Trust-Based Neural Collaborative Filtering

Yejia Zeng$^{1,a}$ and Zehui Qu$^{1,b}$

$^1$School of Computer and Information Science, Southwest University, Chongqing, China

$^a$zengyejia@163.com, $^b$quzehui@swu.edu.cn

Abstract. In order to maintain the strong representational learning of the deep neural network model to learn the interaction function between any users and items, combining the trust relationship between users as a local relationship to enhance the ability to supplement interactive data, to achieve a better recommendation effect. This paper proposes a Trust-based Neural Collaborative Filtering model (TNCF). Firstly, trust information and scoring information are merged through the Generalized Matrix Factorization model (GMF) to obtain recommendations based on trust friend preferences. Then, using the Multi-Layer Perceptron model (MLP), the nonlinear kernel is utilized to learn the interaction function from the data to obtain the recommendation based on the user's personal taste. Finally, all the interaction results are aggregated for implicit prediction. Compared with the three different baselines on the FilmTrust and Epinions datasets, the experimental consequences reveal that the proposed model improves the recommendation and also performs well on sparse statistics.

1. Introduction

Because of the strong representational learning ability of deep learning, deep learning has received more and more attention from the academic and industrial circles in the recommendation system. There are many works on the recommendation system based on deep learning, for example, Salakhutdinov et al. [1] a limited Boltzmann machine was proposed, which is the first time that deep learning has been applied to the recommendation system. In 2016, Google [2] released wide & deep learning. In 2017, He Xiangnan et al. [3] proposed a neural collaborative filtering algorithm. They opened the door to the recommendation system in the field of deep learning. Most of the above algorithm models only consider the ratings between users and items. And then directly decompose the user and item interaction information based on the data: While data is exceedingly sparse, it is hard to find more associations among users and items [4].

In order to explore more interactive information for users to achieve better recommendation results, recommendations based on trust perception began to receive widespread attention. It is because the trust information between users not only reflects the similarity between users but also reflects other social factors. Some algorithms [5-8] only take advantage of the features of the social/trust network and ignore the user's own preferences in the collaborative filtering recommendation system. However, in the real world, the user ultimately decides what to buy based on the balance between his/her personal taste and the influential of his/her trusted friends. Work [9-11] is proposed for this sake. Although these works combine the user's own preferences with the preferences of his trusted friends, the parameters used to balance the tastes of the user and their trusted users are constant. Meanwhile, all these methods utilize trust data in shallow level and ignore the factor that trust relationships are very complicated.
Based on the above considerations, this paper proposes Trust-based Neural Collaborative Filtering (TNCF). We build a new neural collaborative filtering based on the user's rating data and the trust relationship between users. In this paper, we use deep learning to learn the latent factors of users and items, and to learn the latent factors of the influential of users’ trusted friends. Finally make a final recommendation between users’ personal taste and his/her trusted friends’ favors.

2. Trust-based neural collaborative filtering model
Inspired by neural collaborative filtering and recommendation based on trusted friends, this paper proposes a trust-based neural collaborative filtering (TNCF). TNCF model as is shown in figure 1, the bottom layer is the input layer. The input is the one-hot encoding of the ID of the user and his trusted friend, and the one-hot encoding of the ID of the same item they have consumed. The second layer is the embedding layer, which converts the sparse vector of the input layer into a dense vector, and obtains the latent factors' vector corresponding to the user (trust friend or item). The next layer is the entire model framework. The GMF on the left is based on the taste recommendation of trusted friends; the right side is based on the user's own taste recommendation, and the upper side is the fusion of two recommendation to get the user's final recommendation. The details are as follows.

![Figure 1. Overview of the architecture of TNCF.](image)

2.1. Encoding user-item pair and user-trustee information
User-item interaction information and user-to-user trust information as input will use the following encoding.

