CEUR-WS.org/Vol-2540/FAIR2019_paper_51.pdf

Using social networks to enhance a deep learning approach to solve the cold-start problem in recommender systems

Muhammad Shoaib Ikram[1][0000-0001-9034-6643], Anban Pillay[1][0000-0001-7160-6972] and Edgar Jembere[1][0000-0003-1776-1925]

[1] University of KwaZulu Natal, Westville, South Africa

Abstract. The Cold-Start problem refers to the initial sparsity of data available to Recommender Systems that leads to poor recommendations to users. This research compares a Deep Learning Approach, a Deep Learning Approach that makes use of social information and Matrix Factorization. The social information was used to form communities of users. The intuition behind this approach is that users within a given community are likely to have similar interests. A community detection algorithm was used to group users. Thereafter a deep learning model was trained on each community. The comparative models were evaluated on the Yelp Round 9 Academic Dataset. The dataset was pruned to consist only of users with at least 1 social link. The evaluation metrics used were Mean Squared Error (MSE) and Mean Absolute Error (MAE). The evaluation was carried out using 5-fold cross-validation. The results showed that the use of social information improved on the results achieved from the Deep Learning Approach, and grouping users into communities was advantageous. However, the Deep Learning Approach that made use of social information did not outperform SVD++, a state of the art approach for recommender systems. However, the new approach shows promise for improving Deep Learning models.

Keywords: Cold-Start, Deep Learning, Recommender Systems

1 Introduction

Recommender Systems based on collaborative filtering [11] or the hybrid approach [16] have been shown to have the cold start problem [15]. Even the best known collaborative filtering based solution to recommender systems, SVD++ [6], is susceptible to the cold start problem. The cold start problem arises when a Recommender System is not able to infer a recommendation for a user as it does not have enough information needed to make the recommendation. This may be due to the fact that a user is new to the system and have not yet rated good number of items, or it may be because an item has not yet been rated by a number of users. A recommender system is said to have the cold-start problem if the data sparsity is a least 85% [11]. The cold start problem can be viewed from three (3) perspectives as follows:

1. *Item cold start;* recommending a new item to an existing user.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)

2. *User cold start*; recommending an existing item to a new user.
3. *System cold start*; recommending a new item to a new user.

There have been a lot of research efforts to address the cold start problem in literature and these include; (i) integrating collaborative filtering and content-based filtering (ii) applying popularity based recommendation strategy (iii) real-time updating of user/item latent factors, and (iv) users manually suggesting what they may like.

Recently, there have been some efforts using social information to solve the cold start problem [15], [12], [3], [14], [8], [7]. These solutions usually take advantage of the availability of data on social relationships in social media platforms. Results from these efforts have shown that social information does improve the recommendation accuracy in the presence of the cold start problem [15], [12], [3], [14], [8], [7], [9]. These solutions are generally benchmarked against the solutions that would have been extended with the social information. However, none of the existing literature have compared their solution with SVD++.

Deep learning is also one of the techniques that have recently been used to solve the recommender systems problem in literature, with the hope of solving the cold start problem. There also has not been any work comparing deep learning solutions with SVD++. There have also been minimal work on combining the social media data and deep learning to solve the cold start problem.

This research aims to establish whether it is necessary to enhance deep learning solutions with social information and give insights on whether the recommender systems that use social information can outperform SVD++ in solving the cold start problem. This research therefore compares the performance of a community-based deep learning recommender system solution, a purely deep learning solution, and SVD++ in solving the cold start problem.

The remainder of this paper is organised as follows: Section 2 surveys existing literature of various existing solutions to the cold-start problems, Section 3 discusses the design and implementation of the proposed solution, Section 4 outlines the research methodology, Section 5 provides the results and Section 6 concludes this paper with a conclusion, discussion of results and future work.

