Context-aware Graph Collaborative Recommendation Without Feature Entanglement

Tianyi Gu, Ping Li(✉), and Kaiwen Huang

Southwest Petroleum University, Chengdu, China

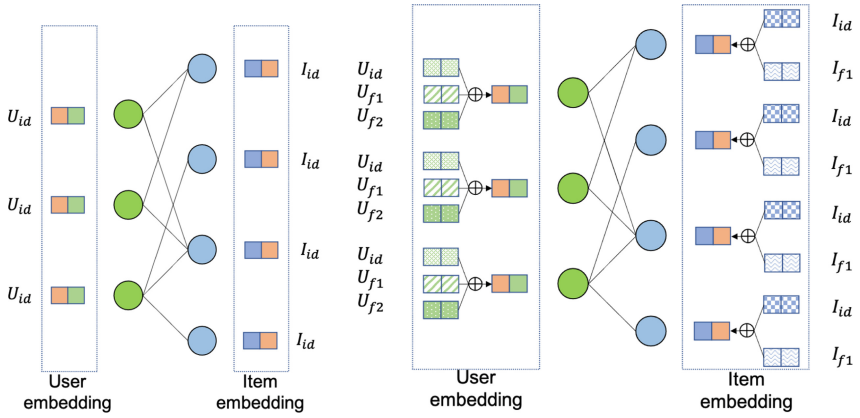
Abstract. Inheriting from the basic idea of latent factor models like matrix factorization, current collaborative filtering models focus on learning better latent representations of users and items, by leveraging the expressive power of deep neural networks. However, the settings where rich context information is available pose difficulties for the existing neural network based paradigms, since they usually entangle the features extracted from both IDs and other additional data (*e.g.,* contexts), which inevitably destroy the original semantics of the embeddings. In this work, we propose a context-aware collaborative recommendation framework called CGCR to integrate contextual information into the graph-based embedding process. Our model converts the bipartite graph to a homogeneous one by placing the users and items in the identical feature space. As our method is free of feature crosses, it can preserve the semantic independence on the embedding dimensions and thus improves the interpretability of neural collaborative filtering. We use generalized matrix factorization as the matching function so that the model can be trained in an efficient non-sampling manner. We further give two examples of CGCR: LGC with linear graph convolutional networks and LGC+ with attention mechanism. Extensive experiments on five real-world public datasets indicate that the proposed CGCR models significantly outperform the state-of-the-art methods on the Top-K recommendation task.

Keywords: Collaborative filtering · Top-K recommendation · Matrix factorization · Graph neural network

1 Introduction

As one of the powerful personalized recommendation approaches in modern recommender systems, collaborative filtering makes prediction based on users' past preference embodied in the user-item interactions of similar users as shown in Fig. 1(a). When it comes to extracting users' preference or interest from the observations, matrix factorization [16] plays an important role. In general, matrix factorization finds a common low-dimensional space to describe users and items

H. Gao and X. Wang (Eds.): CollaborateCom 2021, LNICST 406, pp. 259–276, 2021.
https://doi.org/10.1007/978-3-030-92635-9_16



(a) Pure CF (b) CF with contexts

Fig. 1. The traditional collaborative filtering method and a typical treatment of contextual feature fusion. U_{id} and I_{id} denote the embeddings of user ID and item ID, while U_{fi} and I_{fi} represent the embeddings of the i-th feature (contextual information) for user and item, respectively.

wherein latent features (factors) are indicated by the dimensions, and then compare the user/item vectors directly using dot product. To further improve the representations in latent feature space, prior work resorts to deep neural networks [10]. In particular, the very recent work [7,24] shows that the high-order interactions among features can be successfully captured via graph convolution operation on the user-item bipartite graph.

However, in real recommendation processes, it may be inadequate to consider only the user-item interaction behaviors. It is also important to incorporate rich contextual data into the models to adapt the recommender system to specific situation of the user, which challenges current collaborative filtering scheme. An intuitive solution for introducing the auxiliary information (*e.g.,* user's gender, age or item's price) to the existing graph-based collaborative filtering models, is to pool the embedding of the features into one vector (as shown in Fig. 1(b)) for further convolutional embedding learning. The consequence of such feature entangling is the lack of interpretability, that is, the model is not able to learn meaningful and explainable representations for users and items. This problem will be compounded by graph convolution. In fact, the graph defined by the user-item interactions is naturally a heterogeneous graph, implying that different types of nodes may fall in different feature spaces. However, the traditional graph convolution aggregates different types of features from different semantic space. As a result, the original semantics of node embeddings are destroyed [23].

To tackle the problem of semantic damage when introducing contextual information into collaborative filtering, we design a novel **C**ontext-aware **G**raph **C**ollaborative **R**ecommendation framework (**CGCR**), which consists of three parts: *Feature space construction Layer*, *Graph Neural Network Layer*, and *Pre-*

diction Layer. To preserve the embedding semantics of users and items, we first construct a shared feature space that embodies all features from user and item, so that the original user-item bipartite graph can be easily converted to a homogeneous one. To overcome the representation entangling caused by feature crossing, we discriminate information from different feature fields. Then we use graph convolutional network to obtain high-order information contained in the user-item interactions. Finally, we adopt generalized matrix factorization [4] as matching function and train the proposed model in an efficient non-sampling manner. Different from NCF [10], our model endows the representation learning process with transparency and thus can link the feature fields (or factors) to the prediction results. Besides the visualized explanation, we also evaluate the effectiveness of the model on five real-world datasets. The experimental results show the superior performance in context-aware recommendation tasks, compared to prior feature-crossing-based methods.

