BookCeption: A Proposed Framework for an Artificially Intelligent Recommendation Platform

Aniqa Zaida Khanom[1], Sheikh Mastura Farzana[1], Tahsinur Rahman[1] and Dr. Iftekharul Mobin[1]

[1]Department of Computer Science and Engineering, BRAC University, Dhaka, Bangladesh.

Email: {azkhanom, mastura.farzana, tahsinurrahman5} @gmail.com, iftekharul.mobin@bracu.ac.bd

ABSTRACT

BookCeption is a web-based book recommendation system that allows readers to browse and buy books across multiple platforms at the most affordable price. It fetches data from E-commerce sites like Barnes and Nobles, Amazon, eBay along with Facebook pages and retail bookstore websites. It is a unique book recommender that uses Machine Learning techniques to recommend books as well as offers from other platforms to the registered users. This paper introduces the architecture of the proposed framework, which integrates the book recommendation system with a platform for buying books. For the book recommender discussed here, four different recommendation techniques were used- SVD, Co-clustering, NMF and Deep Learning to determine the best one for the framework. In terms of computational training time, Co-clustering works best with a time of 5.21 minutes and RMSE value of 0.868. However, on the basis of RMSE value, Deep Learning Embedding Model works best with RMSE value of 0.7599 in 8.06 minutes when the number of epochs and the batch size are 20 and 200 respectively.

CCS Concepts

• Information systems → Recommender system • Information systems → Collaborative Filtering

Keywords

Book Recommendation; Co-clustering; Embedded Layers; Collaborative Filtering; Deep Learning.