2 Related Work

This section looks at related work that focused on the cold-start problem in recommender systems. The works of [15], [10], [17], [12], [7], and [3] made use of social information to solve the cold-start. They had been successful in proving that social information contributes to an improved solution to the cold-start problem. Datasets from Twitter and Facebook are amongst the popular social datasets that are used to solve the cold-start problem using social information. The social information from social media platforms have been used to better understand what items a user would be interested in the e-commerce domain [15] by making use of a matrix factorization to solve the item cold-start problem. Social information from one social media platform was used to recommend friends to a user on another social media platform that had no friends [12] by making use of a social graph to determine the users to recommend to a user with little to no friends. In collaborative filtering, social information is used to solve the cold start problem, by gathering information about the users and use it to infer their preferences based on the preferences of the users they have socials relationships with [7], [12]. Deep Learning solutions are now being widely adopted in recommender systems, however they had not been used to directly solve the cold start problem [15]. They have been widely used to form user and item embeddings. With the cold-start problem being an open problem that is constantly being worked on, the results achieved so far on using social information to solve the cold start problem, are proving to be better than the solutions they are based upon, but nothing much has been done on comparing the achieved results with state of the art recommender system solutions, such as SVD++..

3 Methodology

In order to compare the performance of a community-based deep learning recommender system solution, a purely deep learning solution, and the SVD++ in solving the cold start problem, this research adapted concepts from Matrix Factorisation to create a deep learning solution to recommender systems, and extended the created solution to develop a community based solution. For the SVD++ the Surprise python package [5] was used. Figure 1 shows the experimental setup that was used to compare the solutions

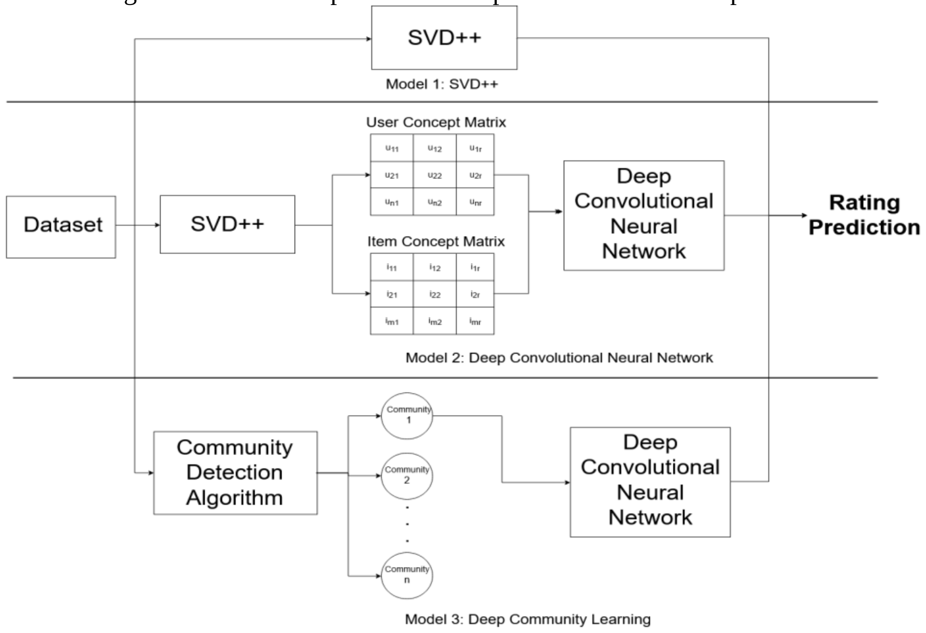
SVD++
Model 1: SVD++
Dataset
SVD++
User Concept Matrix
Item Concept Matrix
Deep Convolutional Neural Network
Rating Prediction
Model 2: Deep Convolutional Neural Network
Community Detection Algorithm
Community 1
Community 2
Community n
Deep Convolutional Neural Network
Model 3: Deep Community Learning

Fig. 1. Flow diagram of SVD++, Deep Conv and Deep Community Learning

The following subsections describe the SVD++, Deep Convolutional Neural Network and Deep Community Learning models and their implementations in detail.

