Hybrid neural network based recommendation system integrating user information

Chengyuan Deng
Department of Computer Science
cd751@scarletmail.rutgers.edu

Xuan Wang
Department of Computer Science
xw285@scarletmail.rutgers.edu

Haodong Lu
Department of Computer Science
hl821@scarletmail.rutgers.edu

ABSTRACT

Multiple approaches have been proposed to construct recommendation systems, while the concept of integrating multi-layer neural networks captivates increasing attentions. In this paper, we present a composite model integrating convolutional neural network with sliding window to train movie title feature while the others are trained by multiple neural networks, with MovieLens dataset as applied data. Two aspects of function are expected, first to predict ratings given a specific user and movie not rated yet, second to customize recommendation list,including favorite movies with the knowledge of user, similar movies with a given movie, and other movies preferred by other users. The predicted ratings are evaluated by MAE and RMSE, while the item recommendation is evaluated by precision, recall, f-measure and NDCG. To perfect the evaluation, we also implemented a simple matrix factorization method to achieve two functions and compare the six evaluators.

KEYWORDS

Recommendation system, multi-layer neural networks,item embeddings, user-integrated, matrix factorization, pyspark

1 INTRODUCTION

Along with the unagitated development in theories of deep neural network, it has been witnessed eventually in recent years the fabulous performance of deep learning in certain research fields or industries. Convolution neural network, for example, has indisputably become the most dominating approach of image recognition. Whereas, the utilization of a neural network approach shall not be absolutely restricted in certain fields, with properly established models and collected data, the performance of deep learning in new area is still prospective. Recommendation system could be a potential example.

Collaborative filtering has been a classical approach to design recommendation system model and adopted most-commonly in real applications, due to its capability of filtering features that not straightforwardly understood by computer, such as abstract concepts concerning personalities, aesthetic preference, in the way of collaborate with other users' history. Furthermore, it's able to provide recommendation from different categories that the user has potential preference in yet not discovered by himself. Customized recommendation, high automation, together with comprehensive utilization of history data, makes collaborative filtering highly-efficient in diverse situations.Nevertheless, problems are still being discovered about collaborative filtering, such as its dependency towards history data, bad initial performance for new user, and limited scalability of the system.

What promotion would deep learning make to recommendation systems? Arguments include its capability of direct feature extraction from data with highly-accurate representation, robustness against noise, and convenience of overall data processing.The proposed concepts of applying deep learning in recommendation system are diverse and new approaches are being invented at random time, following are several methods evaluated to be proper: Learning item embeddings; Deep collaborative filtering; Session-based recommendation with RNN; Deep semantic similarity model; Deep composite models.

In respect of our work, we established convolution neural networks with sliding window to train movie title vector, together with deep neural networks to train other feature representation vectors, which are occupied by embedding lookup. Therefore, it's a deep composite model with learning item embeddings. More details will be discussed in the Proposed Model section.

2 RELATED WORK

The current recommendation systems use quantitative explicit feedback implicit feedback (such as users click history, viewing history,etc.) which is larger and denser than explicit feedback dataset to explore user preferences. Matrix Factorization based on Bayesian formulation is proposed to uncover the latent factors in implicit feedback data, such as BPR-MF. Some methods based on multi-type auxiliary implicit feedback or heterogeneous implicit feedback are also proposed for integrating data from multiple sources.

Researchers and practitioners have known for years that incorporating richer information source beyond numerical ratings can promote recommendation performance. This leads to research on hybrid recommendation techniques. Beyond numerical ratings and textual reviews, visual images reveal user visual preference towards different items.

Though achieving better performance against modeling ratings alone, previous models (including deep approaches) are usually limited to pre-selected information sources or domain knowledge, thus researchers have to develop different models for different types of user-item interactions. Recent promising advances on representation learning shed light on this problem. With well established representation learning theories on texts, images, audios , and many others, we can conduct joint representation learning

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

on heterogeneous information sources in a shared space for more informed recommendation. Furthermore, by building representation learning on top of pair-wise learning to rank techniques , we are able to achieve highly promoted top-N recommendation performance, which is closely related to the business values in real-world recommender systems.

