plan to use MovieLens 1M, which is a stable benchmark dataset for movie ratings which contains around 1 million ratings from 6000s users on 4000s movies. This dataset is divided into three files: users.dat for users' details, movies.dat for movies' information, and ratings.dat for users' ratings over the movies they've seen.

2.1 Users.dat

Format: UserID::Gender::Age::Occupation::Zipcode.

- Gender is denoted by a "M" for male and "F" for female.
- Age groups are indicated by numbers: 1 for "Under 18", 18 for "18-24", 25 for "25-34", etc.
- Occupation is numeralized as well : 0 for "other and not specified",1 for "academic/educator", 2 for "artist", 3 for "clerical/admin", etc.

2.2 Movies.dat

Format: MovieID::Title::Genres.

- Titles are identical to titles provided by the IMDB (including year of release)
- Genres are pipe-separated and are selected from the following genres, like: Action, Sci-Fi, Crime, Musical, Comedy, etc.

2.3 Ratings.dat

Format: UserID::MovieID::Rating::Timestamp.

- UserIDs range between 1 and 6040 and MovieIDs range between 1 and 3952
- Ratings are made on a 5-star scale (whole-star ratings only)
- Timestamp is represented in seconds since the epoch as returned by time

3 Models

3.1 Problems of Traditional Recommendation Systems

Traditional recommendation systems based on collaborative filtering algorithm have two main problems: cold start and data sparsity.

Cold start means that the accumulated data of the system is too small to make personalized recommendation for new users, which is a big problem in product recommendation. [6]

Data sparsity in recommendation systems is caused by the fact that the process of data mining and item recommendation rely highly on the rating data of users on items. However, users of the recommendation system often do not leave the rating data for some reasons, resulting in the lack of data that the recommendation system can rely on. Therefore, it is difficult for the recommendation system to find similar users based on the data, which eventually leads to the low accuracy of the recommendation system and the dissatisfaction of the target users.

Traditional systems only use the user's rating matrix for movies, and make little use of features of users and movies. Moreover, due to the sparse rating matrix, the recommendation accuracy of the recommendation system is obviously reduced.

Based on the above analysis, we proposed 4 deep architectures including Auto Encoder, Variational Auto Encoder, Text-CNN based model and BERT based model to predict the vacant rating value. After that, we can get a fully rated matrix, which can be used for the follow-up movie recommendation process.

3.2 The First Method

The first model is Auto Encoder. Auto Encoder is an artificial neural network learning the representation (encoding) of input data, which is usually used as a method of dimension reduction. Considering the difficulty for collaborative filtering models to learn efficient latent factors due to cold start and data sparsity. [3]

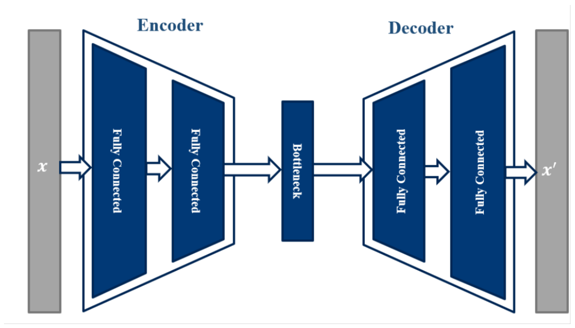

Figure 1: Auto Encoder

Although AutoEncoder is often used in representation learning, we can design it to suit this task. As the above picture shows, the model has 2 main parts, one is encoder, another is called decoder. They are both composed of fully connected layers. In this scenario, the key point is how to design the loss function due to a large number of missing rating data.

For example, suppose that the batchsize is set to equal to 2 which means each batch contains 2 users and suppose the number of movies equals to 6. It can be seen that there exists some nan values. Firstly, these nan values are masked with 0, and this batch is put into Auto Encoder, after that process the reconstructed data is formed. Then the values that are original missing with 0 are remasked and the batch MSE is computed. Through this method, only the loss of reconstruction of rated data will be considered, which may ignore the influence of the vacant values. What's more, a large dropout is added after the first fully connected layer of encoder layer. In this way the model can reconstruct the data more effectively.

The Fogure-2 shows the training process of auto encoder, it's clearly seen that the training loss keeps falling, but the validation loss increases after a slight drop. We thought it may be caused by overfitting.