3.1 SVD++

SVD++ [6], a matrix factorization technique, is a state of the art model for building recommender systems and rating prediction. SVD++ takes as input the set of users and the items rated and the output from the algorithm was a predicted rating for an item that a user did not rate. Equation (1) [6] shows the predicted rating as the summation of the global mean, user bias, item bias and the dot product of the transposed matrix q for item i, with the summation of matrix p for user u and user implicit feedback.

$$rating_{ui} = \mu + b_u + b_i + q_i^T \cdot \left(p_u + |N(u)|^{-\frac{1}{2}} \times \sum_{j \in N(u)} y_j \right) (1)$$

where, u is the target user and i is the target item, μ is the global mean of the training set, b_u and b_i are the bias factors. p_u is the user's latent factors, b_u is the user's bias factor, q_i^T is the item's latent factors, b_i item's bias factor, and $|N(u)|^{-\frac{1}{2}} \times \sum_{j \in N(u)} y_j$ is the user's implicit feedback.

The users' latent factors matrix is called the user-concept matrix and items' latent factors matrix is called the item-concept matrix.

3.2 Deep Convolutional Neural Network

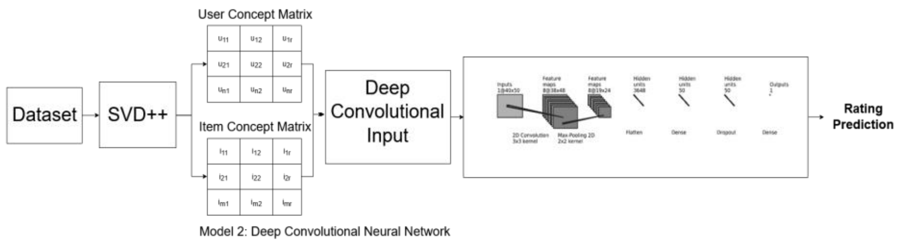
User Concept Matrix
u_{11} u_{12} u_{1r}
u_{21} u_{22} u_{2r}
u_{n1} u_{n2} u_{nr}
Item Concept Matrix
i_{11} i_{12} i_{1r}
i_{21} i_{22} i_{2r}
i_{m1} i_{m2} i_{mr}
Dataset
SVD++
Deep Convolutional Input
Inputs 1@40x50
Feature maps 8@38x48
Feature maps 8@19x24
Hidden units 3648
Hidden units 50
Hidden units 50
Outputs 1
2D Convolution 3x3 kernel
Max-Pooling 2D 2x2 kernel
Flatten
Dense
Dropout
Dense
Rating Prediction
Model 2: Deep Convolutional Neural Network

Fig. 2. Flow diagram of Deep Convolutional Neural Network

It was decided to make use of a deep convolutional neural network architecture over recurrent neural networks or multilayer perceptron, due to the ability of a convolutional neural networks to learn latent factors, and optimize towards the desired output.

At the inception of this research, [13] was the state of the art deep convolutional neural network model in rating prediction. The deep learning model proposed in [13] consisted of two parallel neural networks; one network received user reviews and the other item reviews, and provided a predicted rating based on the reviews written. This approach was not suitable for solving the cold-start problem, since it requires both the user and item reviews to provide a prediction for a given user item pair. Therefore, the model was modified to a single neural network and included a Dropout layer before the output layer to prevent overfitting.

Deep Convolutional Neural Network architecture used in this research, as shown in Fig. 2. The specification of the Deep Learning Architecture is as shown below.