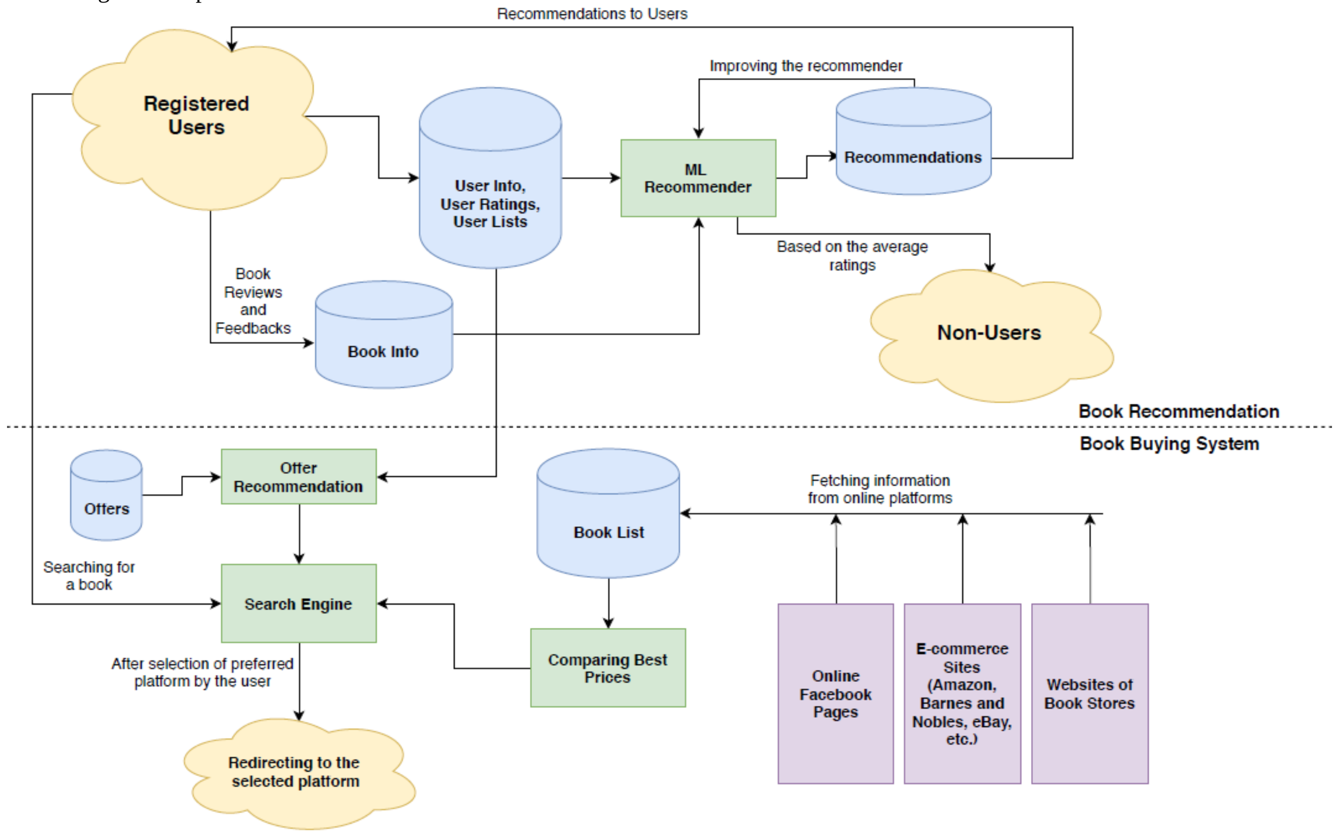
1. INTRODUCTION

With the ever increasing amount of information, people are leaning towards systems that can provide them with personalized suggestions. These suggestions also have to be relevant and delivered in minimal time. Similarly, users nowadays are more invested towards online shopping and as a result, applications of recommendation systems are more visible in E-commerce sites. Famous E-commerce chains such as Amazon, Flipkart, Snapdeal, eBay etc. are using various techniques like collaborative filtering and content-based filtering for product recommendations [1]. Netflix the media-services provider, also uses a variety of such algorithms and techniques to suggest content to its users [2]. Although personalized recommendations have become quite common in many online commercial sites, websites solely dedicated for selling books have not yet adapted this technique on a large scale. In this paper, a system has been introduced for the purpose of recommending books to readers using machine learning techniques along with an online bookselling platform. The primary motivation behind the development of this platform is to reduce customer agitation. Detailed information of the books are fetched from E-commerce sites, online bookstores and social media pages and then displayed to customers in the user interface of the platform. Additionally, this application suggests offers from other platforms which might be preferred by a user based on their data. This platform uses average ratings of individual books to provide recommendations to non-registered users. On the other hand, for a new user, memory-based collaborative filtering technique using Pearson Correlation coefficient is applied to make suggestions. Further, for existing users, model-based collaborative filtering methods are used. Singular value decomposition (SVD), Non-Negative Matrix Factorization (NMF) with Stochastic Gradient Descent and Co-clustering are being applied to predict books that might interest existing users. Additionally, Multi-layered Neural Network based on Embedding Layers has also been incorporated in the system. The purpose of applying multiple recommendation techniques was to find the best approach for this platform. Furthermore, the platform can display personalized offers to customers based on their purchase history along with product price comparison. The rest of the paper discusses the related work done in this field, proposed model, experimental recommendation system, results, model evaluation, discussion and conclusion.

ICSCA '19, February 19–21, 2019, Penang, Malaysia
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6573-4/19/02…$15.00
https://doi.org/10.1145/3316615.3316721

2. RELATED WORK

Collaborative filtering and content-based filtering are among the most popular methods for developing recommendation systems [3]. However, one of the major disadvantages of the second method is the mismatch between item and user profile terms which leads to low accuracy [4]. Collaborative Filtering has been further classified into model-based and content-based approaches. Generally, Pearson Correlation coefficient is used as the basis of weights while applying memory-based collaborative filtering. Matrix factorization algorithms like SVD, Probabilistic Matrix Factorization and Non-Negative Matrix Factorization fall under model-based collaborative filtering along with Clustering and Deep Learning [5][6]. However, deep learning has become a popular collaborative filtering approach in the development of recommender systems in recent times [7][8]. Additionally, collaborative filtering is being applied by popular online e-commerce sites for displaying products that might interest users by analyzing similar user profiles in addition to item profiles [5]. One such e-commerce site Amazon.com, has updated and improved the basic item-based collaborative filtering algorithm to recommend products to its users [1]. Popular book recommendation platform Goodreads also uses similar methods. Several frameworks have been proposed that recommends TV shows and movies to particular users [9][10]. Furthermore, such frameworks have also been introduced to the recommendation of books [11]. However, there is no dedicated platform for bookselling that provides automatic recommendations to customers based on their preferences as well as their past book buying and rating history.



