KNCR: Knowledge-Aware Neural Collaborative Ranking for Recommender Systems

Chen Huang*, Zhongyuan Gan*, Feng Ye†, Pan Wang* and Moxuan Zhang‡
*School of Modern Post, Nanjing University of Posts & Telecommunications, Nanjing, China
†Department of Electrical & Computer Engineering, University of Dayton, Dayton, OH, USA
‡Schools of International Education, Jinling Institute of Technology, Nanjing, China

***Abstract*—As a new type of auxiliary information, The knowledge graph has attracted much attention from researchers in recent years. Knowledge graphs contain rich semantic associations with entities, which provides a potential source of auxiliary information about recommendation systems, as well as the capability of dealing with data sparsity and cold start to improve recommendation. In this paper, we propose an enhanced collaborative filtering recommendation algorithm based on implicit feedback and representation learning of the knowledge graph combined with neural network (KNCR). The proposed algorithm combines the existing semantic data extracted from the public knowledge base with the recommending item. KNCR can bridge intrinsic relationship between items that are not considered by the traditional collaborative filtering algorithm, enhance the effect of the matrix decomposition method of the knowledge level, and effectively solve the problems of sparse scoring matrix and cold start. The experimental results from the real-world public dataset demonstrate that KNCR can improve the performance of personalized recommendations.**

I. Introduction

Collaborative filtering is the most widely used recommendation technology up to now. Literature [1] proposed the matrix decomposition for collaborative filtering in 2009, which maps users and projects to Shared latent space, and uses latent features to represent users or projects. Although MF is effective for collaborative filtering, as the inner product in matrix decomposition is realized by simply multiplying the potential features of linear combination, it is difficult to extract the inherent relationship between users and projects in complex interactions.

In this paper, we propose to ...... (briefly summarize the contribution in this work that will address the challenge mentioned above.)

xxxx has been studied [ABCD]). (Here you shall rewrite the 'related work' by introducing their contribution, drawback, as well as your solution in this work.)

Literature [2] improved the idea of connecting potential vectors with MLP, proposed an NCF framework that uses neural networks to learn functions of data and effectively capture nonlinear interactions between user factors and project factors, including the integration of multilayer perceptron and generalized matrix decomposition components. NCF uses the learned interaction function in- stead of the inner product to get a good result. Literature [3] proposed a personalized pin-ordering model (NPR) based on Bayesian personalized ordering (BPR) and the latest development of neural network based on implicit feedback dataset. Literature [4] proposed a Neural Collaborative Ranking framework (NCR) with new classification strategy based on the widely used pair-sorting hypothesis with the recently proposed NCF framework.

Existing researches showed that the knowledge graph means learning can embed entities and relationships in the knowledge graph in low-dimensional vector space, so as calculating semantic relations between entities and relationships in vector space. Introducing the knowledge graph into the recommendation system as auxiliary information can effectively solve the sparse and cold start problems existing in the traditional recommendation system. Compared with other kinds of auxiliary information, the first are that the knowledge graph introduces more semantic relationships, which can deeply to discover users' interests. Secondly, the knowledge graph provides different types of relation connections, which are conducive to the divergence of recommendation results and avoid the limitation of recommendation results from a single type. Thirdly, the knowledge graph connects users' historical records and recommendation results, thus improving users' satisfaction and acceptance of recommendation results and enhancing users' trust in the recommendation system [5]. Literature [6] proposed a hybrid recommender system termed as CKE, which integrates collaborative filtering and knowledge base for recommendation. Literature [7] proposed RippleNet, an end-to-end framework that naturally incorporates the knowledge graph into recommender systems. Literature [8] propose a deep knowledge-aware network (DKN) that incorporates knowledge graph representation to news recommendation. In this paper, we employ four knowledge graph representations learning methods: TransE [9], TransH [10], TransR [11], TransD [12]. To the best of our knowledge, this paper is the first work that proposes leveraging knowledge graph embedding in a novel general neural network-based collaborative ranking framework for personalized ranking.