- Input Layer: 2-Dimensional Convolutional Layer
  - Dimension n_r x n_c, where n_r number of rows and n_c number of columns
  - Kernel size is 3 with filters of 8
  - Activation Function: Rectified Linear Unit (ReLU)
- Max-Pooling Layer
- Flatten Layer
- Dense Layer returns vector of size 50
  - Activation Function: ReLU
- Dropout Layer with a dropout rate of 0.5 (50%)
- Output: Predicted rating scaled between 0 and 1,
  - Dense Layer returns vector of size 1
  - Activation Function: Sigmoid

Upon training SVD++ [6], the resultant user-concept and item-concept matrices were used to map a user and item to a vector representation of the respective user and item. SVD++ was trained in order to best predict what item a user would like best. The user-concept and item-concept matrices were used after training the model to map users and items. These two matrices have embedded the correlation between a user and item. It was shown by [6] that the more latent factors, the better the prediction accuracy but it comes at a cost of a longer computation time with diminishing improvements.

Key	
nf	Number of users
ni	Number of items
Lf	Number of latent factors

Table 1. Key

Algorithm 1 Deep Convolutional Input Matrix Construction

```
1:  INPUT: user-concept matrix, item-concept matrix
2:  STEPS:
3:  Set number of processors to use
4:  Split the dataset
5:      – Equally across the number of processors
6:      – Last processor gets the remaining portion of the dataset
7:  For each processor  iterate over their subset of the dataset:
8:        • Select target user's vector representation from user-concept matrix
9:        • Select target item's vector representation from item-concept matrix
10:           – Select the matching user's vector representation from user-concept
              interaction matrix
11:       • Select $n_i$ most similar items to target item
12:           – Select the matching item's vector representation from item-con-
              cept interaction matrix
13:       • If there are empty rows
14:           – The empty rows are filled with vectors of size Lf filled with 0's
15: OUTPUT: Matrix of dimensions $(2+n_f+n_i)$ x $(L_f)$
```

The Deep Convolution model input, Algorithm 1, consisted of vectors from the user-concept and item-concept matrices. The input was comprised of the target user's and item's vectors from the user-concept and item-concept matrices respectively. Thereafter, the next n_f rows were the users who were the most similar (Cosine Similarity) to the target user. They were ranked according to how close they were to 1 and the top n_f user vectors being used. Thereafter, the next n_i rows were the items that were the most similar (Cosine Similarity) to the target item. They were ranked according to how close they were to 1 and the top n_i item vectors being used. The benefit of using a Deep Convolutional Neural Network over traditional matrix factorization is that as new users or new items join the recommender system the matrix dimension for each of the matrices for the matrix factorization algorithm would need to be increased to cater for more users and items. A Deep Convolution matrices dimension would not need to be increased, however the vector representation for users and items would need to be updated as a user rates items.

3.3 Deep Community Learning

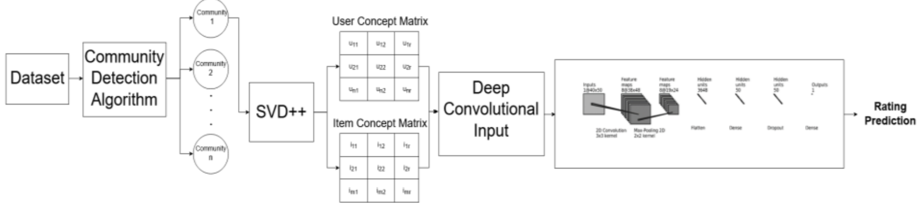

Fig. 3. Flow diagram of Deep Community Learning

The Deep Community Learning model consisted of two main components, community detection and the deep learning components as shown in Fig. 3.

The community aspect of the proposed solution made use of friendship information to form a relationship between users called a social graph. Thereafter, Multi-Level [2] community detection algorithm extracted communities/social networks from the social graph. Once communities were formed, a deep learning network was created for each community. A community's information was passed to an algorithm that formed a link between each community and their deep learning model.

The two main aspects, community and deep learning, needed to be linked through some algorithm so that the deep learning aspect could make use of the community information in predicting a rating for a given user item pair. It was decided to map each user and item to a vector that best represents the user and item within a community. SVD++ was ran for each community to get a user-concept and an item-concept matrix based on the community information. The user-concept matrix consisted of users in the given community and the item-concept matrix consisted of items found in the recommender system. This information for each community was passed to the deep learning component.