3 PRELIMINARIES

- Data:
  Our project uses the ml-1m dataset in MovieLens[7], which contains 6,040 users, 3,900 movies and 1,000,209 ratings. These users, movies and ratings are stored in three files, movies.dat, ratings.dat and users.dat.
  Movies.dat contains:
  - MovieID: 1-3952
  - Title:movie title, containing publishment year
  - Genres:comedy, action, documentary, etc.
  Users.dat. contains
  - UserID: 1-6040
  - Gender: M-Male, F-Female
  - Age: under 18, 18-24, 25-34,35-44...
  Occupation: 0-20
  Ratings.dat. contains:
  - UserID, MovieID and Timestamp
  - Rating: 0,1,2,3,4,5
- Data Downloading:
  We use datadownload.py to automatically download dataset and decompress file into the current directory. This program contains two functions downldata and extractdata, which show the downloading process.
- Data Processing:
  We use dataprocessing.py to automatically process dataset. This program mainly contains four functions. The first three functions are used to deal with users, movies and ratings, and the last function getfeature is to merge and then split new features and target data. The final data is stored in local pickle file.
- Stratified data splitting:
  We implemented spark_split.py to split the data, for each user, we make sure that 80% of his data splitted into traning set and 20% splitted into testing set

4 PROBLEM FORMALIZATION

4.1 Problem Overview

Given a user, our research problem is to predict user's ratings towards movies, in this case we gained prior knowledge from training set and we want to predict ratings in the testing set. Furthermore we want prediction error as small as possible.
A naive approach is to reconstruct user-item-rating matrix and compare with the test data. For each user u is associated with vector p_u, each movie is associated with vector q_m, the rating is predicted as

$$\hat{r}_{u,m} = q_m^T * p_u \tag{1}$$

, we generalized the problem to:

$$\min_{q,p} \sum_{(u,m)\in\mathcal{D}} (r_{u,m} - q_m^T * p_u)^2 + \mathcal{L}_2(q_m) + \mathcal{L}_2(p_u) \tag{2}$$

, where $r_{u,m}$ is the know rating, $\mathcal{D}$ is the data set, $\mathcal{L}_2$ is the regularizer.
Since this is a concave function, we cannot directly find a solution, so we either use iterative approximation method like ALS(Alternating Least Square)[9] or construct a solvable neural network.

4.2 Evaluations

Evaluations are indispensable to test the performance of our model. "Mean absolute error"(MAE) and "root mean square error"(RMSE) are considered as classical and effective methods to evaluate the predicted ratings. Straightforwardly, the mean absolute error is calculated by:

$$MAE = \frac{\sum_{i=1}^{n} |x_i - y_i|}{n} \tag{3}$$

where x_i and y_i represent corresponding ratings in test set and predicted results respectively. RMSE is a quadratic scoring rule measuring the average magnitude of the error, defined by:

$$RMSE = \sqrt[2]{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}} \tag{4}$$

To evaluate the suitability of item recommendation list, four approaches are included in our model: Precision, Recall, F-measure and Normalized Discounted Cumulative Gain (NDCG).

- Precision Precision calculates percentage of testing items in the 10 recommended items, in other word, it measures the correctness of recommended item. Assume $R(u)$ to be the recommendation list based on training set, while $T(u)$ to be the recommendation list based on test set. Formula of precision is derived as:

$$Precision = \frac{\sum_u |R(u) \cap T(u)|}{\sum_u |R(u)|} \tag{5}$$

- Recall In correspondence with precision, recall actually indicates percentage of recommended testing items in all testing sets, it measures the extent of comprehensiveness of the recommendation. Formula of recall is derived as:

$$Precision = \frac{\sum_u |R(u) \cap T(u)|}{\sum_u |T(u)|} \tag{6}$$

- F-measure F-measure is always referred as integrated indicators especially when contradictions occur between precision and recall.Therefore, F-measure is actually weighted harmonic mean of precision and recall, the following formula represents F-measure, where α is a parameter.