Literature [13] found that: if only based on user recommendation results from the project results, and can not accurately reflect the user's preference. some researchers have proposed the sorting technology integration into the recommendation algorithm [14]. In the sorting algorithm, according to the relevant order of learning, these methods are mainly divided into the point-wise method and the pair-wise method. The point-wise method usually takes user scores as classification tags or values, and tries to learn the relevant scores of missing data directly. Pair-wise, on the other hand, tries to capture the order of precedence betwween missing data, which usually improves ranking performance more than point-wise approach [15]. Reasons for us to choose pair-wise approach is to avoid the complexity of adjustment of parameters caused by excessive parameters and its predicting relative order is closer to the nature of ranking than predicting class label or relevance score.

In this paper, we make the following key contributions:

- For the first time, we constructed a middle layer to embed the high-dimensional vectors from representation learning of the knowledge graph to the neural collaborative filtering model, so that the neural network model can make full use of the abundance of prior knowledge in the knowledge graph to mines the deep relationship between items, as well as interaction information between the user and the item.
- Bayesian pairwise learning is implemented to solve the problem of multi-parameter estimation caused by high-dimensional features. Experiments show that the neural collaborative recommendation model with the embedding of knowledge graph based on ranking learning can improve the personalized recommendation effect.
- Through multiple groups of comparison and control experiments, it is proved that by introducing knowledge graph integrated with the scoring matrix based on implicit feedback which effectively solves the problem of sparse and cold starting of scoring matrix, can actually enhance the performance and accuracy of collaborative filtering recommendation.

The rest of this paper is organized as follows: In section II (User label and ref as given in this example, see LATEX source code.), we introduce the knowledge representation learning approach, Section 3 gives the overview of our framework in detail. Section 4 describes the details of our experiment, including datasets description, experimental setting, evaluation metrics, and experimental results. Follow by a brief conclusion of this paper and proposed the next work plan in Section 5.

II. Preliminaries

A. *Knowledge graph representation learning*

1) TransE~~:~~ ~~TransE~~ is a distributed vector representation of entities and relationships in the knowledge graph, which is vectorized to represent entities and relationships in the same vector space. For each triplet (h, r, t), the relationship r is interpreted as a translation vector, as the translation between head-to-tail entities h and r, transE wants when (h, r, t) is true there should be $\|h + r\| \approx t$.

$$f_r(h,t) = \|h + r - t\|_{1/2} \tag{1}$$

2) TransH~~:~~ ~~To distinguish the different effects of entities in different relationships,~~ TransH allows entities ~~to have different representations~~ when involved in different relations by projecting the entity embeddings into relation hyperplanes.

$$f_r(h,t) = \left\|\left(h - w_r^T h w_r\right) + r - \left(t - w_r^T t w_r\right)\right\|_2^2 \tag{2}$$

3) TransR~~:~~ ~~TransR~~ introduces a projection matrix Mr for each relation r to graph entity embeddings to the corresponding relation space. The score function in TransR is defined as:

$$f_r(h,t) = \|M_r h + r - M_r t\|_2^2 \tag{3}$$

4) TransD~~:~~ ~~TransD~~ replaces the projection matrix in TransR by the product of two projection vectors of an entity-relation pair. Where $M_r^h = r_p h_p^T + I^{m\times n}$, $M_r^t = r_p t_p^T + I^{m\times n}$, h_p,t_p,r_p are mapping vectors, and $I^{m\times n}$ is the identity matrix. The score function in TransD is defined as:

$$f_r(h,t) = \left\|M_r^h h + r - M_r^t t\right\|_2^2 \tag{4}$$

III. A Framework for Knowledge-Aware Neural Collaborative Ranking

A. *Overview of the Framework*

An overview of the proposed KNCR framework is shown in Fig. 1. The framework consists of *input-layer*, *knowledge-embedding-layer*, *embedding-layer*, *neural-network-training-layers* and *prediction-layer*, described in the following. The mapping and training process of KNCR are presented in Algorithm 1.

