Attribute-Aware Non-Linear Co-Embeddings of Graph Features

Ahmed Rashed
Information Systems and Machine Learning Lab, University of Hildesheim, Germany
ahmedrashed@ismll.uni-hildesheim.de

Josif Grabocka
Information Systems and Machine Learning Lab, University of Hildesheim, Germany
josif@ismll.uni-hildesheim.de

Lars Schmidt-Thieme
Information Systems and Machine Learning Lab, University of Hildesheim, Germany
schmidt-thieme@ismll.uni-hildesheim.de

ABSTRACT

In very sparse recommender data sets, attributes of users such as age, gender and home location and attributes of items such as, in the case of movies, genre, release year, and director can improve the recommendation accuracy, especially for users and items that have few ratings. While most recommendation models can be extended to take attributes of users and items into account, their architectures usually become more complicated. While attributes for items are often easy to be provided, attributes for users are often scarce for reasons of privacy or simply because they are not relevant to the operational process at hand. In this paper, we address these two problems for attribute-aware recommender systems by proposing a simple model that co-embeds users and items into a joint latent space in a similar way as a vanilla matrix factorization, but with non-linear latent features construction that seamlessly can ingest user or item attributes or both (GraphRec). To address the second problem, scarce attributes, the proposed model treats the user-item relation as a bipartite graph and constructs generic user and item attributes via the Laplacian of the user-item co-occurrence graph that requires no further external side information but the mere rating matrix. In experiments on three recommender datasets, we show that GraphRec significantly outperforms existing state-of-the-art attribute-aware and content-aware recommender systems even without using any side information.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; *Supervised learning by regression*; *Learning latent representations.*

KEYWORDS

Collaborative Filtering; Recommender Systems; Graph Embedding; Non-Linear Factorization Machines

ACM Reference Format:
Ahmed Rashed, Josif Grabocka, and Lars Schmidt-Thieme. 2019. Attribute-Aware Non-Linear Co-Embeddings of Graph Features . In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3298689.3346999

RecSys '19, September 16–20, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-6243-6/19/09…$15.00
https://doi.org/10.1145/3298689.3346999

1 INTRODUCTION

Recommender systems play an essential role in our daily lives and have been employed in various applications, including, but not limited to, online market places, search engines, and online media websites. The goal of recommender systems is to provide users with a personalized short list of items by filtering the massive amount of all available items. A widely used approach to generate such a list is to predict the ratings of all items with respect to a specific user and provide a ranked list of items.

Recommender systems are usually categorized into content filtering, and collaborative filtering (CF) approaches [8]. Although CF approaches such as matrix factorization (MF) provide a very competitive performance, they cannot directly leverage any additional external side features, and further modifications are required [6, 8, 18]. Such modifications often lead to overly complicated hybrid models that might not generalize again to the simple scenarios where external side information is scarce.

To tackle this challenge, we propose a simple non-linear co-embedding model (GraphRec) that extracts item and user latent features similar to the matrix factorization techniques, however, with non-linear embedding functions like the neural networks. Unlike the matrix factorization approaches, GraphRec can leverage any available side attributes seamlessly by utilizing their numerical vector representation similar to the factorization machines. To address the problem of scarce external side attributes, GraphRec utilizes the Laplacian information of the users-items co-occurrence graph as generic internal attributes which only require the mere rating matrix. Once employed to GraphRec, these features allow it to outperform attribute and content aware models without using any actual side features.

Our contributions can be summarized as follows :

- We introduce a simple non-linear co-embedding model GraphRec for rating prediction. GraphRec can be applied to diverse settings and can leverage additional attributes and content features by seamlessly utilizing their numerical vector representations.
- We introduce a set of internal graph-based features that are extracted from the Laplacian of the user-item co-occurrence graph to capture users' and items' rating profiles. Once integrated into GraphRec, these features improve the overall accuracy to the extent of outperforming multiple state-of-the-art attribute and content aware models.
- We evaluate the proposed model on three real-world recommender datasets for rating prediction. The results show that graph-based features can effectively replace the scarce external side features and they allow the GraphRec model to

outperform state-of-the-art matrix factorization, attribute-aware, and content-aware models without using any external side information.