The deep learning model used was a Deep Convolutional Neural Network. It received as input a data matrix that consisted of the target user vector, target item vector, most similar users' vectors, and most similar items' This information was used to predict the rating for the target user-item pair. A challenge was that the number of users per community varied and could be below n_f which led to the input matrix for the convolutional network having empty rows. This was solved by setting the empty rows as vectors of zeros since a vector of zeros hold no significance as input, they served to represent empty rows or padding to ensure that the input matrix had the correct dimension for the network.

4 Experiments

This section discusses how the experiments were carried out. It gives details on the dataset used, and how it was pre-processed to showcase the cold-start problem existence. It also discusses how the models were trained and evaluation metrics used.

4.1 Dataset

The Yelp Academic dataset Round 9[1] was used. The dataset includes reviews of businesses and restaurants. The reviews were conducted by users who provide a review and rating for a business and/or restaurant that they have searched for on the Yelp website. Since this research focuses on solving the cold-start problem, users who have 1 review

[1] https://www.yelp.com/dataset/challenge

in the dataset are not discarded as they are key to us ascertaining how well does each of the compared solution perform in solving the cold start problem. Table 2 shows the statistics of the dataset used in this research and the evidence of the cold-start problem.

Dataset Sparsity = (Number of missing ratings / Number of expected ratings) % (2)

Statistics	Original Dataset	Research Dataset
#User	1M	-
#Users with friends	441.4K	424.5K
#Ratings	4.1M	2.7M
#Items	144K	138.7K
Average Ratings per User	4.03	6.42
Sparsity of Ratings	99.9972%	99.9954%

Table 2. Statistics of original dataset and research dataset

4.2 Pre-processing

The dataset was pre-processed to remove information that was not used by this research such as user reviews and for the items the information about where the business (item) was located was removed. The reviews a user had written were removed since it meant a review would need to be written by a user first in order to predict how well a user would like an item and due to hardware limitations it was not used to aid in training the model to be able to better predict what rating a user would give for an unrated item. Future work would look at using the reviews written by users to reinforce the rating prediction and recommendation accuracy. Thereafter the dataset was split into 5 equal parts into order to carry out 5-fold cross-validation more efficiently. Users who did not have friends are removed. The remaining users who have friends were used to form a social communities. The Multi-Level community detection algorithm [2] was used to group users into communities. For Multi-Level community detection algorithm the python-igraph package [19] was used. The Multi-Level community detection algorithms works by making use of a two phase approach and iterating between the two phases until the modularity measure is maximised. The algorithm starts with each user being a community on its own. In the first phase an isolated user is moved to a neighbouring community. The second phase builds a new social network from the communities in the first phase. The communities which have less than 11 users were excluded from the dataset used in this research. Since this research focused on the user cold-start where the user profile exists but they have not rated any items. The community structure formed was kept as a static community structure for each of the folds.

The input for the deep learning model for the Deep Community Learning model and the Deep Convolution model had a relatively high computational time. The high computational time was due to the amount of checks involved in selecting the most similar users and items to the target user and item respectively. Therefore, the solution to this was to parallelize the input. This was possible by the CHPC [1]. Algorithm 1 shows the process of parallelizing the data.

The dataset for this research was formed by pruning users who had few friends (11), the cold-start problem still did exist. A recommender system has the cold-start problem if the sparsity of ratings is at least 85% [11]. The Social Dataset had a sparsity of ratings of 99.9954% using Equation (2). A lower data sparsity means that the cold-start problem is less evident, and there may be sufficient information to predict what a user would like with high accuracy. The information available about a user are their friends. The information available about a business (item) was the business. The information available about a review conducted was the user who conducted it, the business it was about and the rating given.

4.3 Evaluation

The dataset was split into 5 equal parts – 5 fold cross-validation, 10% of the training set was used as a validation set. The batch size used was 100, learning rate was 0.002, and number of epochs was 2. The RMSprop optimizer was used for the Deep Convolutional Neural Network based models as the optimizer [13].