Recommendations to Users
Registered Users
User Info, User Ratings, User Lists
ML Recommender
Improving the recommender
Recommendations
Based on the average ratings
Non-Users
Book Reviews and Feedbacks
Book Info
Book Recommendation
Book Buying System
Offers
Offer Recommendation
Book List
Fetching information from online platforms
Searching for a book
Search Engine
Comparing Best Prices
Online Facebook Pages
E-commerce Sites (Amazon, Barnes and Nobles, eBay, etc.)
Websites of Book Stores
After selection of preferred platform by the user
Redirecting to the selected platform

Figure 1. BookCeption Architecture.

3. PROPOSED MODEL

This paper proposes a platform that incorporates the book recommendation and book buying system into a web application. The primary features of the platform includes-

- Book Recommendation System and
- Book Buying System.

The book recommender is built using Collaborative Filtering by using Matrix Factorization and Deep Learning. Furthermore, Machine Learning techniques are also used in the book buying system, to select offers for specific users based on their preferences. Figure 1 represents the overview of the architecture of the proposed system, BookCeption. The upper half shows the Book Recommendation System and the lower half demonstrates the Book Buying System. The database contains six data tables;

- User Info (General information of the users),
- User Ratings (Rating of books given by the users out of 5),
- User Lists (List of books added as read or to be-read),
- Book Info (General Information of the specific books),
- Book List (List of books available in each platforms including their prices) and
- Offers (Collection of available offers for the users based on the user's data).

3.1 Book Recommendation System

The book recommender works in three variations- i) for non-registered users, the recommendation is based on average ratings of the books, suggesting the most popular books rated by most users, ii) for new users with no rating or lists, Pearson Correlation coefficient is used where the system takes a book name as input from the user and suggests similar books and iii) for users with enough data, collaborative filtering is implemented. The recommendations are saved in the database and are fed to the recommender so it does not recommend the same book twice. This makes the whole system dynamic and will give better recommendations when more data is collected. Ideally, the app will run the recommender every few hours. The pseudo code of the book recommender is as below:

```
BookRecommendation():
   If (UserLogin):
       FetchData (User);
       If (UserRatings && ListOfBooks == 0):
           DisplayBooks(PearsonCorrelationCoefficient());
       Else
           DisplayBooks(CollaborativeFiltering());
           SaveRecommendation();
           If UserSelection(book):
               BookBuying(book);
   Else
       DisplayBooks(AverageRatings());
       Redirect(Login/Signup);
```

3.2 Book Buying System

The book buying system allows unregistered users to browse and search for books, while it recommends offers to signed-in users only. The algorithm of the book buying system is as follows:

```
BookBuying(book):
    DisplayToUser(ComparePrices(book));
    SuggestOffers(book);
    If UserSelectsPlatform():
       Redirect(platform);
```

When a user searches for a book, the search engine fetches the list of books from the book list table, and compares the best prices. Information from different online platforms is crawled and updated every five hours, then stored in the "book list" data table. Additionally, the offer recommendation is displayed to the users from the available offers in the crawled data. Once the user selects their preferred platform from where they want to buy the book, the page redirects to the selected platform. The data crawling method works as follows:

```
DataCrawling():
    t=0;
    While(1):
       If (t=5):
          FetchData(platforms);
          SaveData(prices,offers);
          t=0;
       Else
          ++t;
```

3.3 Offer Recommendation

In the book buying system, there is an additional feature that recommends offers to users based on their preferences from the pool of available offers. The offers are crawled and fetched from the online platforms and saved in the database. When a user searches a book for buying, the best offers are recommended to the users based on their previous books read, ratings given, age and location of user. This is implemented using Deep Learning in Keras. The Neural Network consisted of two hidden layers and one output layer. The two hidden layers consisted of 100 nodes each with an activation function of Rectified Linear Unit while the output layer had one node and used the sigmoid activation function. The model's loss function was binary cross-entropy and the optimizer was Adam. It achieved a test accuracy of 90 percent when it was executed for 10 epochs with the batch size of each gradient update being 100.

3.4 User Interface

The interface of the website is developed using Flask. It is a web service development framework in Python that comes along with an inbuilt web server. This is done by serializing the model and converting it into an API. The homepage of the website includes a search box, user profile options and also displays top recommended books for the current user. Search result returns top listed books along with rating, price and the vendor of the book. Once a user selects specific item from this list, they are redirected to the platform which is providing the item. Figure 2 represents the web-based user interface of the framework.