We use one-hot encoding to represent the ID of the user $u$ and the item $i$, i.e., $x_u \in \mathbb{R}^{m \times 1}$ and $y_i \in \mathbb{R}^{n \times 1}$. Through the embedding layer, we got the latent factors vector of user and item, as shown below.

$$p_u = P^T \cdot x_u, q_i = Q^T \cdot y_i,$$  \hspace{1cm} (1)

Where $P \in \mathbb{R}^{K \times m}$ and $Q \in \mathbb{R}^{K \times n}$ are the feature matrix for users and items respectively. $K$ denotes the number of latent factors.
Similarly, we use one-hot encoding to represent the ID of the trustor $a$ and trustee $b$, i.e., \( t_a \in \mathbb{R}^{m} \) and \( t_{ee_b} \in \mathbb{R}^{m} \). Similarly, \( C \in \mathbb{R}^{K \times m} \) and \( D \in \mathbb{R}^{K \times m} \) are the feature matrix for trustor and trustee respectively. Through the embedding layer, we got the latent factors vector of trustor and trustee, as shown below.

\[
c_a = C^T \cdot t_a, \quad \phi_{ee} = D^T \cdot t_{ee_b},
\]

(2)

2.2. TNCF Model

The integrating of user-item prediction and user-trustee prediction can be divided into two parts, one is a recommendation based entirely on reliable friends’ preferences, and the other one is a recommendation purely based on the user-item latent factors; Finally, the two parts are combined in the same way as the neural matrix factorization to form the final prediction of \( y_{ui} \).

2.2.1. Recommend by Trust friend. In the first part, as shown in [9], the score prediction based purely on the taste of a trusted friend is the product of the user-item rating matrix and the user-trust rating matrix. However, it is difficult for users to score the trust value of a trusted friend. This article only uses 1 to indicate that user \( a \) trusts user \( b \), and the influence of each trusted friend on the user is explored through deep learning. To put it another way, it is to explore the trust metric of each user \( a \) to user \( b \).

To effectively explore the trust metric of each user \( a \) to user \( b \), we used the GMF method proposed in [3], and then the latent factors of the trust metric function can be defined as below:

\[
\phi_1(c_a, d_b) = c_a \odot d_b,
\]

(3)

In the same way, we can get the latent factors that trust friend \( u_b \) for the rating of item \( i \). The function as below:

\[
\phi_2(p_{u_b} \cdot q_i^G) = p_{u_b} \odot q_i^G,
\]

(4)

Where \( q_i^G \) indicates the latent factors of item \( i \) in GMF. Note that \( d_b \) and \( p_{u_b} \) here both represent user \( u_b \), since the encoding result of the \( u_b \) as a trustee and as a user is inconsistent, it is represented by different symbols.

Thus, in this model, latent factors that are based entirely on the tastes of trusted friends can be represented by the following function:

\[
\phi_{trust}(\phi_1, \phi_2) = \phi_1 \odot \phi_2,
\]

(5)

2.2.2. Recommend by User’s Taste. In the second part, we use the way same as the first part to learn the interaction function in user \( u_a \) and item \( i \). The formula is expressed as follows:

\[
\phi_{rating} = a_L \left( W_L^T \left[ a_{L-1} \left( \cdots a_2 \left( W_2^T \left[ q_i^M \right] + b_2 \right) \cdots \right] + b_L \right) \right],
\]

(6)

Where \( q_i^M \) indicates the latent factors of item \( i \) in MLP. Where \( W_L, b_L, a_L \) represent the weight matrix, bias vector, activation function of the each hidden layer, \( L \)-th layer’s perceptron, respectively.

In our model, we choose ELU function to be the activation functions of MLP layers [12]. Due to its positive value, ELU can alleviate the problem of gradient disappearing like ReLU, lReLU, and preLU. Compared to ReLU, the ELU has a negative value, which can push the output unit of the active unit to 0, achieving the effect of batch normalization and reducing the amount of calculations.