$$F = \frac{(\alpha^2 + 1)P * R}{\alpha^2(P + R)} \tag{7}$$

  Set $\alpha = 1$, we have the common definition of F-measure:

$$F = \frac{2 * P * R}{P + R} \tag{8}$$

- NDCG Once a recommendation list is returned, a relevance value of each item is required to evaluate the list, which is

also regarded as gain.Cumulative Gain is actually the sum of all relevance values. Thus, we have:

$$CG_K = \sum_{i=1}^{k} rel_i \tag{9}$$

where k=10 in our experiments. Nonetheless, when a recommendation is presented in real application, the sequence of the recommended item dominate the users' priority to some extent. Therefore, we would like the results with higher relevance come first in the recommendation list. Then Discounted Cumulative Gain is put forward as:

$$DCG_K = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{10}$$

The limitation of DCG reflects in incapability of evaluation across recommendation lists, whereas performance of a recommendation system is assessed by multiple recommendation results. To overcome this, we introduce Normalized DCG and Ideal DCG.IDCG is the DCG of the best sequence in recommendation list, and for each user u, NDCG is defined by:

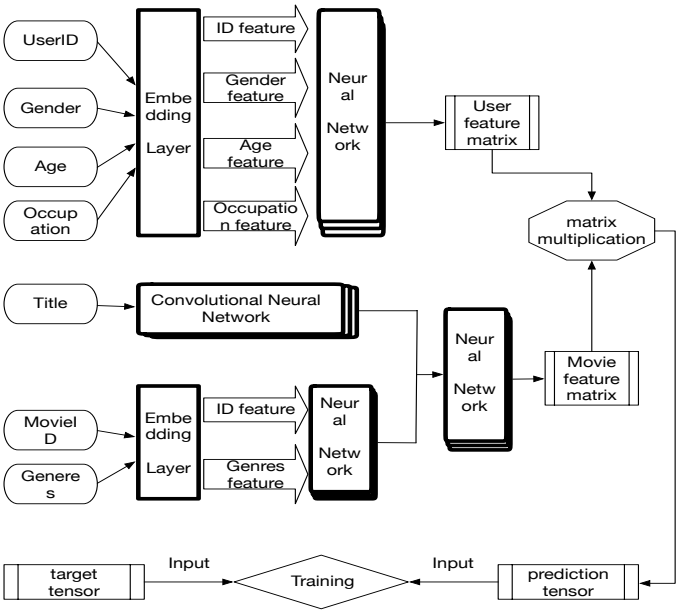
$$NDCG_u^k = \frac{DCG_u^k}{IDCG_u^k} \tag{11}$$

5 THE PROPOSED MODEL

5.1 Neural Network Model

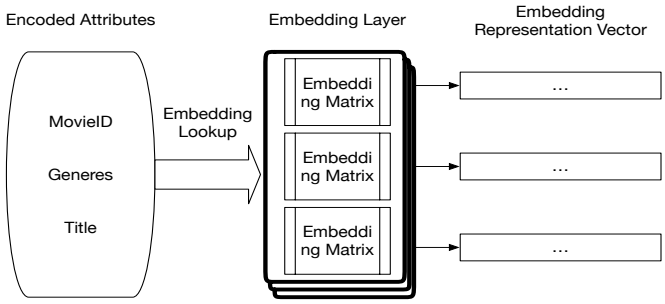
5.1.1 Model Overview. We developed a multi-layered neural network [10][4][2][8]

Figure 1: Overview



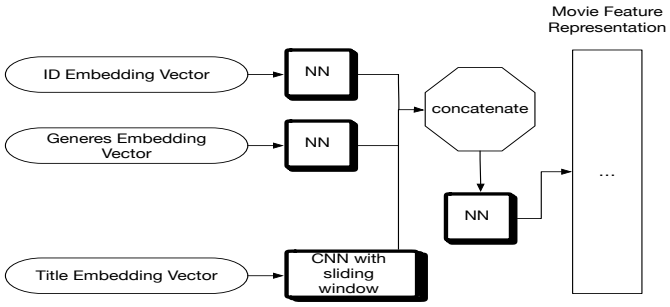
5.1.2 Movie Embedding Layer. Preprocessed movie data contains 3 encoded features: 'MovieID', 'Title', 'Genres', our Movie Embedding Layer performed embedding operation [2]