The-Input-Layer is at the bottom. U_{mlp}, U_{mf}, I_{mf_pos}, I_{mf_neg} will be fed into *input-layer*, which are the partial input of MLP and MF as well. U_{mlp} and U_{mf} are based on user number, while I_{mf_pos} and I_{mf_neg} are the item number from implicit feedback data, indicate the positive and negative feedback of users respectively. ~~We must declare that *input-layer* is just to show the specific input form of the model, and not participate in any calculations.~~

The-Knowledge-Embedding-Layer is also at the bottom. The input of *knowledge-embedding-layer* are the items from implicit feedback data. The output is the high-dimensional eigenvectors I'_{mlp_pos} and I'_{mlp_neg} which is different from I_{mf_pos} and I_{mf_neg}, learning by TransE, TransR, TransH, TransD after associated with the knowledge graph.

The-Embedding-Layer is above *input-layer* and *knowledge-embedding-layer*. U_{mlp}, U_{mf}, I_{mf_pos} and I_{mf_neg} are mapped to high-dimensional eigenvectors in the *embedding-layer* as U'_{mlp}, U'_{mf}, I'_{mf_pos} and I'_{mf_neg}. They can be thought of as the latent vector of user and item. Because I'_{mlp_pos} and I'_{mlp_neg} have been mapped to high-dimensional feature vectors in *knowledge-embedding-layer*, they do not participate in the *embedding-layer* operation.

The-Neural-Network-Training-Layers is above *embedding-layer*. Considering it unifies the strengths of linearity of MF and non-linearity of MLP for modelling the user–item latent structures, this layer is divided into two parts. We feed U'_{mf}, I'_{mf_pos}, I'_{mf_neg} into first part, which can also be called as MF part, then calculating according to the original matrix decomposition method:

$$V_{mf} = \begin{bmatrix} U'_{mf} \odot I'_{mf} \\ -U'_{mf} \odot I'_{mf\_neg} \end{bmatrix} \tag{5}$$

The second part, which can also be called as MLP part, will be trained with the input of U'_{mlp}, I'_{mlp_pos} and I'_{mlp_neg}. We

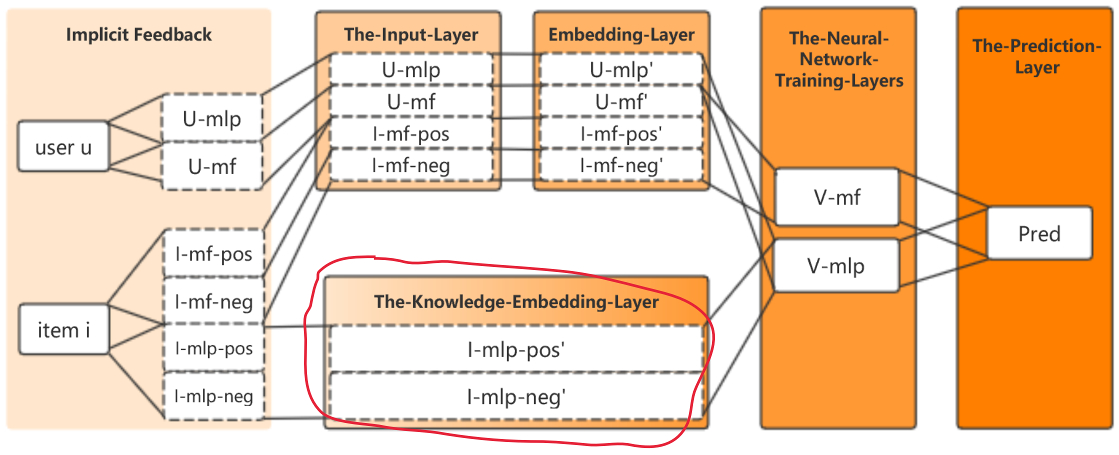

Fig. 1. Overview of the KNCR framework.

define the output of the first layer of MLP as V_{mlp_out1}. Then the final output of the MLP for the n-layer is defined as V_{mlp}, where σ is the activation function and b is the offset.