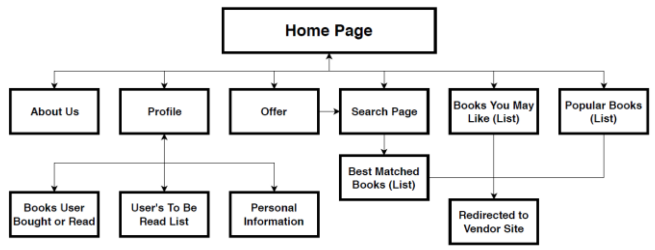


Figure 2. User Interface.

4. EXPERIMENTAL MODEL

4.1 Dataset Description

The experimental model uses two datasets as dummy data for the application. The first dataset, Goodbooks-10k, contains almost 6 million ratings of the 10,000 most popular books from Goodreads. This dataset was used to construct the book recommendation system. However, since there is no user information in this dataset, a second dataset, Book-Crossing Dataset, is used that contains demographic user information of users including 1.1 million ratings of 270,000 books and is used for offer recommendation.

4.2 Matrix Factorization

Several collaborative filtering algorithms were considered and the performance was compared using root mean squared error (RMSE) and computational time for each algorithm. One of the most common algorithm for recommendation systems is Singular Value Decomposition (SVD) [12]. One major issue with the SVD algorithm is that it if a user did not rate a particular book, that missing value is considered to be 0. Since most books would have no ratings, it poses a problem for recommendation. The solution to this problem is the "The BellKor Solution" to the Netflix Grand Prize [13]. By running Gradient Descent on the error function, the ratings value of the matrix is consistently updated. After a certain number of iterations, the matrix starts to represent the correct values. In this paper, the modified SVD model proposed by Koren (2009) is implemented with two different sets of parameters, optimized using Stochastic Gradient Descent [13][14]. Initially, the number of latent factors considered are 100 while stochastic gradient descent is ran for 20 iterations. Afterwards, the number of factors are increased to 300 while reducing the number of iterations to 10. An alternative to these approaches is co-clustering, where the idea is to obtain user and book clusters or neighborhoods simultaneously. Using this co-clusters, ratings are generated based on the average of the co-clusters (user-book neighborhoods) which also considers the individual biases [15]. Allowing scalability is a major advantage in co-clustering since the algorithm dynamically changes according to incremental input of data. Another collaborative filtering algorithm used is Non-Negative Matrix Factorization (NMF) where the matrices are considered to be positive making it easier to infer information [16]. For each of the above mentioned algorithms, cross validation was applied using 10-fold cross validation.

4.3 Deep Learning

Alternatively, collaborative filtering can also be applied using deep learning to generate recommendations [17][18]. Zhang et al. (2018) discusses the various researches conducted in the field of deep learning based recommender systems [8]. In this paper, a neural network with an embedding layer was used for the prediction. Keras, a python based open source neural network library provides an Embedding Layer that is used for the collaborative filtering [18]. In total the neural network has 7 layers consisting of 2 input layers, 2 embedding layers, 2 flatten layers and a dot layer. The input layers takes input for the books and users and then each embedding layer separately creates embedding for the books and users. The embedding layer initializes with random weights assigned to all the books and plots them as points in the vector space accordingly. These embedding can be used for extracting information about the data. The embedding are then flattened using flatten layers before combining the embedding of books and users using a dot product with the help of the dot layer. Initially, the number of epochs used to train the model is 10, the batch size is 32 samples for each gradient update and the optimizer of the used for this is the Adam optimizer. Then while keeping the optimizer same, the number of epochs is increased to 20 and the batch size to 100 and 200.

5. RESULTS

Table 1 contains the RMSE value and the Time required for the matrix factorization algorithms - SVD, Co-Clustering and NMF respectively. SVD was implemented in two combinations where number of factors were 100 and 20, and the number of epochs were 20 and 10 respectively. Co-clustering was implemented with the number of clusters as 3 and the number of iterations as 20 while the number of factors was 15 and the number of epochs was 50 for NMF.

Table 1. Performance of Matrix Factorization Algorithms.

Algorithms	Parameters		RMSE Score	Time (mins)
SVD	n_factors=100	n_epochs=20	0.840	13.13
SVD	n_factors=300	n_epochs=10	0.8588	14.52
Co-Clustering	n_clusters=3	n_iters=20	0.868	5.21
NMF	n_factors=15	n_epochs=50	0.884	14.56

Alternatively, Deep Learning was implemented in four different combinations. Table 2 represents the RMSE value and the Time required for the recommender. The epoch sizes are 10 in the first two experiments and 20 in the second two experiments respectively, while the batch size was 32, 100, 100 and 200 respectively in the four cases.