Figure 2: Movie Feature Processing



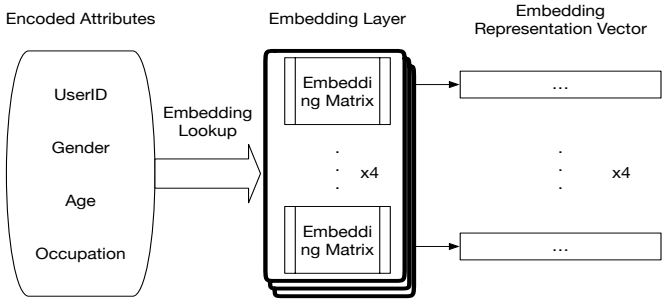
5.1.3 Movie Neural Networks. For movieID and Genres vectors we constructed 2 neural networks separately for training. For title representation matrix we expand it to 3 dimensions first and then use a sliding window to do convolution on this 3D representation. We combined outputs from this 3 neural networks into 1 input for a new neural network, the output from this neural network is the movie feature representation.[3]

Figure 3: Nerual Networks



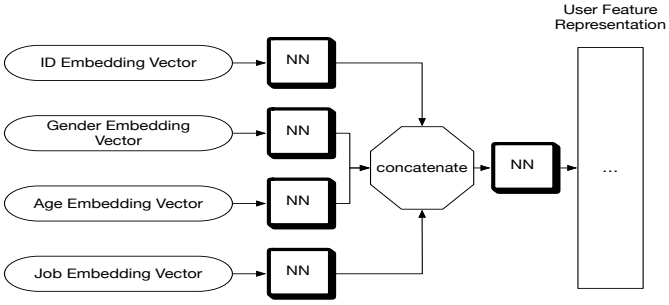
5.1.4 User Embedding Layer. Similarly, our User Embedding Layer performed embedding operation on 4 encoded features: 'UserID', 'Gender', 'Age', 'Occupation'.[4]

Figure 4: User Feature Processing



5.1.5 User Neural Networks. We input 4 embedding representation vector into 4 neural networks respectively, we concatenate 4 outputs into 1 input and go for another neural network, the ouput should be expected user feature representation. [5]

Figure 5: Neural Networks



5.1.6 Training. We got user feature representation and movie feature representation from previous feature processing. Let $\mathbb{U}$ be the user feature matrix, let $\mathbb{M}$ be the movie feature matrix, we performed matrix multiplication and the result is considered to be our model prediction. [14]
Let $prediction = \mathbb{U} \times \mathbb{M}$ be the feature matrix, y be the target value. We minimize loss $||prediction - y||_2$ using Adagrad optimizer[5]. Recall that AdaGrad **??**:

Algorithm 1: AdaGrad Alg

Input: α, learning rate, $L(\cdot)$, loss function
Initialize diagonal matrix $G \in \mathbb{R}^{d \times d}$, smoothing term ϵ;
For each turn:
$g_{t,i} \leftarrow \nabla_\Theta L(\Theta_i)$
$G_{t,ii} \leftarrow G_{t,ii} + \sum_{\tau=1}^{t} g_{\tau,i}^2$
$\Theta_{t+1,i} \leftarrow \Theta_{t,i} - \frac{\alpha}{\sqrt{G_{t,ii}+\epsilon}} g_{t,i}$

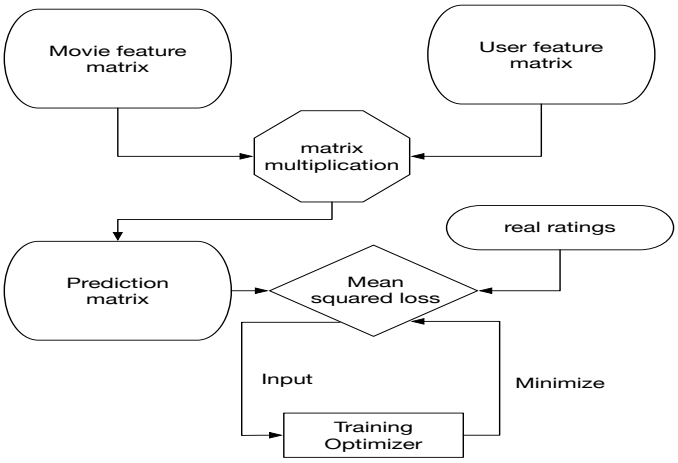
5.2 Matrix Factorization

In addition to the neural network model, we proposed a trivial Matrix Factorization approach[1][11] as a comparison. Apache spark is used to handle data set faster.