2.2.3. Integrating user-item prediction with user-trustee prediction. In order to explore the interaction between trusting friend tastes latent factors and user-item rating latent factors, we use multi-layer
perceptron (Expressed by MLP2) to integrate these latent factors, i.e. as shown in the top of figure.1, we can have:

\[
\phi_{\text{result}} = a_L \left( W_L^T \left( a_{L-1} \left( \cdots a_2 \left( W_2^T \left[ \phi_{\text{rating}} \quad \phi_{\text{trust}} \right] + b_2 \right) \cdots \right) + b_L \right) \right),
\]

(7)

As for the output layer, \( \sigma(\cdot) \) is the sigmoid function defined as \( \sigma(x) = \left(1 + \exp(-x)\right)^{-1} \). The final prediction \( y_{u,i} \) is as follows

\[
y_{u,i} = \sigma(\phi_{\text{result}})
\]

(8)

We can get the user-item-trustee set \( T = \{ u, i, u, v \} \) through the user scoring matrix and the user trust matrix, and then we take 4 times negative sampling to get the final training set, i.e., \( X \leftarrow T^{+} \bigcup_{T_{\text{sampled}}}^{T_{\text{sampled}}} \), we use cross entropy loss as the optimization goal.

\[
L = -\sum_{x \in X} \left( y_{u,i} \cdot \log \hat{y}_{u,i} + (1 - y_{u,i}) \cdot \log \left(1 - \hat{y}_{u,i}\right)\right).
\]

(9)

We use stochastic gradient descent (SGD) to train the model. The hyper-parameters we set are as follows: the learning rate is set to 0.0001, the batch size is set to 512, the negative sampling multiple is 4, and the number of tests is set to 100. Note that the embedding size of MLP layers can be modified, as long as the size of the upper layer's output vectors has the same number of neurons of the next layer. In this experiment, the embedding size is set to 16 in the input layer, and to the first hidden layer 1 of MLP1 (as shown in Figure 1), the embedding size is set to 32, we use the tower mode, where the bottom layer is the widest and each successive layer has a smaller number of neurons, and the layer size of each successive higher layer is halved. Also, the number of MLP1 layers is set to 2; the number of MLP2 layers is set to 3.

3. Experiments

3.1. Experimental data set

To evaluate our approach with other algorithms, we utilize two real world datasets with both rating and trust data for comparison: FilmTrust and Epinions datasets. The statistics of these data sets are shown in the Table 1.

| Data Set   | Num of users | Num of items | Num of ratings | Num of trust |
|------------|--------------|--------------|----------------|--------------|
| FilmTrust  | 1508         | 2071         | 5,497          | 1853         |
| Epinions   | 49,290       | 139,738      | 664,824        | 487,181      |

FilmTrust: FilmTrust is a data set that was completely captured from the website FilmTrust in 2011 [13]. Through data preprocessing, we have complemented trust.txt (users trust their own data, i.e., add trustid=trusteeid=userid to all users). Then trust.txt contains 3196 data records.

Epinions: The dataset was from the Epinions.com Web site [6]. Due to a large number of users and items, the one-hot encoding of the user ID and item ID will be very large and sparse. Therefore, in order to save memory, in this experiment, the user ID whose data record is less than 20 and the item ID whose data record is less than 5 is filtered out. Finally, the remaining user IDs (item IDs) are mapped to consecutive IDs. The processing of trust information is the same as FilmTrust.
3.2. Evaluation protocols and indicators
Evaluation Protocols: Like other top-n recommendation algorithms[14-15], we adopted the leave-one-out evaluation. In order to saving time, we used the method same as [16] to randomly negative sample 100 items for each users, and rank the test items.
Evaluation indicators: The evaluation indicators is Hit Ratio (HR) [17] and Normalized Discounted Cumulative Gain (NDCG) [18]. For the top-K recommendation, HR is a commonly used measure of recall rate; NDCG considers the effect of each recommendation result at different locations on the overall recommendation effect. Both metrics value are between 0 and 1, the larger the better the recommendation.