3.3 The Second Method

The second model is Variational Auto Encoder, its improvement over AE is to use neural networks to learn a normal distribution, and then reconstruct

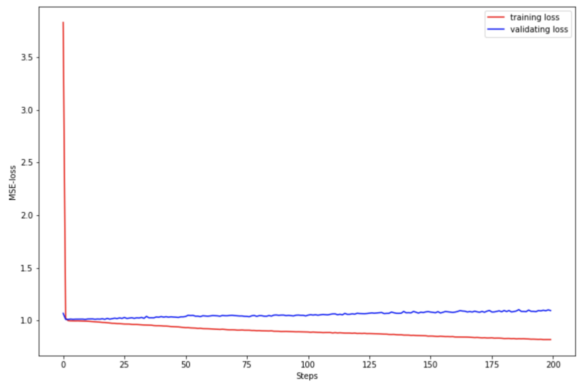

Figure 2: Training Process of AE

data through sampling from the latent vector. [1]The loss of the model includes MSE and KL-divergence. As the below picture shows, compared with AE, the training loss and validation loss of VAE both drop steadily, which is because that the model learned a better data distribution. [5]

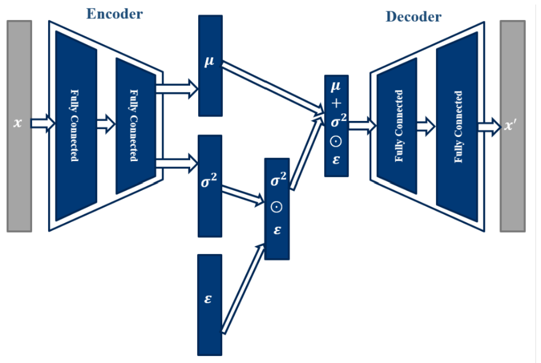

Figure 3: Variational Auto Encoder

However, these two methods only use the information of the rating data, without considering the user data and movie data, which is their main disadvantage. Our next two models improve this situation by exactly using users' and movies' data.

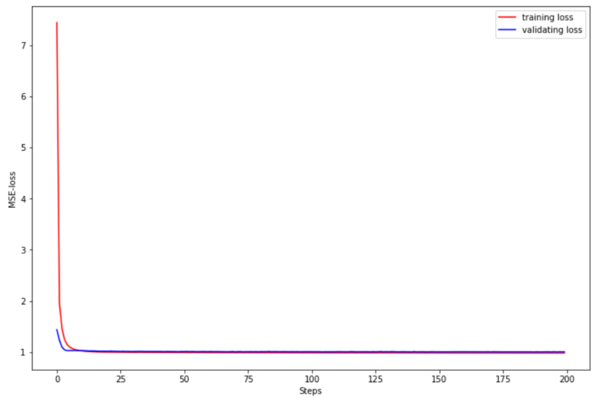

Figure 4: Training Process of VAE

3.4 The Third Method

Apart from the method of using rating data to predict rating data based on models like AE and VAE, we also implement a Text-CNN based neural network structure using both users' and movies' data as the third model. The above picture shows the exact structure of the third model. At first we noticed that besides these meta-data like 'user-id', 'age', etc., which can be easily fit into the neural network embedding layers, there are features like the movie titles that we should pay more attention to because it may

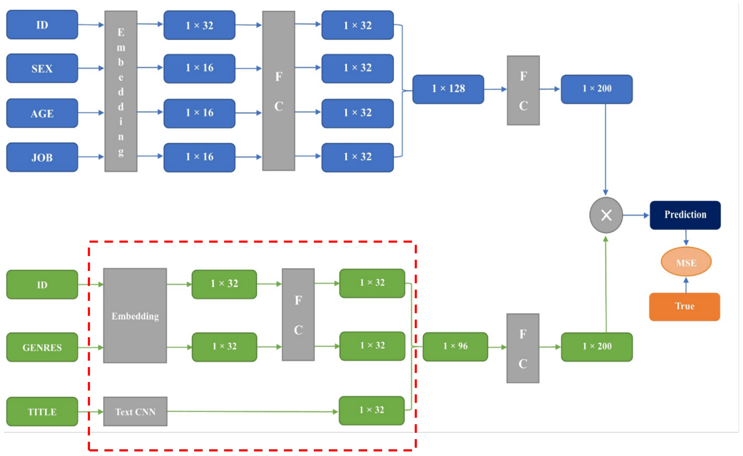

Figure 5: Text-CNN based Neural Network Structure

contain more information. TextCNN applies convolutional neural network to text related tasks, so for special features like movie titles, use TextCNN for embedding text data may achieve a better result.