5.2.1 Overview. The basic idea for matrix factorization is to find 2 matrices, in this case we want to find $R = M * U^T$. M contains features of movies, U contains features of users. We need to find latent factors of users and movies: U and M, so that the product of 2 can fit the known data.

5.2.2 Motivation. When we were training our data set in the neural network, it usually cost a lot of time, a relatively simple and fast approach is necessarily needed for comparison and verification, because we are storing and computing big data in the computer RAM, automatically Map-Reduce technique came to our mind and we prefer to use Spark rather than Hadoop. On the other hand we want to use a classic and dictinct Collaborative Filtering method, so Matrix Factorization is used.

Figure 6: Training Step



5.2.3 Implementation. For Apache spark, we used pyspark, we read the 'ratings' text file and store in RDD(Resilient Distributed Datasets), we used pyspark.mllib.recommendation library: ALS(Alternating Least Squares matrix factorization)[9] to train the model.

Algorithm 2: Alternating Least Square Alg

Input: A, user-item matrix
Initialize U^1 and V^1;
For k =1,2,...
$U^{k+1} = \arg\min_U f(U, V^k)$
$V^{k+1} = \arg\min_V f(U^{k+1}, V)$

For evaluation part, in pyspark.mllib.recommendation the MatrixFactorizationModel can predict from 'ratings' RDD, the result is a tuple (x.product, x.ratings). '.join' operation can join predictions and test set RDD by key thus we can compute MAE and RMSE easily.
For recommendation part, we read 'movies' text file and store in RDD, still use '.join' operation to join predictions and movie_titles. We use naive approach: sort the result RDD in descending order and take top 10 as recommendation.

6 EXPERIMENTS

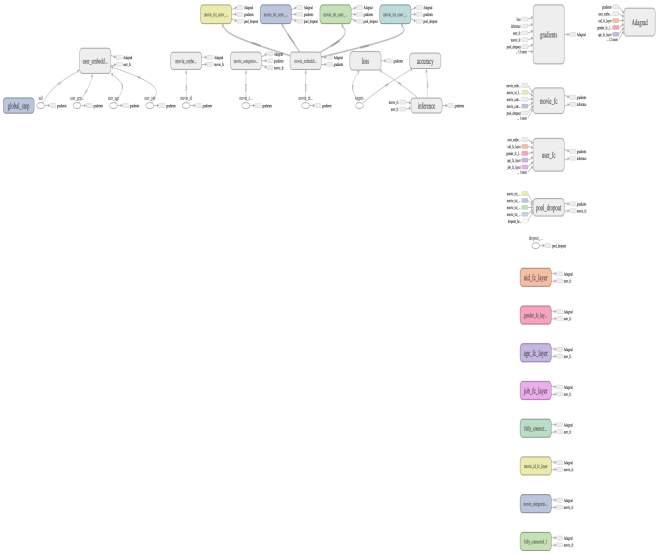
6.1 Neural Network setup

We used TensorFlow[4] to build our network, and the following graph is how it is constructed in the computer[7]:
We set $epochs = 1, batch\_size = 256, drop\_out\_keep = 0.5, learning\_rate = 0.0001$

6.2 Predicting ratings

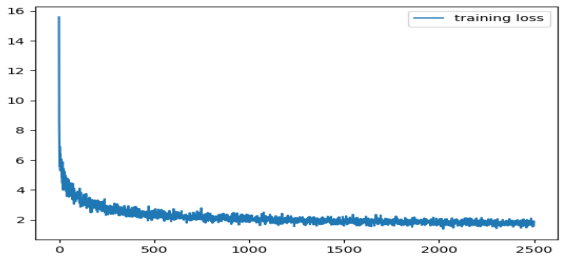
After the feature representation was obtained from two neural network models(movie_nn and user_nn), multiplication of two vectors produces the output prediction of the model, indicating the rating of movies from specific users. The loss function selected in the

Figure 7: Neural Network Graph



training model is quadratic loss function. The plot of loss function is provided in figure 6.