$$V_{mlp\_out1} = \sigma_1 \left(W_1^T \begin{bmatrix} U'_{mlp} \\ -I'_{mlp\_pos} \\ -I'_{mlp\_neg} \end{bmatrix} + b_1 \right) \quad (6)$$

$$V_{mlp} = \sigma_n \left(W_n^T \left(\dots \left(\sigma_2 \left(W_2^T V_{mlp\_out1} + b_2 \right) \right) \dots \right) + b_n \right) \quad (7)$$

The-Prediction-Layer is at the top, which is a fully connected layer with the input of V_{mf} and V_{mlp}. Finally, we get a value and name it as Prediction. The higher the value, the more users like positive items more than negative items. Prediction is calculated as follows:

$$Pred = \sigma_p \left(W_p^T \begin{bmatrix} V_{mlp} \\ V_{mf} \end{bmatrix} + b_p \right) \quad (8)$$

With the above settings, we then define the loss function as follows, Y represents the positive feedback data set in implicit feedback, while $Y-$ is the opposite.

$$L = -\sum_{(u,i)\in\mathcal{Y}} \log Pred - \sum_{(u,j)\in\mathcal{Y}^-} \log(1 - Pred)$$
$$= -\sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} y_{ui} \log Pred + (1 - y_{ui}) \log(1 - Pred) \quad (9)$$

B. *Knowledge Embedding Layer*

Relevant symbols involved in the model in this chapter are defined as follows: the triplet of knowledge graph is expressed as $k = (E, R, S)$, where E is the set of all entities in the knowledge graph, R is the set of all relations, and $S \subseteq E \times R \times S$ represents the set of triples in the knowledge graph. For a particular triple, it is represented by (h, r, t), where h and t represent the head and tail entities in the triple, and r represents the relationship between h and t.
According to the previous introduction, the purpose of knowledge graph representation learning is to obtain the vector

Algorithm 1 KNCR Mapping and Training Process

```
Input: $S$: a set of items; $V$: a set of high-dimensional vector;
    $e$: epochs
1: for $i$ =0 ; $i <$ length of $e$ ; $i$ ++ do
2:     initialize a set of train instance $T$;
3:     initialize a set $H = \emptyset$;
4:     for all $I$ such$I \in T$ do
5:         generate $Hi$ mapped from $V$;
6:         $H = H \cup Hi$
7:     end for
8:     for all $h$ such $h \in H$ do
9:         training KNCR model within $h$;
10:    end for
11:    evaluate model with pairwise approaches;
12: end for
```

representation of all entities and relationships of triples in low-dimensional continuous space by learning the semantic information of the triples structure in the knowledge graph. We take the item in the dataset (Movielens-1M) and the knowledge graph triple S as the input of KEM-layer. The processing process of KEM-layer is divided into four parts.

- Firstly, for each movie in the Movielens data set, we find the corresponding entity in the knowledge graph. Moreover, the corresponding entity set E, relation set R and training set S are constructed for item respectively.
- Then, use knowledge representation learning to learn the entity set E, relation set R and training set S constructed in the previous step, and generate the entity set vector matrix E'.
- Thirdly, we map E' to item and generate a higher-dimensional vector I_{em} for each item.
- In the fourth part, we select positive and negative examples from I_{em}, generate I'_{mlp_pos} and I'_{mlp_neg} as output of kem-layer, and send them to input-layer to start training.

TABLE I
BASIC STATISTICS OF THE DATASETS

Movielens-1M		YAGO		Dataset	
#ratings	1,000,209	#item-entity	3,221	#ratings	929,367
#user	6,040	#relation	6	#user	6,015
#item	3,883	#triple	43,837	#item	2,917

IV. EVALUATION RESULTS

In this section, we evaluate KNCR and the present corresponding results with a real-world dataset for movie recommendation. The experimental results demonstrate evidence of significant improvement over many competitive baselines.