3.3. Baselines
We compared our proposed TNCF methods with the following methods:
- **ItemPop**: ItemPop is a way to recommend popular items to users [19].
- **DMF**: Deep Matrix Factorization (DMF) is a new deep learning framework that uses matrixes including explicit ratings and non-preference implicit feedback as model inputs [20]. Both explicit and implicit feedback can be considered by designing a special cost function. We use the hyper-parameters that make the best experimental results. It is the most related work to us.
- **NCF**: This model only used the rating data. We tuned its hyper-parameters to make the best experimental results.

3.4. Experimental results
3.4.1. Comparison with Baseline. We present the results in Table 2. On the FlimTrust dataset, we can have the following observations: Our proposed TNCF performed best, followed by the DMF model and ItemPop model, and the NCF model performed the worst. FlimTrust data is dense and the amount of data is small. Four models have achieved good performance. The TNCF we presented is outstanding, especially in NDCG. Perhaps because of the special of FilmTrust data (people like to watch popular movies), so ItemPop performance well too.

| Model | Film Trust | Epinions |
|-------|------------|----------|
|       | HR@5 | HR@10 | NG@5 | NG@10 | HR@5 | HR@10 | NG@5 | NG@10 |
| ItemPop | 0.8786 | 0.9137 | 0.7902 | 0.8053 | 0.3086 | 0.3627 | 0.2032 | 0.2482 |
| DMF    | 0.8846 | **0.9204** | 0.7815 | 0.7931 | 0.3501 | 0.4583 | 0.2464 | 0.2849 |
| NCF    | 0.8726 | 0.9018 | 0.7194 | 0.7650 | 0.3252 | 0.3875 | 0.2384 | 0.2719 |
| TNCF   | **0.8971** | 0.9196 | **0.8057** | **0.8110** | **0.5042** | **0.5266** | **0.3374** | **0.3684** |

On the Epinions dataset, we can have the following observations: Compared with the FlimTrust data, the Epinions data is very sparse and the data volume is large. The performance of these four models is degraded. The TNCF model we proposed performance is relatively stable, which reflects the effectiveness of the model for sparse data.

3.4.2. Comparison with Baseline. We present the results in Table 2. On the FlimTrust dataset, we can have the following observations: Our proposed TNCF performed best, followed by the DMF model and ItemPop model, and the NCF model performed the worst. FlimTrust data is dense and the amount of data is small. Four models have achieved good performance. The TNCF we presented is outstanding, especially in NDCG. Perhaps because of the special of FilmTrust data (people like to watch popular movies), so ItemPop performance well too.

On the Epinions dataset, we can have the following observations: Compared with the FlimTrust data, the Epinions data is very sparse and the data volume is large. The performance of these four models is degraded. The TNCF model we proposed performance is relatively stable, which reflects the effectiveness of the model for sparse data.

We present the performance of HR@10 and NDCG@10 on the different number of latent factors in Figure.2 and Figure.3, respectively. As shown in the picture, the NCF model is the model most affected by the number of hidden factors. The DMF model and the TNCF model perform better and more stable. The performance of the TNCF model with a hidden factor of 8 has exceeded the NCF model with a hidden factor of 64. The number of hidden factors is smaller, the memory required for the calculation is smaller. For the same recommended effect, it is clear that the memory space cost of the TNCF model is less. When latent factors number is 64, the HR@10 of DMF model is little better than TNCF model, but the NDCG @ 10 is much lower than the TNCF model.
3.4.2. The necessity of Nonlinear Combination. As showed in Figure 1, the MLP2 non-linear approach employed at the top of the model combines recommendations based on trusting friend tastes with recommendations based on the user's personal taste. In fact, linear combination can also be achieved, but why is a nonlinear combination used here? There are two reasons for this: MLP2 makes the network deeper, meanwhile, the recommended performance better; it has been found through experiments that MLP2 is more time-saving than linear combination.