Figure 8: Training Loss



The acquisition of the predicted rating of a movie given by a certain user follows the forward propagation of trained model. For example, provided user_id=12 and movie_id=359, the rating returned is 3.205, three other experiments are implemented with different user and movie, details shown in figure 7.

In rapid sequence, we would like to implement the evaluation method MAE and RMSE, for each item in test set, compare the predicted results with actual ratings after extraction of user_id and movie_id in each item. The ultimate results for both evaluations are $MAE = 0.99$, $RMSE = 1.23$

6.3 Item recommendation

The quality and accuracy of customizing acts as significant reflection of the performance of a certain recommendation system. In this section, recommendation results will be demonstrated in three aspects: Top n recommendation list for a specific user; Top n similar movies provided a movie; Top n other movies users may like given a specific movie. N will be set to 10 in all experiments for consistency and clarity, but is scalable to any value in real applications.

Figure 9: Predicted ratings

```
For user:12, predicting the rating for movie:100 [array([[3.1954303]], dtype=float32)]

For user:12, predicting the rating for movie:359 [array([[3.2047985]], dtype=float32)]

For user:180, predicting the rating for movie:40 [array([[4.5370526]], dtype=float32)]

For user:135, predicting the rating for movie:256 [array([[3.7295978]], dtype=float32)]
```

6.3.1 Top n recommendation list for user. Straightforwardly, the recommendation list can be derived from a predicted ratings of given user to all movies that he hasn't rated yet, the movies with highest K rating values will be selected into the recommendation list. A detailed experiment results is shown in figure 8

Figure 10: Top 10 recommendation list

```
Top 10 recommendation list:
[2468 "Jumpin' Jack Flash (1986)" 'Action|Comedy|Romance|Thriller']
[3301 'Whole Nine Yards, The (2000)' 'Comedy|Crime']
[1389 'Jaws 3-D (1983)' 'Action|Horror']
[2769 'Yards, The (1999)' 'Crime|Mystery']
[3216 'Vampyros Lesbos (Las Vampiras) (1970)' 'Horror']
[2003 'Gremlins (1984)' 'Comedy|Horror']
[2027 'Mafia! (1998)' 'Comedy|Crime']
[2787 "Cat's Eye (1985)" 'Horror']
[2028 'Saving Private Ryan (1998)' 'Action|Drama|War']
[1568 'MURDER and murder (1996)' 'Crime|Drama|Mystery']
```

6.3.2 Top n similar movies. With feature representation matrix for movies, we are capable of utilizing cosine similarity to measure the differences between movies. The inclination to cosine similarity rather than others is due to the efficiency of avoiding concentration on numeric difference but dimensional difference, which is more proper for recommendation motivation. The figure below shows a specific experiment result of top n similar movies recommendation.

6.3.3 Top n other favorite movies. The last recommendation function concentrates on movies potentially preferred by people who has already watched the given movie. User features is integrated with the knowledge of the top n users who like the given movie, the ratings toward all movies from these users will then be predicted. The movie rated highest by each user will be included in the recommendation list. Here shows an example in figure 10 and figure 11.

Figure 11: Top 10 similar movies

```
Top 10 movies similar with [666 'All Things Fair (1996)' 'Drama']:

[666 'All Things Fair (1996)' 'Drama']
[3379 'On the Beach (1959)' 'Drama']
[1139 'Everything Relative (1996)' 'Drama']
[1228 'Raging Bull (1980)' 'Drama']
[1559 'Next Step, The (1995)' 'Drama']
[926 'All About Eve (1950)' 'Drama']
[2236 'Simon Birch (1998)' 'Drama']
[1820 'Proposition, The (1998)' 'Drama']
[758 'Jar, The (Khomreh) (1992)' 'Drama']
[1673 'Boogie Nights (1997)' 'Drama']
```