A. Experimental Settings

1) Experimental Environment: We implement the algorithm with GeForce GTX 1080 on CentOS Linux release 7.2.1511. The training framework is Keras 2.0.0 with TensorFlow-gpu 1.4.0 as the backend, and the corresponding CUDA and CuDNN versions were 8.0 and 6.0 respectively.

2) Datasets Description: YAGO[1] is an open source knowledge base which automatically extracted from Wikipedia and other sources and used to build a triple for each movie entity in Movielens-1M. The format of the triple is like $< h, r, t >$. composed of two entities h, t and a relationship r where h is extracted from Movielens-1M and r from YAGO represented as followed: $< wroteMusicFor >, < directed >, < created >, < actedIn >, < edited >, < gener >$. Since not every item in Movielens-1M can find the corresponding entity in YAGO, the number of items is reduced by a certain amount compared to the original datasets.

Movielens-1M[2] contains 1,000,209 anonymous ratings of 3,883 movies made by 6,040 Movielens-1M users, which is as the primary experimental dataset for our experiment. First, to ensure the balance of the dataset as much as possible, we filtered out the users who scored less than 10 times and the movies that were scored less than 10 times. Then, we sorted the timestamp of the interaction between users and items, last interaction and second to last interaction were used as the test set and the validation set, respectively, and both of which were 6015. Basic embedding statistics of the dataset are reported in Table I, where the $Dateset$ column displays the training datasets that are actually used for training.

3) Evaluation Protocols. : To evaluate the performance of item recommendation, we adopted the leave-one-out evaluation. For each user, we held-out her latest interaction as the test set and utilized the remaining data for training. Since it is too time-consuming to rank all items for every user during evaluation, we followed the common strategy [2], [4] that randomly samples 100 items that are not interacted by the user, ranking the test item among the 100 items. The performance of a ranked list is judged by Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG). Without special mention, we truncated the ranked list at 10 for both metrics. As such, the HR intuitively measures whether the test item is present on the Top10 list, and the NDCG accounts for the position of the hit by assigning higher scores to hits at top ranks.

[1]https://github.com/yago-naga/yago3
[2]https://grouplens.org/datasets/movielens/1m/

4) Parameter Settings. : Representation learning indicate our model needs to transform the knowledge representation learning of the items in the MovieLens-1M datasets into high-dimensional vectors with correlations. In this part we choose OpenKE [16] for knowledge graph representation learning. The basic parameters we set are shown in Table II.

KNCR optimized the model with Adam, trained by binary cross-entropy loss of Equation 9. The model's hyperparameters include the learning rate, batch size, number of negative items for every positive item, and their values are set to 0.0005, 128, 2, respectively. We did different experiments for different number of dimensions(#Dim) extracted from the knowledge graph and neurons(predictive factor) in the last layer of the MLP part. For example, when we select 128-dimensional items as input, the different last layer structure of the MLP (16, 24, 32, 64) will be evaluated in order, in other word, in the first set of experiments, the order of neurons in the MLP will be $128 \rightarrow 64 \rightarrow 32 \rightarrow 24 \rightarrow 16$ as is the rest of the experiments.

TABLE II
KNOWLEDGE GRAPH REPRESENTATION LEARNING PARAMETER SETTINGS

λ	learning rate	0.001
γ	margin	1.0
d	dissimilarity measure	L1
k	latent dimension	128/256
b	Batch-size	100
opm	Optimizer	Stochastic gradient descent

5) Performance Comparison. : We compared our proposed methods of NeuMF [2] and NeuPR [4]. NeuMF is a state-of-the-art neural network based collaborative filtering method of binary cross-entropy loss. NeuPR is a general neural network based collaborative ranking method of personalized ranking. For fair comparison, we employ the same embedding size, number of hidden layers, and size of predictive factors of NueMF, NeuPR and our model. From Table III, we can see that when the Dim of representation learning is 256, the effect is better than 128. We think that the higher the value of Dim, the more meaningful the semantic information it represents, which means it is easier to mine the intrinsic relationship between item and item to improve recommendation accuracy.