4. Conclusion
This paper proposes a neural network model (TNCF) that combines user ratings and user trust information for a better recommendation. Experiments with real-world data sets demonstrate the effectiveness of our model and trust-based for implicit recommendation tasks. Compared with the baseline, it is verified that the TNCF model has a better recommendation performance. However, the TNCF model still has the disadvantage of being less explanatory. Meanwhile, we ignore the dynamics of the user's trust information. In the future, we will add the dynamics of trust information in our future work, and combine implicit feedback and explicit ratings to build trust-based recommendations.

Acknowledgments
Thanks to all the workers who provided the data set, without the hard work of them, this experiment will be difficult. At the same time, I would like to thank the reviewers for their careful review.

Reference
[1] R Salakhutdinov, A Mnih, G Hinton 2007 Restricted Boltzmann machines for collaborative filtering. Proc. of 24th ICML’07 791-798.
[2] H-T Cheng et al. 2016 Wide & deep learning for recommender systems. Proc. of DLRS 2016 7-10.
[3] X He, L Liao, H Zhang, L Nie, X Hu, and T-S Chua 2017 Neural Collaborative Filtering. Proc. of WWW’17 173–182.
[4] T Bai, J R Wen, J Zhang, Wayne X Zhao 2017 A Neural Collaborative Filtering Model with Interaction-based Neighborhood. Proc. of CIKM ’17 1979-1982.
[5] R Bhatt, V Chaoji and R Parekh 2010 Predicting product adoption in large-scale social networks. Proc. of 19th ACM CIKM’10 1039–1048.
[6] P Massa and P Avesani 2007 Trust-aware recommender systems. Proc. of RecSys ’07 17-24.
[7] L. Jiang, Y. Cheng, L. Yang, J. Li, H. Yan, X. Wang 2018 A trust-based collaborative filtering algorithm for E-commerce recommendation system Proc.of Journal of Ambient Intelligence and Humanized Computing. 10.1007/s12652-018-0928-7.
[8] N. Lathia, S. Hailes, and L. Capra 2008 Trust-Based Collaborative Filtering. Trust Management II. IFIP 2008. IFIP 263 119-134.

[9] I Konstas, V Stathopoulos, JM Jose 2009 On social networks and collaborative recommendation. Proc. of 32nd ACM SIGIR’09 195–202.

[10] J Tang, H Gao and H Liu 2012 mTrust: discerning multi-faceted trust in a connected world. Proc. of 5th ACM WSDM 93–102.

[11] H Ma, I King, and M R Lyu 2011 Learning to recommend with explicit and implicit social relations. ACM Trans. TIST. 2 29.

[12] C Djork-Arné, U Thomas and H Sepp 2015. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). Proc. of ICLR2016 arXiv:1511.07289

[13] G Guo, J Zhang and N Yorke-Smith 2013 A Novel Bayesian Similarity Measure for Recommender Systems. Proc. of 23rd IJCAI’13 2619-2625.

[14] X He, M -Y Kan, P Xie, and X Chen 2014 Comment-based multi-view clustering of web 2.0 items. Proc. of 23rd WWW’14 771-782.

[15] I Bayer, X He, B Kanagal, and S Rendle 2017 A generic coordinate descent framework for learning from implicit feedback. Proc. of 26th In WWW 1341-1350.

[16] A M Elkahky, Y Song, and X He 2015 A multi-view deep learning approach for cross domain user modeling in recommendation systems. Proc. of 24th WWW’15 278-288.

[17] JUNMAIL 2017 hit ratio. [Online]Available: https://blog.csdn.net/junmail/article/details/4792085.

[18] X He, T Chen, M -Y Kan, and X Chen 2015 TriRank: Review-aware explainable recommendation by modeling aspects. Proc. of 24th CIKM’15 1661-1670.

[19] S Rendle, Z Gantner, C Freudenthaler, and L Schmidt-Thieme 2011 Fast context-aware recommendations with factorization machines. Proc. of the 34th ACM SIGIR 635-644.

[20] H J Xue, X Y Dai, J B Zhang, S J Huang and J J Chen 2017 Deep Matrix Factorization Models for Recommender Systems. Proc. of 26th IJCAI. 3203-3209.