Table 2. Performance of Deep Learning Embedding Model.

Number of Epochs	Batch Size	RMSE Score	Time (mins)
10	32	0.8143	27.22
10	100	0.8202	7.18
20	100	0.7591	15.41
20	200	0.7599	8.06

Figure 3 exemplifies the RMSE Scores in blue for Matrix Factorization and in red for Deep Learning. From Figure 3 it can be seen that SVD has the least RMSE score while NMF has the highest RMSE. Additionally, it can be observed that the range of RMSE score is small for different Matrix Factorization algorithms whereas the range is greater for Deep Learning with varying parameters. These were all implemented using version 3.6 of the Python programming language with the training performed on NVIDIA GeForce 940MX with a RAM of 12GB.

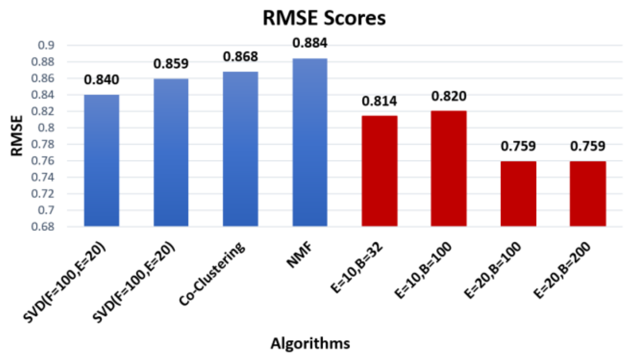


Figure 3. RMSE Scores for Matrix Factorization and Deep Learning.

6. MODEL EVALUATION

The python library Surprise was used in the implementation of the algorithms. SVD was implemented in two combinations of number of factors and epochs. For the default case, where number of factors and epochs are 100 and 20 respectively, the RMSE score is 0.84 and time taken to train is 13.13 minutes. However, for 300 factors and 10 epochs, the RMSE increases to 0.8588 and the training time increases to 14.52. So, it is worse in terms of both metrics. Conversely, when Co-clustering is implemented with number of clusters for users and books as 3 and 20 iterations, even though the RMSE score increases to 0.868, the training time almost decreases by one-third to 5.21 minutes. On the other hand, for NMF, the RMSE score is 0.884 and the time to train is 14.56 minutes, which are the highest among these four cases. Comparatively, the co-clustering model takes only 5.21 minutes to execute. So even though co-clustering has a greater error in terms of RMSE compared to the other algorithms, in this paper co-clustering is being considered as the best among the matrix factorization algorithms. An added advantage of Co-clustering it can handle dynamic update of data for prediction. If the results are observed closely, it can be seen that Deep Learning apparently produces better recommendation with lower RMSE scores for different combinations of epochs and batch size numbers. The RMSE score increases slightly to 0.8202 from 0.8143 when the batch size is increased from 32 to 100. However, the Training time decreases significantly, from 27.22 to 7.18 minutes respectively. On the other hand, if the epochs are increased to 20 for the same batch size of 100, the RMSE significantly decreases to 0.7591 but the time required is increased to 15.41 minutes again. In the last case, when the batch size is increased to 200, the RMSE increases slightly to 0.7599 but the training time becomes almost half, to 8.06 minutes. After comparing the different algorithms, the best result in terms of RMSE was given by the Deep Learning Embedding Model where number of epochs is 20 and batch size is 100 with an error of 0.7591. The rest of the algorithms register higher RMSE scores, deep learning with 20 epochs and batch size 200 being the closest with 0.7599. However, the Deep Learning Embedding Model of 20 epochs and a batch size of 100 is computationally much more expensive than other models with a runtime of 15.41 minutes.

7. DISCUSSION

This paper has explored various techniques to propose a book recommendation platform that can successfully give recommendations to users. Surprise, a python library, was used to implement SVD, NMF and Co-clustering, while Keras was used to run the Embedded Neural Network. For this particular dataset, it was seen that the RMSE scores did not differ much, while there were significant differences in the required training time. It is vital that the book recommender has a low RMSE value but also a low training time since the system needs to adapt to the choices of the users in order to run it frequently for better recommendations. Co-clustering uses an incremental partial update mechanism by assigning new entities to a global transitional cluster [7]. This allows incremental training of the recommender using new data which is a key feature of this framework. This feature along with a low computational time for training makes Co-clustering suitable for scalability. However, on the basis of RMSE value, Deep Learning Embedding Model with 20 epochs and a batch size of 200 works better with a RMSE value of 0.7599 and computational time of 8.06 minutes. Since this combination is the best, it has been used as the book recommendation algorithm.