Fig. 2 shows the performance of Top-K recommended lists where the ranking position K ranges from 1 to 10, which demonstrates the performance of our KNCR(TransE) and NeuMF, NeuPR. As can be seen, compared to the other two models, KNCR has the best performance regardless of the value of K. For a fair comparison, we set the predictive factor of the three models to 32. It is worth mentioning that this setting is not the best parameter for KNCR as can be seen from the following description. which also indicates that the integration of the knowledge graph into the model does improves the accuracy of the recommendation system.

TABLE III
HR@10 AND NDCG@10 COMPARISONS OF DIFFERENT METHODS $w.r.t.$ THE NUMBER OF PREDICTIVE FACTORS

Factors & Metrics		Models		KNCR							
		NeuMF	NeuPR	KNCR+TranE		KNCR+TranH		KNCR+TranR		KNCR+TranD	
				128dim	256dim	128dim	256dim	128dim	256dim	128dim	256dim
16	HR	0.6320	0.7300	0.7368	0.7420	0.7398	0.7436	0.7315	0.7443	0.7405	0.7426
	NDCG	0.4170	0.4635	0.4669	0.4700	0.4690	0.4726	0.4609	0.4705	0.4681	0.4733
24	HR	0.6311	0.7400	0.7516	0.7571	0.7526	0.7544	0.7490	0.7506	0.7520	0.7588
	NDCG	0.4295	0.4718	0.4799	0.4861	0.4810	0.4855	0.4786	0.4824	0.4802	0.4884
32	HR	0.6422	0.7425	0.7461	0.7546	0.7608	0.7579	0.7520	0.7485	0.7566	0.7539
	NDCG	0.4324	0.4735	0.4768	0.4870	0.4935	0.4911	0.4795	0.4829	0.4899	0.4883
64	HR	0.6354	0.7278	0.7390	0.7368	0.7426	0.7426	0.7463	0.7378	0.7450	0.7397
	NDCG	0.4201	0.4643	0.4723	0.4763	0.4794	0.4782	0.4828	0.4753	0.4810	0.4790

TABLE IV
COMPARISON OF DIFFERENT MODELS

Metrics	Factor@16		Factor@24		Factor@32		Factor@64	
	NeuPR	NeuMF	NeuPR	NeuMF	NeuPR	NeuMF	NeuPR	NeuMF
HR	+1.92%	+17.77%	+2.48%	+20.23%	+2.41%	+18.47%	+2.48%	+17.45%
NDCG	+2.11%	+13.50%	+3.52%	+13.71%	+4.22%	+14.13%	+3.98%	+14.93%

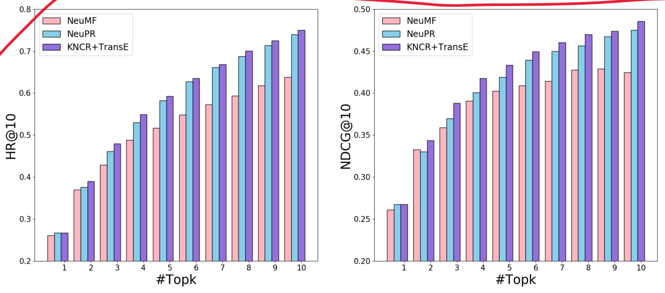

Fig. 2. Evaluation of Top-K item recommendation where K ranges from 1 to 10 on Movielens.

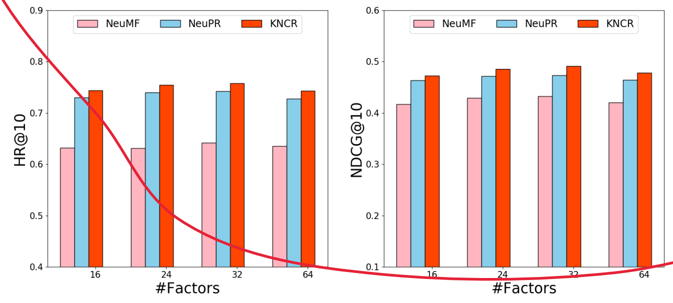

Fig. 3. Performance of HR@10 and NDCG@10 $w.r.t.$ the number of predictive factors on Movielens.