The main contributions of this work are as follows:

- We propose a new graph-based recommendation framework for context-aware collaborative filtering. By putting the feature representations of users and items into the same subspace, our model is able to incorporate contextual information to the graph convolutional embedding process on the resultant homogenous graph.
- We further propose two examples of CGCR, namely LGC and LGC+, and for the first time use the efficient non-sampling scheme as the optimization solution for the context-aware collaborative filtering.
- Besides visualizing explanations, we also conduct extensive experiments to evaluate the proposed models on five benchmark datasets (two of them are for context-aware recommendation while the rest is for pure collaborative filtering tasks). The results show the significant improvement of our two models over several state-of-the-art models, in terms of model effectiveness and interpretability.

2 Problem Formulation

We formulate the problem of context-aware collaborative filtering as follows: Given the collections of users and items, which are denoted by $\mathbf{U}$ and $\mathbf{V}$ respectively, the user-item interaction pattern is depicted by the matrix $\mathbf{R}$, where $[r_{uv}] \in \{0, 1\}$ indicates whether user u interacts with item v. Then the goal of the collaborative filtering recommendation task is to recommend a list of items that target user u may be interested in, which is formally defined as follows:

Input: Users $\mathbf{U}$, items $\mathbf{V}$ and user-item interaction matrix $\mathbf{R}$.

Output: A ranked item list based on the predicted probability $\hat{r}_{uv}$ that user u would interact with item v (ordered from highest to lowest).

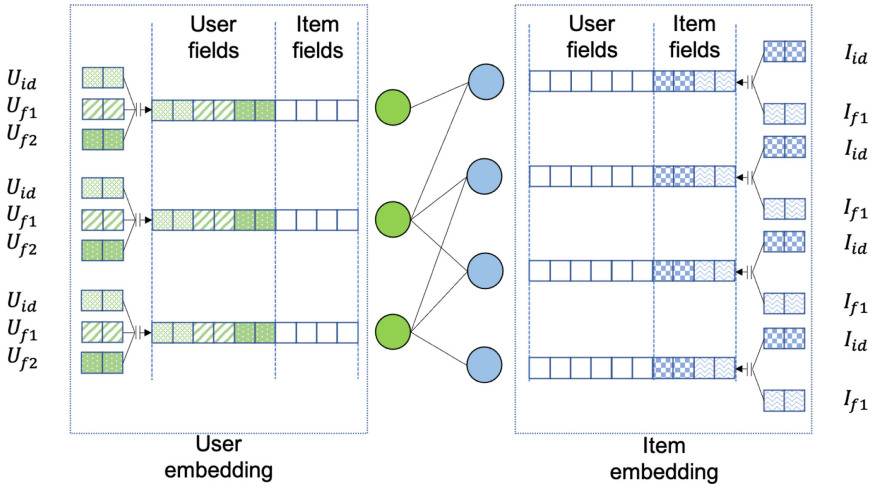


Fig. 2. Feature fusion in the *Feature Space Construction Layer* of the CGCR framework. All colored boxes indicate that the position has been filled with real values, and a blank box indicates that the position is empty and filled with 0.

3 Methodology

In the following, we introduce the proposed context-aware collaborative recommendation framework CGCR, which is based on the aforementioned learning scheme. We first explain the general framework of CGCR. Then, we use generalized matrix factorization with a non-sampling strategy to learn the parameters. Finally, we discussed the model complexity of CGCR.

3.1 General Framework

There are three components in CGCR: (1) a *Feature Space Construction Layer* that initializes the embeddings of users and items and maps them onto the same subspace; (2) a *Graph Neural Network Layer* that aggregates neighbor information onto the target node; and (3) the *Prediction Layer* using generalized matrix factorization scheme.

Feature Space Construction Layer. In the very recent neural collaborative filtering model [10], the embedding layer encodes user IDs and item IDs with multi-layer perceptron to account for the nonlinear interactions. However, in the settings where contextual information is available, how to merge the context data into the embedding should be carefully designed. The reason is that current neural network based collaborative filtering models regard the information about users and items as homogeneous by default (i.e., coming from the same feature space), while different contexts actually have different semantics. To preserve the semantics of various types of features and offer the model explanability, it is

necessary to construct a new feature space that contain all factors involved in recommendation. A natural choice is to expand the k dimensional feature space of users (items) with the r dimensional feature space of items (users). Then, the dimension of the derived feature space is $k + r$. Specifically, we construct a new shared feature space for users and items by concatenating all the initial embedding dimensions in the order of *user ID*, *user features*, *item ID*, and *item features* as follows:

$$\mathbf{E} = [\underbrace{\mathbf{e}_{u_1}, \cdots, \mathbf{e}_{u_n}}_{user\ IDs}, \underbrace{\mathbf{e}_{u_{n+1}}, \cdots, \mathbf{e}_{u_N}}_{user\ features}, \underbrace{\mathbf{e}_{v_1}, \cdots, \mathbf{e}_{v_m}}_{item\ IDs}, \underbrace{\mathbf{e}_{v_{m+1}}, \cdots, \mathbf{e}_{v_M}}_{item\ features}]. \tag{1}$$

This operation allows us to convert a heterogeneous bipartite graph to be homogeneous. As is graphically depicted in Fig. 2, there are only single type of nodes represented by concatenated vectors[1] in the same feature space, which endows the model with the flexibility to leverage the inherent complex relations in the data by graph neural network techniques while keeping each dimension relatively independent.