8. CONCLUSION

In conclusion, to keep up with people's inclination towards e-commerce, an online based framework that is able to recommend books based on user preferences has been proposed in this paper. Different algorithms were implemented in order to determine the best combination for the book recommendation system. For the comparisons, RMSE and computational time was considered.

According to the results obtained, it can be said that collaborative filtering is an effective and efficient way to deduce predictions, making a system more accurate and dynamic at the same time. While traditional matrix factorization based collaborative filtering methods are still relevant, the focus is now shifting towards the usage of deep learning for collaborative filtering. This framework implements a similar deep learning based model. However, since deep learning presents a huge number of possibilities, some of these will be attempted in the future to improve the system further. Moreover, the platform will also be integrated into a mobile app. Overall, the paper proposes a framework for a web application that facilitates book reading and buying for users through recommendation of suitable books and corresponding offers.

9. REFERENCES

[1] Smith, B., & Linden, G. (2017). Two decades of recommender systems at Amazon. com. Ieee internet computing, 21(3), 12-18.

[2] Gomez-Uribe, C. A., & Hunt, N. (2016). The netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS), 6(4), 13.

[3] Mathew, P., Kuriakose, B., and Hegde, V. (2016, March). Book Recommendation System through content based and collaborative filtering method. In Data Mining and Advanced Computing (SAPIENCE), International Conference on (pp. 47-52). IEEE.

[4] Shoval, P., Maidel, V., and Shapira, B. (2008). An ontology-content based filtering method.

[5] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer,(8), 30-37.

[6] Grover, P. (2017, December 29). Various Implementations of Collaborative Filtering. Retrieved from https://towardsdatascience.com.

[7] Wang, H., Wang, N., and Yeung, D. Y. (2015, August). Collaborative deep learning for recommender systems. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1235-1244). ACM.

[8] Zhang, S., Yao, L., & Sun, A. (2017). Deep learning based recommender system: A survey and new perspectives. arXiv preprint arXiv:1707.07435.

[9] Hsu, S. H., Wen, M. H., Lin, H. C., Lee, C. C., & Lee, C. H. (2007, May). AIMED-A personalized TV recommendation system. In European Conference on Interactive Television (pp. 166-174). Springer, Berlin, Heidelberg.

[10] Vassiliou, C., Stamoulis, D., Martakos, D., & Athanassopoulos, S. (2006, April). A recommender system framework combining neural networks & collaborative filtering. In Proceedings of the 5th WSEAS international conference on Instrumentation, measurement, circuits and systems (pp. 285-290). World Scientific and Engineering Academy and Society (WSEAS).

[11] Goel, A., Khandelwal, D., Mundhra, J., & Tiwari, R. (2018). Intelligent and Integrated Book Recommendation and Best Price Identifier System Using Machine Learning. In Intelligent Engineering Informatics (pp. 397-412). Springer, Singapore.

[12] Polat, H., and Du, W. (2005, March). SVD-based collaborative filtering with privacy. In Proceedings of the 2005 ACM symposium on Applied computing (pp. 791-795). ACM.

[13] Koren, Y. (2009). The bellkor solution to the netflix grand prize. Netflix prize documentation, 81, 1-10.

[14] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010 (pp. 177-186). PhysicaVerlag HD.

[15] George, T., and Merugu, S. (2005, November). A scalable collaborative filtering framework based on co-clustering. In Data Mining, Fifth IEEE international conference on (pp. 4-pp). IEEE.

[16] Luo, X., Zhou, M., Xia, Y., and Zhu, Q. (2014). An efficient nonnegative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Transactions on Industrial Informatics, 10(2), 1273-1284.

[17] Karatzoglou, A., and Hidasi, B. (2017, August). Deep learning for recommender systems. In Proceedings of the Eleventh ACM Conference on Recommender Systems (pp. 396-397). ACM.

[18] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T. S. (2017, April). Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web (pp. 173-182). International World Wide Web Conferences Steering Committee.