First we preprocess the raw data. For meta data, we just keep the original data fomula, and for categorical data we transfer it to numbers. Then these data are put into embedding layer and dense layer. For relatively comprehensive data like movie titles, we build a number dictionary on every word in titles and then numerize titles into a list of numbers, then we pass the list into our Text CNN neural network.

In Text CNN, we first do a embedding of the input, and then pass the intermediate output into a convolutional layer. One major difference between traditional CNN and Text CNN is that the kernel size is number of words in input multiplies the vector dimension. [4] After then Convolutional layer we did a Max Pooling, then pass the data to multiple Dense layers with Dropout and finally output the result from Softmax.

In the end, all features are collected and a feature matrix for each of users' data and movies' data is created. Next a matrix multiplication of users' and movies' feature matrix is done as the inferred ratings from the model. The model will learn through comparing the inferred ratings and true ratings from training data. The training and testing process are shown Figure-6 and Figure-7.

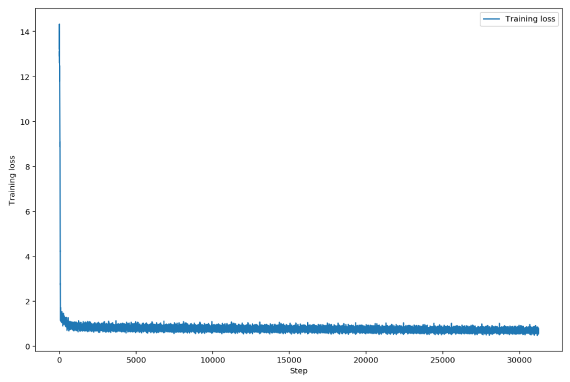

Figure 6: Training Process of CNN

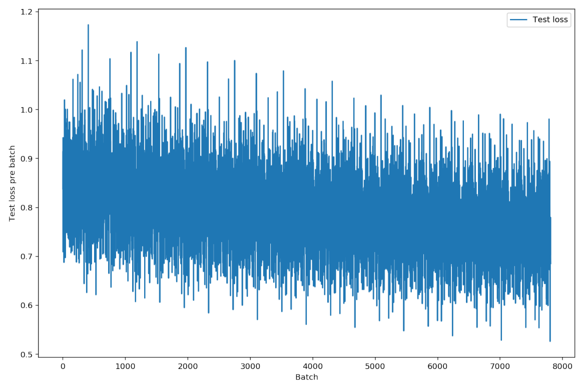

Figure 7: Testing Process of CNN

3.5 The Forth Method

BERT (Bidirectional Encoder Representations from Transformers) is an state-of-art language representation model that maps a token sequence into a sequence of vectors. It's also a state-of-art pre-trained model to perform word-embeddings. [2]

We use BERT to embed the movie titles into vectors as a replacement of our Text CNN structure. As the result shows the bert based model is slightly better than Text CNN.