It is worth noting that the expansion of feature space in the form of vector concatenation also allows each factor (or contextual feature) to have its specific role in the representation learning, which is different from commonly used sum operation in feature fusion. The latter potentially assumes that the summed features have the same weight in forming new representation.

Graph Neural Network Layer. The core to achieve collaborative filtering is to capture users' historical preference as well as the high-order correlations between users and items, which remind us of recent graph neural networks [15,25,28]. The aggregation and message passing involved in graph neural networks can easily resolve those two problems, on top of the converted homogeneous graph. According to graph convolution, the embeddings of users and items can be updated uniformly as:

$$\mathbf{e}_i^{(l+1)} = AGG(\{\mathbf{e}_j^{(l)} : j \in \mathcal{N}_i\}), \tag{2}$$

where the pair (i, j) corresponds to the user-item interaction in a bipartite graph and $\mathcal{N}_i$ is the neighbor of node i. $AGG(\cdot)$ is the aggregation function that can be specified. Here are two aggregation functions, LGC and LGC+ using attention. Their aggregation functions are organized as shown in Table 1.

To make the embeddings more traceable, we restrict each layer of graph convolution with constant dimension. Due to feature space expansion, the initial embeddings will contain many 0s in the extended dimensions (indicated by blanks in Fig. 3). Thus an interesting phenomenon in the learning process is that, the information is updated only on a part of the embedding. Take the user

[1] An example of concatenation (*aka.* $\|$) is that $[a_1, a_2, \ldots, a_m] \| [b_1, \ldots, b_n] = [a_1, a_2, \ldots, a_m, b_1, \ldots, b_n]$, where a and b are both scalar.

Table 1. Neighbor aggregation function at a graph convolutional layer for LGC and LGC+.

Model	Neighbor aggregation function
LGC	$\mathbf{e}_v^{(k)} = \sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{\lvert\mathcal{N}(v)\rvert \cdot \lvert\mathcal{N}(u)\rvert}} \mathbf{e}_u^{(k-1)}$
LGC+	$\mathbf{e}_v^{(k)} = \mathrm{ReLU}(\sum_{u \in \mathcal{N}(v) \cup \{v\}} a_{vu}^{(k)} W^{(k)} \mathbf{e}_u^{(k-1)})$

embedding as an example: at the first graph convolution layer, the aggregation for the user is the embeddings of its preferred items, and these item embeddings do not contain any user information. That is, the update occurs only on the item dimensions. Similarly, the update of items' embeddings only involve users' information. From this process one can see that our architecture does not introduce any feature entanglement. The final representations of the users and items are obtained by taking the sum of the embeddings in each layer.

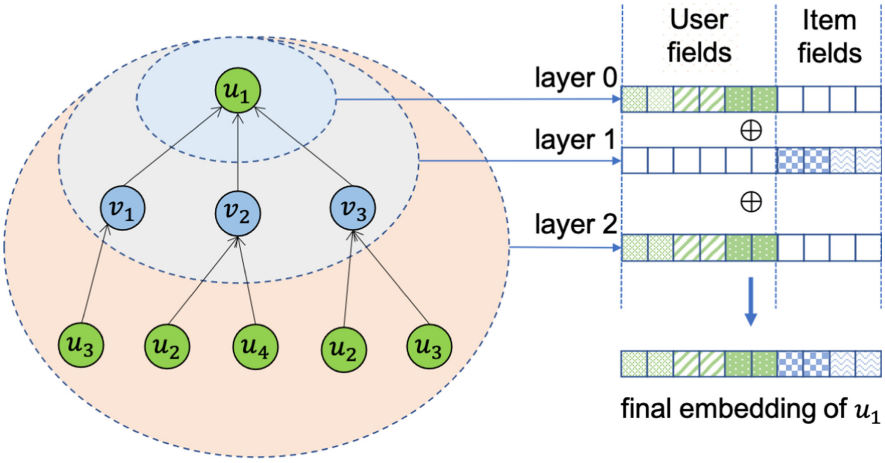


Fig. 3. An illustration of *Graph Neural Network Layer* of the CGCR framework. The result of each layer of graph convolution will leave blanks, and the final embedding representation of u_1 is the element-wise sum of result of each layer.

To implement the model more efficiently, we provide the matrix form of LGC. We first construct the adjacency matrix corresponding to the homogeneous graph using:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^\top & \mathbf{0} \end{bmatrix}, \tag{3}$$

where $\mathbf{0}$ now is all-zero matrix. The Laplacian matrix $\mathcal{L}$ for the adjacency matrix is formulated as:

$$\mathcal{L} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}, \tag{4}$$

where $\mathbf{D}$ is the diagonal degree matrix. Let the output of the *feature space construction layer* be $\mathbf{E}^{(0)}$, our goal is $\mathbf{E}^{(l)}$ after l-layer graph convolution. By applying the matrix-form propagation rule, we have:

$$\mathbf{E}^{(l)} = \mathcal{L}\mathbf{E}^{(l-1)} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{E}^{(l-1)}. \tag{5}$$

As a result, we can simultaneously update the representations for all users and items in a rather efficient way. Similar to LightGCN, we only aggregate the immediate neighbors and do not integrate the target node itself, which means that there is no self-connection in each graph convolution layer.

For the final representations, we combine the results of all graph convolution layers:

$$\mathbf{E} = \alpha_0\mathbf{E}^{(0)} + \alpha_1\mathbf{E}^{(1)} + \cdots + \alpha_l\mathbf{E}^{(l)}. \tag{6}$$

In our experiments, we set α uniformly as $\frac{1}{l+1}$, which is equivalent to the mean pooling of all vectors.