Figure 12: Return user source

```
The movie specified: [1024 'Three Caballeros, The (1945)' "Animation|Children's|Musical"]
2019-04-26 11:24:33.263678: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU
WARNING:tensorflow:From D:\python\lib\site-packages\tensorflow\python\training\saver.py:12
Instructions for updating:
Use standard file APIs to check for files with this prefix.
People gave good ratings of this movie: [[1199 'M' 50 13]
 [5301 'M' 25 17]
 [2547 'F' 35 0]
 [728 'M' 25 6]
 [4463 'M' 50 7]
 [2143 'M' 18 4]
 [5939 'F' 45 15]
 [1335 'M' 35 0]
 [4842 'F' 35 1]
 [2056 'M' 25 4]]
```

Figure 13: Top 10 other favorite movies

```
[3670 'Story of G.I. Joe, The (1945)' 'War']
[2602 'Mighty Peking Man (Hsing hsing wang) (1977)' 'Adventure|Sci-Fi']
[2944 'Dirty Dozen, The (1967)' 'Action|War']
[3745 'Titan A.E. (2000)' 'Adventure|Animation|Sci-Fi']
[3670 'Story of G.I. Joe, The (1945)' 'War']
[3745 'Titan A.E. (2000)' 'Adventure|Animation|Sci-Fi']
[1068 'Crossfire (1947)' 'Crime|Film-Noir']
[3471 'Close Encounters of the Third Kind (1977)' 'Drama|Sci-Fi']
[2039 'Cheetah (1989)' "Adventure|Children's"]
[266 'Legends of the Fall (1994)' 'Drama|Romance|War|Western']
```

6.3.4 Evaluations for recommendation list. Four approaches of evaluation will be discussed in this section: precision, recall, f-measure and NDCG.Formula of the first three evaluators have been provided above, the strategy is accumulating precision and recall for each recommendation for a user, then calculate the average. As for NDCG, we define relevance score with the following rules:

- If the recommended movie has been recorded in the dataset with rating¿4, set the relevance score to 2.
- If the recommended movie has been recorded in the dataset with rating¿3, set the relevance score to 1.
- For other circumstances, namely, rating¡3 or movie not in dataset, set the relevance score to 0.

A review of the results of four evaluators is shown below:

Evaluators	NN Model	MF Model
Precision	0.0151	0.0232
Recall	0.0249	0.0111
F-measure	0.0193	0.0150
NDCG	0.0590	0.0783

Microsoft benchmark[3] uses Movielens 100k but is still a valuable reference is shown below:

Evaluators	ALS	FastAI	SVD
Precision	0.0437	0.1303	0.091304
Recall	0.0145	0.0538	0.031478
F-measure	N/A	N/A	N/A
NDCG	0.0399	0.1478	0.0963

ALS is the same method as MF model but different parameters used, FastAI is a open source library, SVD is Singular Value Decomposition method.

6.4 MF Model Setup

Using pyspark and RDD data structure is usually faster, but it is very costly to memory, for this case we set the memory allocation limit for spark is at least 16 GB.
We refer MyMediaLite[6] for hyperparameter choosing, we set $latent\_factors = 10$, $\#of iterations = 10$, $regularizer\lambda = 0.1$.

6.5 Matrix Factorization Result

6.5.1 Evaluation. For testing data the $RMSE = 0.8706276607946675$
For testing data the $MAE = 0.6973634563104943$

6.5.2 Recommendation. For user 1996, we gives the top 10 recommendations, (rating,(movieID,movieTitle,# of movie reviews))

Figure 14: Top 10 movies for User:1996

```
(5.7099953600917654, (572, 'Foreign Student (1994)', 2))
(5.071304517938943, (1851, 'Leather Jacket Love Story (1997)', 2))
(5.039702385469181, (3338, 'For All Mankind (1989)', 27))
(4.984492824056332, (3172, 'Ulysses (Ulisse) (1954)', 1))
(4.943364158666611, (2905, 'Sanjuro (1962)', 69))
(4.884068807786474, (527, "Schindler's List (1993)", 2304))
(4.873440289671931, (318, '"Shawshank Redemption', 2227))
(4.815397335096439, (1198, 'Raiders of the Lost Ark (1981)', 2514))
(4.769736777576786, (787, '"Gate of Heavenly Peace', 3))
(4.75921837350444, (989, 'Schlafes Bruder (Brother of Sleep) (1995)', 1))
```