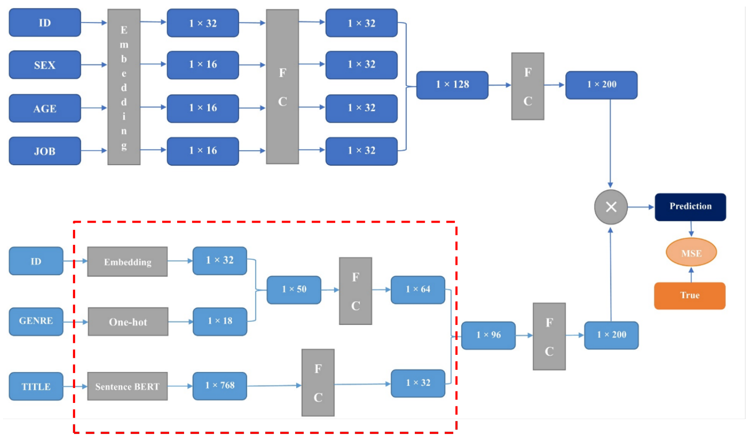

Figure 8: BERT based Neural Network Structure

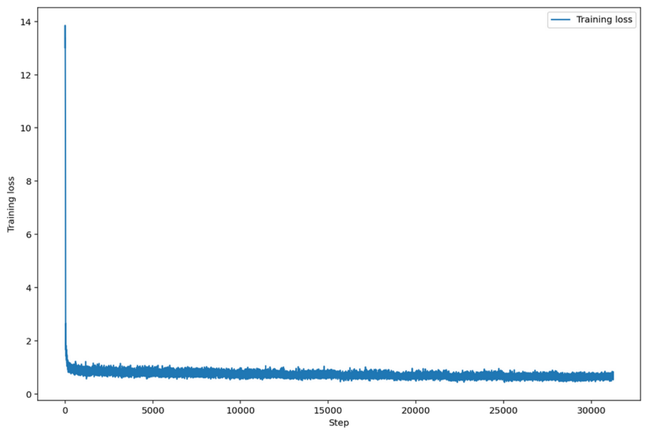

Figure 9: Training Process of BERT

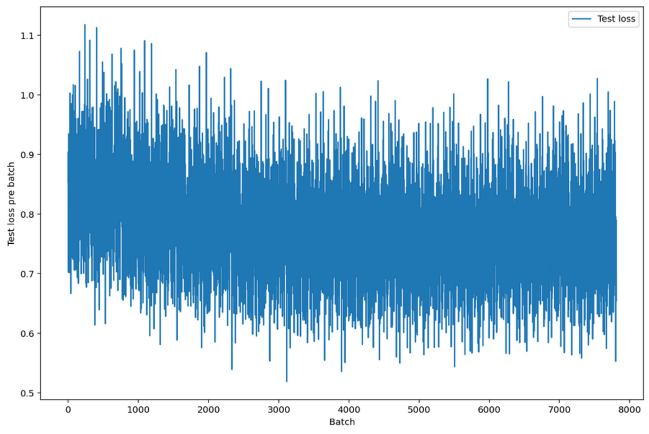

Figure 10: Testing Process of BERT

4 Results and Conclusions

4.1 Results

The results of our 4 models and some traditional models are shown below. It is obvious that deep learning models achieves a much better performance. Among our 4 models, Bert-based model achieves the best performance on MSE of test data, slightly better than our Text CNN based model.

Models	**Score**
Auto-Encoder	1.0837
Variational-Auto-Encoder	0.9956
Text-CNN-based	0.7521
BERT-based	**0.7507**

Table 1: Models Score.

4.2 Conclusions

To sum up, our system can deeply mine users', movies' and ratings' data, smooth the problems of cold start and data sparsity through deep learning, thus improving the traditional model and providing a better recommendation.

As a result, our movie recommendation system provides users with the following three services.

- The first service is to recommend movies of the same type for a certain movie input.
  Fistly, we calculate the cosine similarity between the feature vector of the current movie and every row of the entire Movies' feature matrix, and then we take the Top-K best similarity for recommendation.
- The second service is to input a certain user and recommend movies for the user
  Firstly we multiply the user's feature vector corresponding to the input user id with all vectors in the movies' feature matrix, and then we select the Top_K highest score for recommendation
- The third service is to recommend other movies liked by users who have watched this movie
  First we multiply all feature vectors corresponding to the movie id with the users' feature vector separately, then take each one's the top k types as the id list. Next we multiply each id in the id_list with the movies' feature vector, take the highest value. And finally choose 5 for recommendation.

References

[1] Yifan Chen and Maarten de Rijke. A collective variational autoencoder for top-n recommendation with side information. In *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems*, pages 3–9, 2018.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Diana Ferreira, Sofia Silva, António Abelha, and José Machado. Recommendation system using autoencoders. *Applied Sciences*, 10(16):5510, 2020.

[4] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[5] Xiaopeng Li and James She. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 305–314, 2017.

[6] Jian Wei, Jianhua He, Kai Chen, Yi Zhou, and Zuoyin Tang. Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications*, 69:29–39, 2017.