Prediction Layer. Different from the way of employing inner product in the existing graph neural network based collaborative filtering models [4,10], we use the more generalized similarity function to estimate the matching degree as follows:

$$\hat{r}_{uv} = \mathbf{h}^\top(\mathbf{e}_u \odot \mathbf{e}_v), \tag{7}$$

where $\mathbf{e}_u = \mathbf{e}_u^{(1)} \oplus \mathbf{e}_u^{(2)} \oplus \ldots \mathbf{e}_u^{(l)}$ (so is $\mathbf{e}_v$) is the learned embedding representation, and $\mathbf{h}$ is what we call prediction layer.

The above generalized similarity measure provides our model better interpretability. Note that the feature dimensions are never entangled during representation learning and matching, input feature dimensions can be directly associated with the components of $\mathbf{h}$ to manifest their influences on recommending certain items. Since the contexts and IDs are embedded with various dimensions, we evaluate the relative importance of a factor (or feature) as:

$$s_i = \frac{\sum_{k=1}^{d} \mathbf{h}_{i_k}}{\sum_{p=1}^{|\mathbf{h}|} \mathbf{h}_p}, \tag{8}$$

where i indicates the i-th factor whose embedding dimension is d. This way, our model exhibits not only the transparency in representation learning but the explanability in prediction.

3.2 Optimization

Instead of using the most popular *Bayesian Personalized Ranking* [20] based on negative sampling to learn the model, we train the model on the whole data with

the latest proposed *Efficient Non-sampling Matrix Factorization* [4](ENMF). Following this setting, we have the loss of the predictions[2] as:

$$\begin{aligned} \boldsymbol{L}(\theta) = & \sum_{u \in \mathbf{U}} \sum_{v \in \mathbf{V}^+} ((c^+ - c^-)\hat{r}_{uv}^2 - 2c^+ \hat{r}_{uv}) \\ & + \sum_{i=1}^{d} \sum_{j=1}^{d} ((h_i h_j)(\sum_{u \in \mathbf{U}} e_{u,i} e_{u,j})(\sum_{v \in \mathbf{V}} e_{v,i} e_{v,j})). \end{aligned} \tag{9}$$

Note that there are two hyper-parameters, namely, c^+ and c^-, in the objective. They are used to tune the penalty of poor predictions. In practice, we always value the positive samples than negative ones. So the weight of the prediction on the positive samples is often set to 1, and the weight corresponding to the negative samples will be a smaller positive number (lying between 0 and 1). However, different weights on negative sampling could have different impacts on the performance of the model, which will be explored in the experimental part.

3.3 Time Complexity Analysis of CGCR

In addition, compared to negative sampling, we use the efficient non-sampling framework to greatly reduce the training time. It can be found that the time complexity of the whole learning process is primarily determined by the computation of the loss shown in Eq. (9), which is similar to ENMF, namely, $O((|\mathbf{U}| + |\mathbf{V}|)d^2 + |\mathbf{R}|d)$ for d-dimensional embeddings. However, it should be noted that the feature space expansion in the first layer of our model compromises the computation efficiency, compared to ENMF model, as it enlarges the embedding dimensions.

4 Experiments

We now perform experiments to (1) compare CGCR framework to several strong baseline methods in collaborative filtering on five real-world datasets; (2) and analyze the interpretation of our model with visualization. In addition, we explore the effects of negative samples and other hyper-parameters involved in the model.

4.1 Dataset Description

To evaluate the performance of the proposed method, we conduct extensive experiments on the following *implicit feedback* datasets: ***Movielens***[3], ***Zhihu***[4],

[2] The proof of Eq. (9) see the previous work [4] for the details.
[3] https://grouplens.org/datasets/Movielens/1m/.
[4] https://github.com/THUIR/CC-CC/tree/master/dataset.

Table 2. Statistics of the datasets.

Dataset	Movielens	Zhihu	Pinterest	Tiktok	KKBOX
User	6,040	9,177	55,187	18,855	24,613
Items	3,706	9,946	9,916	34,756	61,877
User fields	4	4	2	2	2
Item fields	2	3	2	2	2
Instances	1,000,209	810,485	1,500,809	1,493,532	2,170,690
Density	0.0447	0.0088	0.0027	0.0023	0.0014

Pinterest[5]. ***Tikiok***[6], ***KKBOX***[7]. We summarize the statistics of three datasets in Table 2:

- ***Movielens*** is widely used as the benchmark for recommendation performance evaluation. In our experiment, we choose the version that contains one million ratings and binarize the data into implicit feedback. User contextual information includes user ID, gender, age, and occupation. The context of items consists of movie ID and movie genre.
- ***Zhihu*** is obtained from the previous work [21]. Specifically, besides the IDs, for user features, we preserve the number of focusing users, the number of focusing questions, and the city where the user is located; as for item features, the numbers of item likes and item comments are kept.
- ***Pinterest***, ***Tiktok***, ***KKBOX*** are all used for pure collaborative filtering without contextual data. In this work, we use this dataset to demonstrate that our model can not only be applied to context-aware recommendations but pure collaborative filtering tasks. In order to exploit the characteristics of feature combination of the FM-based models, we added a tag feature after the ID of the user and the item, to indicate that the ID belongs a user or an item.

4.2 Experimental Settings

Baseline Methods. We compare our models with the following state-of-the-art methods:

- **FM** [18]: Though the original FM is written in C++ [19], to make comparisons in the same environment, we use the package built-in TensorFlow framework.
- **AFM** [26]: AFM adds an attention mechanism to the FM framework so that more useful feature combination items will have more positive effects on the results.