The mean squared error (MSE) Equation (3) and mean absolute error (MAE) Equation (4) metrics were used to evaluate the performance of each of the models. MSE has the benefit of penalizing large errors, but has the drawback to greatly penalizing outliers. In this research the max error possible between the true and predicted value was 4, this prevented any extreme outliers and MSE from growing to be large because of any extreme outliers. MAE served as a second evaluation metric, since it does not penalizing the model by being far from the true value and it better describes the average error.

$$\text{MSE} = \frac{\Sigma\left(x_{true} - x_{prediction}\right)^2}{n} \tag{3}$$

$$\text{MAE} = \frac{\Sigma\left|x_{true} - x_{prediction}\right|}{n} \tag{4}$$

Experimentation was carried out on the research dataset with users that had friends listed being considered. To evaluate whether social networks are beneficial and improve on a deep learning approach. The proposed model was compared against a non-community-based Deep Convolutional Neural Network. The state of the art benchmark for the cold-start problem is the matrix factorization model SVD++. The comparison models made use of all the training data from the research dataset. The parameter values for n_f, $n_{i,}$ and L_f are 19, 19 and 50 respectively. The test set consisted of a user and item with their matching rating, and the community the user fell into. Each review was considered independent to the next review in the test set and as such it did not alter the community structure and change the modularity value. MSE and MAE were used to evaluate the testing error and used as a measure to evaluate how well the models being compared performed.

5 Results

This section presents the results that were obtained from the experiments carried out in this research and provides a discussion on the results gathered is provided as well as the shortcomings of the relevant models.

The comparison between Deep Community Learning and Deep Conv was to evaluate whether social information enhanced a deep learning recommender system in solving the cold-start problem. Thereafter, evaluate how well the Deep Community Learning performs against the state of the art recommender system model, SVD++, in solving the cold-start problem. As shown in Table 3 and Table 4, SVD++ on average outperforms Deep Community Learning and Deep Conv at solving the cold-start problem.

Error Model	User Cold-Start	Item Cold-Start	System Cold-Start	Average Cold-Start
Deep Community Learning	2.4015	2.3480	2.8569	2.4021
Deep Conv	2.4406	2.3291	2.8813	2.4361
SVD++	2.2067	2.2696	2.8388	2.2188

Table 3. MSE error for the different models across cold-start cases.

Error Model	User Cold-Start	Item Cold-Start	System Cold-Start	Average Cold-Start
Deep Community Learning	1.3263	1.2958	1.5051	1.3257
Deep Conv	1.3728	1.3206	1.5435	1.3704
SVD++	1.2697	1.2838	1.5113	1.2735

Table 4. MAE error for the different models across cold-start cases.

The results in Table 3 and Table 4 show that, for the user cold-start problem, social information improved the deep learning model by achieving a lower MSE and MAE. SVD++ achieved a lower MSE than both models. For this cold-start problem the community-based solution performed better than a deep learning model without social information.

For the item cold-start problem, it is not clear if the use of social information improves the deep learning model, since the community-based solution achieved a lower MAE but higher MSE compared to deep learning model without social information. SVD++ was found to perform slightly better than the Deep Community Learning and Deep Conv solutions. For this cold-start problem social information does not provide evidence as to whether it improves a deep learning model.

For system cold-start problem, the use of social information improved the deep learning model. For MSE, SVD++ achieved better than Deep Community Learning, but for MAE Deep Community Learning achieved a lower error than SVD++. The system cold-start is the hardest problem to solve for since there is minimal information to no information available about users and items, and the use of social information to enhance a deep learning is proving to be useful.

6 Conclusion

The results showed that on average the Deep Community Learning enhanced the Deep Learning Approach, and grouping users into communities was advantageous. However, the approach did not outperform SVD++, a state of the art approach for recommender systems. The enhancement of social information shows promise for improving Deep Learning models at solving the cold-start problem. This finding is corroborated by other related work which include [9], [18].