Fig. 3 shows the performance of HR@10 and NDCG@10 with respect to the number of predictive factors which also demonstrates the impact of depth of layers in KNCR. We implement transH with the Dim of 256. Even in comparison with the strongest baseline NeuMF and NeuPR, KNCR consistently outperforms them. Table. IV shows the overall superiority of KNCR compared with NeuPR and NeuMF. It is worth noting that when the predictive factor is 64, in other words, the number of layers is 4, compared to the case of 3, the performance of the model has decreased due to overfitting.

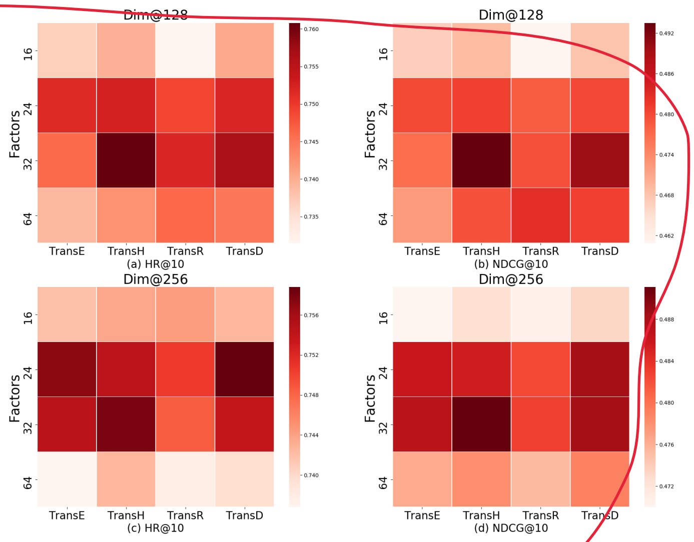

Fig. 4. Performance of HR@10 and NDCG@10 $w.r.t.$ the difference of Trans and Dim and different number of predictors.

Even in this case, KNCR performs best, as is the case with fewer number of layers.

6) *Impact of Trans and Dim. :* Fig. 4 shows the (a) and (b) respectively represent the performance of HR@10 and NDCG@10 with respect to the difference of representation learning methods (Trans) when the embedded dimension is 128, while (c) and (d) represent the embedded dimension is 256. The heat map indicates that the darker shade of red , the better the recommendation. As shown in the figure, when the embedded dimension is 128, TransH and prediction factor of 32 to achieve the best effect. When the embedded dimension is 256, it is better to use TransH and prediction factor of 32 or TransD and prediction factor of 24. Experiments show that TransH with prediction factor of 32 achieves optimal performance.

7) Training Loss. : Fig. 5 shows the (a) training loss, (b) HR@10 and (c) NDCG@10 of training process of KNCR and NeuPR on Movielens-1M. Here we employed predictive factors of 32, learning rate of 0.0005, batch-size of 128, negative sampling ratio of 2, and report the training loss within 50 iterations. We can see that with more iterations, the training loss of KNCR models gradually stabilizes and KNCR realizes HR@10 and NDCG@10 higher than NeuMF. In order to ensure the fairness of the experiment, the number of network layers in the three models is set to 5, which makes the NeuMF's loss curve too turbulent to be represented in a same figure of other models when the number of iterations exceeds 40. The above findings provide empirical evidence for the rationality and effectiveness of optimizing the log loss for learning from implicit data.

V. Conclusion and Future work

In this paper, we propose KNCR, a framework based on implicit feedback, which naturally incorporates the knowledge graph embedding into neural network based collaborative ranking framework of personalized ranking. We experimentally on the real-world public dataset illustrate the effectiveness of knowledge graph combined with neural network strategy for recommendation. For future work, we plan to (1) try other more complex neural network models combined with knowledge graph to maximize the correlation between item and item for improving the accuracy of the recommendation system; (2) apply this algorithm to recommend content other than movie ontology, and improve the performance.