[5] https://github.com/hexiangnan/adversarial_personalized_ranking/tree/master/Data.
[6] http://ai-lab-challenge.bytedance.com/tce/vc/.
[7] https://www.kaggle.com/c/kkbox-music-recommendation-challenge/data.

- **NFM** [6]: NFM adds MLP to the FM framework to recognize more complex patterns in the data. It can be regarded as the parallel work of AFM.
- **ENMF** [4]: ENMF is a non-sampling framework which learns from the whole training data without sampling by reformulating a commonly used square loss function with rigorous mathematical reasoning.
- **ENSFM** [3]: ENSFM tackles the recommendation problem in context scenarios, which incorporates FM component in the architecture to implement the second-order interactions among the user-self features and item-self features. ENSFM is special in that it uses its own new non-sampling method.
- **LightGCN** [7]: LightGCN is currently the most popular graph-based recommendation model for pure collaborative filtering. In the experiments on Pinterest dataset, LightGCN only uses the IDs of users and items.

Since both LightGCN and ENMF are not designed for context-aware recommendation, to adapt the contextual datasets to these methods, we use sum-pooling to combine the features of all users (or items) into a vector as the user's (or item's) ID embedding on Movielens and Zhihu datasets. Although the learning process (graph convolution) of the proposed LGC is quite close to LightGCN, the original LightGCN is learned with BPR loss, which encourages the prediction of an observed entry to be higher than its unobserved counterparts. For a fair comparison, we implement a non-sampling version for LightGCN using the loss depicted in Eq. (9).

Parameter Settings. There is an important parameter in the non-sampling framework we adopt here, namely, the negative sample weight c^-. To see the impact of this parameter on the performance and find the optimal weight for the models, we explore a range of values. In particular, for the baselines (ENMF, ENSFM, LightGCN) and our models (LGC and LGC+), we search the negative sample weight in $\{0.005, 0.01, 0.1, 0.3, 0.5, 1\}$ and learning rate in $\{0.005, 0.01, 0.02, 0.05\}$. Based on the optimal weights, we can choose the proper embedding dimensions for ENMF, ENSFM, LightGCN, LGC and LGC+, while for FM, AFM, and NFM that are based on negative sampling, we fix the embedding dimension to 32. We optimize all models with the Adam [14] optimizer, where the batch size is fixed at 512 for FM, AFM and NFM, and 256 for the rest. Besides, the attention factors for AFM is set as 32, the number of MLP layers for NFM is set as 1 with 32 neurons. For LightGCN, LGC and LGC+, the number of layers of graph convolution is set in $\{1, 2, 3, 4\}$.

Evaluation Metrics. To evaluate the effectiveness of top-K recommendation and preference ranking, we adopt two widely-used evaluation protocols: ***HR@K*** and ***NDCG@K*** [13]. To speed up the computation of metrics, the common practice presented in recent work [5,9,10] is to use sampled metrics wherein only a smaller set of random items and the relevant items are ranked. Sampled metrics, however, are inconsistent with their exact version [17]. Therefore, in our experiment, we adopt the *leave-one-out* scheme of *all ranking*, that is, we treat

all items the user does not interact with as negative samples and rank them. Although this scheme makes the results presented in this work smaller than that of the sampling evaluation scheme, it guarantees the reliability of the evaluation.

4.3 RQ1: Does the Proposed Method Perform Better Than Other Comparison Methods?

We first compare our proposed models with the baseline methods. We investigate the *top-K* performance when K is 20 and number of Graph Convolution Layer is 3. For the purpose of a fair comparison, the embedding size is set as 32 for all approaches. The performance of all methods on five datasets is shown in Table 3, where the percentages of relative improvement on each metric are also presented.

Table 3. Overall performance comparison. NG stand for NDCG. The best result of the state-of-the-art methods is underlined.

	Movielens		Zhihu		Pinterest		Tiktok		KKBOX	
	HR	NG	HR	NG	HR	NG	HR	NG	HR	NG
FM	0.1156	0.0457	0.1064	0.0426	0.0777	0.0291	0.0683	0.0218	0.2101	0.1161
AFM	0.1258	0.0499	0.1519	0.0617	0.0848	0.0322	0.0722	0.0254	0.2643	0.1444
NFM	0.1291	0.0501	0.1696	0.0704	0.0952	0.0373	0.0739	0.0277	0.2681	0.1479
ENMF	0.1459	0.0584	0.2044	0.0876	0.0968	0.0384	0.0753	0.0302	0.2785	0.1516
ENSFM	0.1525	0.0582	0.2074	0.0897	0.1058	0.0412	0.0813	0.0316	0.2822	0.1579
LightGCN	0.1515	0.0593	0.2064	0.0882	0.1011	0.0397	0.0827	0.0319	0.2823	0.1588
LGC	0.1566	0.0611	0.2134	0.0924	**0.1154**	**0.0458**	0.0877	0.0342	0.2920	**0.1659**
LGC+	**0.1566**	**0.0627**	**0.2178**	**0.0936**	0.1146	0.0457	**0.0881**	**0.0344**	**0.2921**	0.1657