6.6 Prediction Evaluation

7 CONCLUSIONS AND FUTURE WORK

In this project we successfully implement 2 independent models for ratings prediction and item recommendation. A multi-parts neural

Figure 15: RMSE

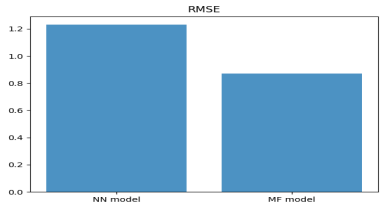
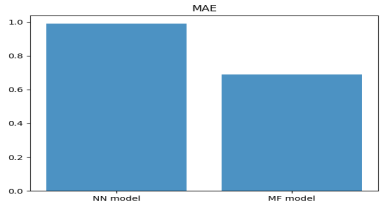


Figure 16: MAE



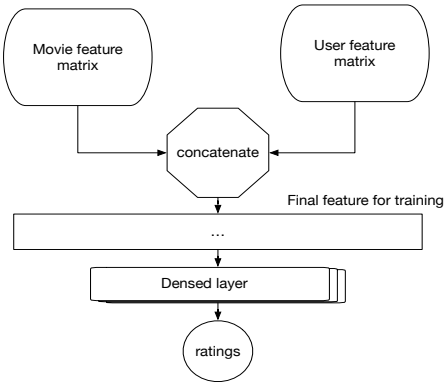
network model which contains CNN and fully connected-layers. A matrix factorization model which make full use of PySpark technique. For the rating prediction part, MF model has a overall better performance than the neural network model. For the item recommendation part, Neural Network model has a higher F-measure score compared to MF model but MF model has higher NDCG score.

7.1 Alternative rating prediction for NN model

The final output of the neural network model is a user representation vector and a movie representation vector, we set the rating to be the multiplication of two vectors, and we trained the model based on this.

An alternative method is to abandon multiplication and add a new densed layer and the output from this layer is rating[17]:

Figure 17: Alternative NN



7.2 Optimize Item Recommendation

7.2.1 Independent model. We are using sorted ratings from prediction model for item recommendation, similarity-based algorithm could be used like k-NN.

7.2.2 Running time. We are finding an algorithmic solution for less computing time, for the Matrix Factorization model, when it comes to a larger data set like MovieLens 100M, the user-item matrix is too large and it is accordingly very sparse. A possible solution is to transform matrix into Recursive Bordered Block Diagonal Form (RBBDF)[12], since each block matrix is solvable, we reduced the problem size.

ACKNOWLEDGEMENT

We would like to thank Prof.Yongfeng Zhang and TAs (Hanxiong Chen, Yunqi Li) for their guidance and advice on this project.

REFERENCES

[1] 2017. (2017). https://github.com/youhusky/Movie_Recommendation_System
[2] 2018. (2018). https://github.com/AgFeather/Movie_Recommendation_System
[3] 2019. (2019). https://github.com/microsoft/recommenders
[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). http://tensorflow.org/ Software available from tensorflow.org.
[5] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* 12 (July 2011), 2121–2159. http://dl.acm.org/citation.cfm?id=1953048.2021068
[6] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. MyMediaLite: A Free Recommender System Library. *RecSys* (2011).
[7] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages. DOI: http://dx.doi.org/10.1145/2827872
[8] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (08 2014). DOI: http://dx.doi.org/10.3115/v1/D14-1181
[9] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.
[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[11] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. 2016. Apache Spark: A Unified Engine for Big Data Processing. *Commun. ACM* 59, 11 (Oct. 2016), 56–65. DOI: http://dx.doi.org/10.1145/2934664
[12] Yongfeng Zhang, Min Zhang, Yiqun Liu, Shaoping Ma, and Shi Feng. 2013. Localized Matrix Factorization for Recommendation Based on Matrix Block Diagonal Forms. In *Proceedings of the 22Nd International Conference on World Wide Web (WWW '13)*. ACM, New York, NY, USA, 1511–1520. DOI: http://dx.doi.org/10.1145/2488388.2488520