Out of the 3 cold-start problems, the use of social information enhanced the deep learning approach for two of them. Social information achieved lower error values than deep learning at solving the cold-start problem. The shortfalls of the social information and deep learning models exist because deep learning models generally have a black box and pose the challenge of designing the ideal architecture to solve the recommender system problem and cold-start problem. Future work would look at blending communities so a user can fall into more than one community. SVD++ was used to map a user or item to a vector, future work would look at designing a deep learning model to map a user or item to a vector while forming a correlation between users and items.

References

1. Centre for high performance computing. https://www.chpc.ac.za/, last accessed 2019/10/01.
2. Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment, 2008(10):P10008, 2008.
3. Eduardo Castillejo, Aitor Almeida, and Diego López-De-Ipiña. Alleviating colduser start problem with users' social network data in recommendation systems. In Workshop on Preference Learning: Problems and Applications in AI (PL-12) at ECAI, pages 28–33, 2012.
4. Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pages 7–10. ACM, 2016.
5. Nicolas Hug. Surprise, a Python library for recommender systems. http://surpriselib.com, 2017.
6. Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 426–434. ACM, 2008.
7. Prijila Nair, Melody Moh, and Teng-Sheng Moh. Using social media presence for alleviating cold start problems in privacy protection. In 2016 International Conference on Collaboration Technologies and Systems (CTS), pages 11–17. IEEE, 2016.

8. Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, Lexing Xie, and Darius Braziunas. Low-rank linear cold-start recommendation from social data. In Thirty-First AAAI Conference on Artificial Intelligence, 2017.
9. Zhoubao Sun, Lixin Han, Wenliang Huang, Xueting Wang, Xiaoqin Zeng, Min Wang, and Hong Yan. Recommender systems based on social networks. Journal of Systems and Software, 99:109–119, 2015.
10. S Vairachilai, MK Kavithadevi, andMRaja. Alleviating the cold start problem in recommender systems based on modularity maximization community detection algorithm. Circuits and Systems, 7(08):1268, 2016.
11. 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.
12. Ming Yan, Jitao Sang, Tao Mei, and Changsheng Xu. Friend transfer: cold-start friend recommendation with cross-platform transfer learning of social knowledge. In Multimedia and Expo (ICME), 2013 IEEE International Conference on, pages 1–6. IEEE, 2013.
13. Lei Zheng, Vahid Noroozi, and Philip S Yu. Joint deep modeling of users and items using reviews for recommendation. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pages 425–434. ACM, 2017.
14. Ankush Bhatia. Community detection for cold start problem in personalization: Community detection is large social network graphs based on users' structural similarities and their attribute similarities. In 2016 IEEE International Conference on Computer and Information Technology (CIT), pages 167–171. IEEE, 2016.
15. Wayne Xin Zhao, Sui Li, Yulan He, Edward Y Chang, Ji-Rong Wen, and Xiaoming Li. Connecting social media to e-commerce: cold-start product recommendation using microblogging information. IEEE Transactions on Knowledge and Data Engineering, 28(5):1147–1159, 2016.
16. R Devika and V Subramaniyaswamy. A novel model for hospital recommender system using hybrid filtering and big data techniques. In 2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2018 2nd International Conference on, pages 267–271. IEEE, 2018.
17. Furong Peng, Jianfeng Lu, Yongli Wang, Richard Yi-Da Xu, Chao Ma, and Jingyu Yang. N-dimensional markov random field prior for cold-start recommendation. Neurocomputing, 191:187–199, 2016.
18. Leschek Homann, Denis Mayr Lima Martins, Gottfried Vossen, and Karsten Kraume. Enhancing traditional recommender systems via social communities. Vietnam Journal of Computer Science, 6(01):3–16, 2019.
19. Igraph – The network analysis package. https://igraph.org, last accessed 2018/05/31.