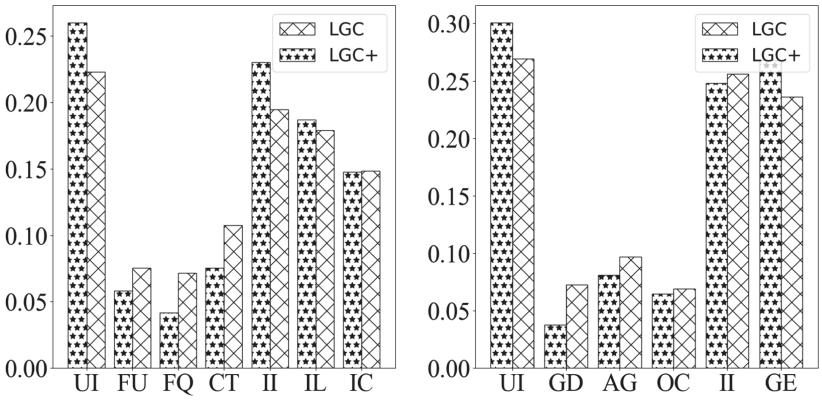
Based on these results, the following observations are drawn:

- Under the non-sampling framework, our model is more effective than LightGCN, though these two methods use the similar graph network for representation learning. Albeit introducing contextual information into LightGCN, the experimental results prove that our model is superior to LightGCN. Moreover, the comparison between our model and LightGCN sheds light on the positive effect of preserving semantics of the embedded dimensions on recommendation performance, which coincides with the prior conclusion [8,27] that points out the importance of maintaining semantics.
- By using the attention mechanism in LGC+, LGC is further improved in most cases. It is also worth noting that LGC+ surpasses the most competitive baseline method ENSFM, whose performance is better than ENMF on the three datasets (Movielens, Zhihu, Tiktok). The superiority of LGC+ may root in the non-linearity embodied in attention mechanism.

4.4 RQ2: Does the Proposed Method Elucidate the Meanings of Each Dimension of the Embedding?

The prediction layer is an important part of the CGCR model, which not only acts as a matching function to calculate the similarity between user-item

pairs, but also offers the model interpretability. Thanks to distinguishing feature domains between users and items and no feature entanglement in every step, the prediction layer can be used to identify the importance of feature domains, which is indicated by the learned parameters $\mathbf{h}_i$. More specifically, we normalize the learned $\mathbf{h}$ and use the Eq. (8) to quantify the importance score of each feature domain.



(a) Zhihu (b) Movielens

Fig. 4. The weight distribution of the prediction layer on Zhihu and Movielens (UI: user id; FU: focusing users; FQ: focusing questions; CT: city; II: item id; IL: item likes; IC: item comments; GD: gender; AG: age; OC: occupation; GE: genre).

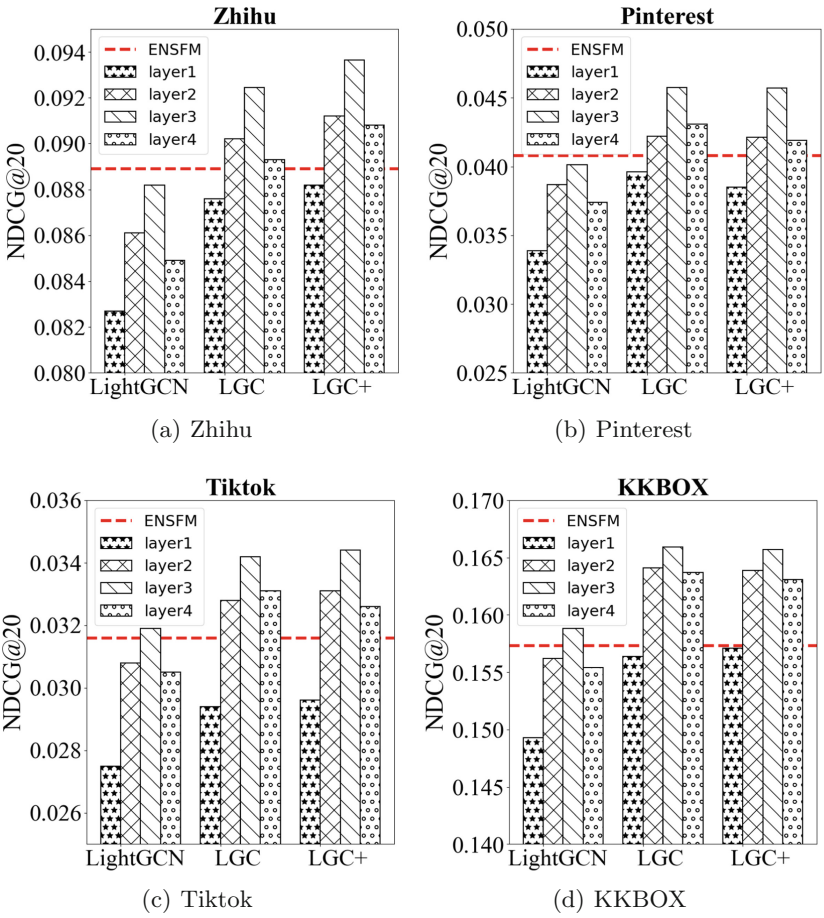
We provide the visualization of explainable results in Fig. 4. We can see that

- First and most intuitive, the importance of feature domains is heterogeneously distributed. Among the features, the most important feature is the user IDs and item IDs, followed by the item features like 'Item likes' and 'Item comments' in Zhihu dataset, while several user features are relatively insignificant, fitting well with our intuitions. For example, in Movielens it is shown from Fig. 4(b) that models' recommendation depends mainly on the category of the film, besides the ID features that embed the historical behaviors of the users. It thus give insight into the roles of different feature domains.
- For LGC and LGC+, the importance of different feature domains is marginal, which shows: 1) Introducing attention mechanism into CGCR has little effects on the importance of the feature domain; 2) In both LGC and LGC+, the IDs of users and items are the most important features. That is, historical records play a leading role in the recommendation, which further explain the reason that the proposed models work well on pure collaborative filtering task, even though it is customized for context-aware recommendation.