Acknowledgment

In this work, Pan Wang and xxx were supported by the National Science Foundation of China (61972211)

References

[1] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[3] W. Niu, J. Caverlee, and H. Lu, "Neural personalized ranking for image recommendation," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 2018, pp. 423–431.

[4] B. Song, X. Yang, Y. Cao, and C. Xu, "Neural collaborative ranking," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2018, pp. 1353–1362.

[5] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han, "Personalized entity recommendation: A heterogeneous information network approach," in *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 2014, pp. 283–292.

[6] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 353–362.

[7] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "Ripplenet: Propagating user preferences on the knowledge graph for recommender systems," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2018, pp. 417–426.

[8] H. Wang, F. Zhang, X. Xie, and M. Guo, "Dkn: Deep knowledge-aware network for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2018, pp. 1835–1844.

[9] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[10] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Twenty-Eighth AAAI conference on artificial intelligence*, 2014.

[11] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[12] T. Qiu, Z. Ge, S. Lee, J. Wang, Q. Zhao, and J. Xu, "Modeling channel popularity dynamics in a large iptv system," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1. ACM, 2009, pp. 275–286.

[13] J.-F. Pessiot, T.-V. Truong, N. Usunier, M.-R. Amini, and P. Gallinari, "Learning to rank for collaborative filtering." *ICEIS (2)*, vol. 7, 2007.

[14] N. N. Liu and Q. Yang, "Eigenrank: a ranking-oriented approach to collaborative filtering," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2008, pp. 83–90.

[15] H. Li, "Learning to rank for information retrieval and natural language processing," *Synthesis Lectures on Human Language Technologies*, vol. 4, no. 1, pp. 1–113, 2011.

[16] X. Han, S. Cao, X. Lv, Y. Lin, Z. Liu, M. Sun, and J. Li, "Openke: An open toolkit for knowledge embedding," in *Proceedings of EMNLP*, 2018, pp. 139–144.

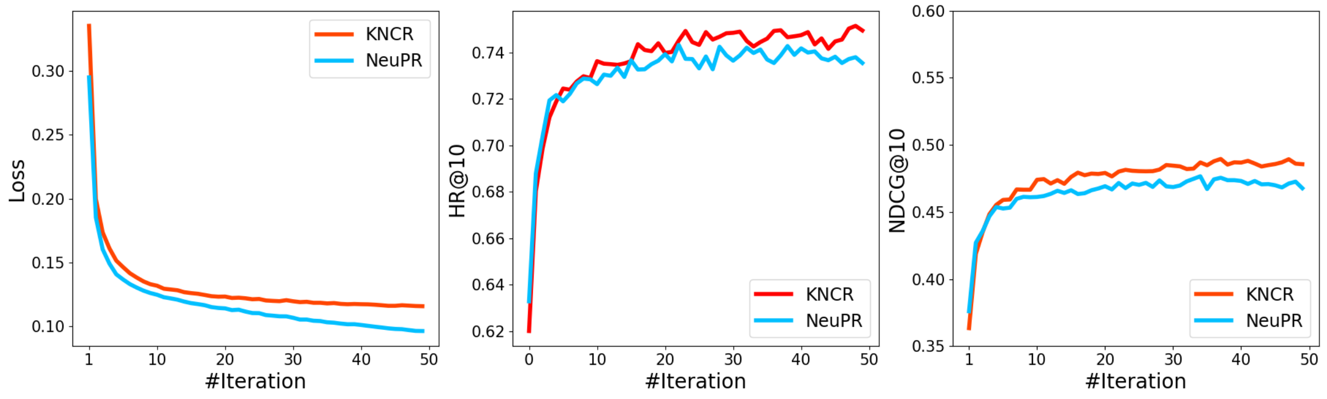

Fig. 5. Training loss and recommendation performance of KNCR methods $w.r.t.$ the number of iterations on Movielens.