Note that LightGCN, which also uses the similar prediction layer, fail to find explanations from $\mathbf{h}$. As in LightGCN, the user(item) features are interacted

with each other, it is difficult to elucidate the meanings of each dimension of the embedding or separate latent factors.

4.5 RQ3: How Does Number of Graph Convolution Layer Impact?



(a) Zhihu (b) Pinterest

(c) Tiktok (d) KKBOX

Fig. 5. Effect of graph convolution layers.

To see whether LGC and LGC+ can benefit from multiple embedding propagation layers, we vary the model depth. Figure 5 summarizes the experimental results, from which we have the following observations and analysis:

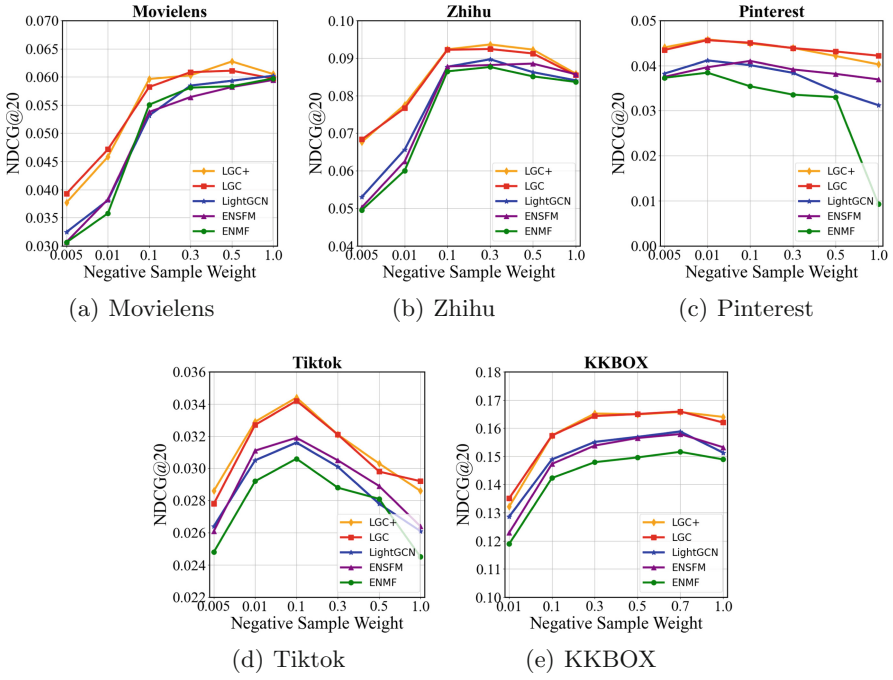
– It is shown on the four datasets that the increase of the convolutional layer will definitely increase the effectiveness of the model, and when the number of graph convolutional layer reaches 3, the effectiveness of the model will

be weakened. The experiments further empirically verify the drawback of graph networks, that is, deeper structure may lead to the oversmoothing that makes all the node embeddings move closer to each other, thereby drastically reducing the recommendation performance.

– We particularly compare our model with the state-of-the-art graph network based model LightGCN, and perform pure collaborative filtering task on Pinterest, Tiktok and KKBOX. By comparing the metric's variation among the models, we observe that the proposed models are more robust against oversmoothing inherent in graph convolutional networks, for both context-aware recommendation (Movielens and Zhihu) and pure collaborative filtering task (Pinterest, Tiktok and KKBOX). Here again, the experimental results demonstrate the effectiveness of our feature reconstruction on representation learning.

4.6 RQ4: How Does Non-sampling Strategy Impact?

Since the negative sample weight plays a key role in the non-sampling strategy, next we study its impact on the recommending performance under various metrics.



(a) Movielens (b) Zhihu (c) Pinterest

(d) Tiktok (e) KKBOX

Fig. 6. Negative sample weight of each model under three datasets.

We visualize the results in Fig. 6 and reach the following conclusions:

- The shapes of the curves indicate that all models with non-sampling strategy are sensitive to negative sample weights, though the extent of the impacts differs on different datasets. This suggests that in practical applications, the first priority is to search for an optimal weight for certain dataset.
- Note that although the models have different optimal negative sample weights for different datasets, their optimal values on the same dataset are almost the same (corresponding to the peaks of the curves in each plot). This phenomenon reflects the fact that there exists a unique optimal weight in ENMF framework based models on specific dataset, which inspires us a way of fast finding the proper negative weight for the proposed models (e.g., LGC/LGC+) in practice: one could first search for the optimal negative sample weight using ENMF or ENSFM model, and then use it directly in LGC's hyper-parameter setting. This approach can largely improve the learning efficiency.

5 Related Work

5.1 Collaborative Filtering

Recently, graph neural networks (GNNs) [15] have received increasing attention due to their great capacity in various tasks. Inspired by the success in other fields such as node classification and link prediction, researchers also investigate the suitability of GNNs for the task of recommendation. Representative works like NGCF [24] and LightGCN [7] have proven that GNN can significantly enhance collaborative filtering. In addition, side information like the features or attributes of users and items can also play a non-negligible role in graph-based collaborative filtering. However, there is no ideal solution to model graph-based collaborative filtering with additional attributes and features. This is our original intention of proposing CGCR.

5.2 Non-sampling Learning for Top-K Recommendation

It is critically important to design recommendation algorithms that can work with implicit feedback data. To learn from a sparse data, there are generally two optimization strategies: 1) negative sampling strategy and 2) non-sampling (whole-data based) strategy. The first strategy samples negative instances from missing entries, which has been widely adopted for efficient training in previous work [10,20]. However, some studies [2,11,30] have shown that sampling can limit the performance of recommendation, as it is not robust but highly sensitive to the sampling distributions. In contrast, non-sampling strategy sees all the missing data as negative and leverages the whole data with a potentially better coverage, but computation efficiency can be an issue. Recently, some efforts [1,4,11,12] have been devoted to resolving the inefficiency of non-sampling learning. Among them, we point out that ENMF [4] framework is most

relevant to our work. ENMF is to learn neural recommendation models from the whole training data without sampling. By reformulating a commonly used square loss function with rigorous mathematical reasoning, each parameter can be updated in a manageable time complexity without sampling, which makes it consistently and significantly outperform the state-of-the-art recommendation models in terms of both recommendation performance and training efficiency.

6 Conclusion and Future Work

In this work, we tackle the problem of semantic separation when integrating contextaul and user-item interaction information in collaborative filtering recommendation. We have proposed a simple but interpretable framework CGCR for context-aware Top-K recommendation. CGCR not only connects graph neural networks and matrix factorization, but also preserves the semantic independence of each feature domain. As a result, CGCR achieves two remarkable advantages: 1) offers a solution to convert a heterogeneous graph to the homogeneous one; 2) seamlessly incorporates contextual features to collaborative filtering with better explainability. Extensive experiments on five real-world datasets have shown that the proposed CGCR consistently and significantly outperforms the state-of-the-art methods including feature-crossing-based models NFM, AFM, and CFM, in terms of both recommendation performance and efficiency.

The proposed CGCR framework is not limited to the recommendation task presented in this paper. We believed that this framework could benefit many other tasks where heterogeneous graph is used. In the future, we will explore our CGCR method on the related tasks like social recommendation [29] and knowledge graph-based recommendation [22].

References

1. Chen, C., Zhang, M., Ma, W., Liu, Y., Ma, S.: Jointly non-sampling learning for knowledge graph enhanced recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (2020)
2. Chen, C., et al.: An efficient adaptive transfer neural network for social-aware recommendation. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (2019)
3. Chen, C., Zhang, M., Ma, W., Liu, Y., Ma, S.: Efficient non-sampling factorization machines for optimal context-aware recommendation. In: Proceedings of The Web Conference 2020 (2020)
4. Chen, C., Zhang, M., Zhang, Y., Liu, Y., Ma, S.: Efficient neural matrix factorization without sampling for recommendation. ACM Trans. Inf. Syst. (TOIS) **38**, 1–28 (2020)
5. Deng, Z.H., Huang, L., Wang, C.D., Lai, J., Yu, P.S.: Deepcf: a unified framework of representation learning and matching function learning in recommender system. In: AAAI (2019)

6. He, X., Chua, T.S.: Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (2017)
7. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: Lightgcn: simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (2020)
8. He, X., Du, X., Wang, X., Tian, F., Tang, J., Chua, T.S.: Outer product-based neural collaborative filtering. In: IJCAI (2018)
9. He, X., He, Z., Du, X., Chua, T.S.: Adversarial personalized ranking for recommendation. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (2018)
10. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web (2017)
11. He, X., Zhang, H., Kan, M.Y., Chua, T.S.: Fast matrix factorization for online recommendation with implicit feedback. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (2016)
12. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: 2008 Eighth IEEE International Conference on Data Mining, pp. 263–272 (2008)
13. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. **20**, 422–446 (2002)
14. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2015)
15. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
16. Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Comput. **42**(8), 30–37 (2009)
17. Krichene, W., Rendle, S.: On sampled metrics for item recommendation. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2020)
18. Rendle, S.: Factorization machines. In: 2010 IEEE International Conference on Data Mining, pp. 995–1000 (2010)
19. Rendle, S.: Factorization machines with libfm. ACM Trans. Intell. Syst. Technol. **3**, 57:1–57:22 (2012)
20. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: bayesian personalized ranking from implicit feedback. In: UAI, pp. 452–461 (2009)
21. Shi, S., et al.: Adaptive feature sampling for recommendation with missing content feature values. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management (2019)
22. Wang, X., He, X., Cao, Y., Liu, M., Chua, T.S.: Kgat: knowledge graph attention network for recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019)
23. Wang, X., He, X., Chua, T.S.: Learning and reasoning on graph for recommendation. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management (2019)
24. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (2019)

25. Wu, F., Zhang, T., Souza, A., Fifty, C., Yu, T., Weinberger, K.Q.: Simplifying graph convolutional networks. In: ICML (2019)
26. Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.S.: Attentional factorization machines: Learning the weight of feature interactions via attention networks. In: IJCAI (2017)
27. Xin, X., Chen, B., He, X., Wang, D., Ding, Y., Jose, J.: Cfm: convolutional factorization machines for context-aware recommendation. In: IJCAI (2019)
28. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: ICLR (2018)
29. Yu, J., Gao, M., Li, J., Yin, H., Liu, H.: Adaptive implicit friends identification over heterogeneous network for social recommendation. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management (2018)
30. Yuan, F., Xin, X., He, X., Guo, G., Zhang, W., Chua, T.S., Joemon, J.: fBGD: Learning embeddings from positive unlabeled data with BGD. In: UAI (2018)