2 RELATED WORK

Earlier collaborative filtering approaches relied mainly on neighborhood based methodologies [1, 4] and they were succeeded by the matrix factorization (MF) approaches [5, 7, 10] with their impressive performance [6]. Learning the expressive latent factors plays an important role in traditional MF techniques, and it is achieved by factorizing the rating matrix. However, despite their impressive performance, MF models usually cannot directly support additional sources of information such as the demographics of user and item attributes. To address this drawback, Steffen Rendle [11] proposed the factorization machines(FM) which capture the second order interaction between features and are able to leverage any additional information by the simple concatenation of all feature vectors. Subsequently, further hybrid models were introduced that improve the underlying feature extraction process by merging deep neural networks and MF techniques. An earlier example of such models is mSDA-CF [8] which is a combination between MF and marginalized stacked denoising autoencoders. Another more recent hybrid approach is ConvMF [6] which combines convolutional neural networks with MF to extract latent features from the items' textual descriptions. Recently, more powerful hybrids have also been introduced such as AutoSVD [18] and the Non-linear factorization machines (NFM) [3]. AutoSVD relies on contracted autoencoders for features extraction and an SVD model for rating predictions. NFM follows a different approach by stacking an FM layer and a fully connected one to capture the higher order non-linear interactions between features. Although these hybrid models provide better performance compared to the MF techniques, they tend to be more complicated and less generalizable across different scenarios especially when the additional information on which the model relies on are nonexistent such as the items' descriptions. Similar challenges are encountered in the graph embedding domain. To tackle it, multiple models have utilized the Laplacian information as internal features for gaining better performance [13, 15]. Using such information has also been studied in recommender systems [17], however, they are used as auxiliary features from an available item-item interaction graph but not for the main user-item graph.

In this paper, we aim to tackle these challenges and drawbacks simultaneously by proposing a model that is as simple as the MF models but with the power of non-linearity as neural networks. By merging those two approaches, the model can extract rich latent representations through the co-embedding of the item's and user's input features which capture the higher order non-linear interactions between them. To tackle the scarce side information, the model utilizes the Laplacian information as input features which significantly improves the performance to the extent of outperforming multiple state-of-the-art matrix factorization, attribute-aware, and content-aware models.

3 PROBLEM DEFINITION

In explicit rating prediction tasks, there exist a set of users $U := \{u_1, u_2, ..., u_N\}$ with profile attributes $C_u \in \mathbb{R}^{NxL}$, a set of items $I := \{i_1, i_2, ..., i_M\}$ with attributes and content information $C_i \in \mathbb{R}^{MxJ}$ and a sparse rating matrix $R \in \mathbb{R}^{NxM}$ that indicate user's preferences on items. The main goal of the prediction task is to fill the missing entries in the rating matrix that indicate the preferences of users on some items they did not interact with yet. In matrix factorization models, this is done through factorizing the original rating matrix using two separate latent matrices $P \in \mathbb{R}^{NxK}$ and $Q \in \mathbb{R}^{MxK}$ for user and items respectively. After the model is learned, the rating $\hat{r}_{ui}$ of user u on item i is computed using the dot-product of their latent feature vectors $\hat{r}_{ui} = P_u^T Q_i$. In this work we aim to follow a similar approach to the matrix factorization, however, we extend it to utilize any available side information through a dedicated non-linear embedding functions ϕ_U , ϕ_I that will extract a more expressive latent feature vectors P_u and Q_i by utilizing any side information through its numerical vector representation.

4 PROPOSED MODEL

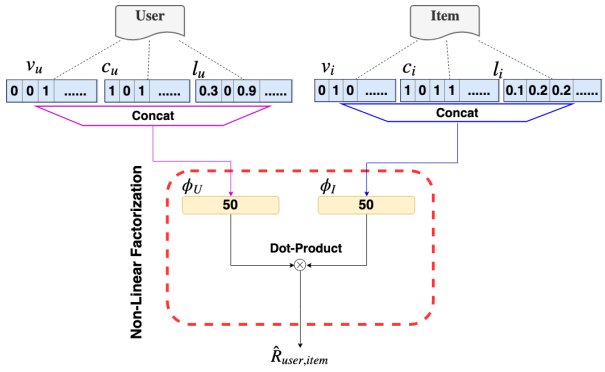


Figure 1: GraphRec Architecture

The proposed GraphRec model is divided into two main components. The first component is a non-linear factorization model that predicts user-item ratings by extracting user and item latent factors through the non-linear co-embeddings of their input features. The second component is a set of graph-based features that allow the model to learn expressive latent features especially when item and users side attributes are not available. Figure 1 illustrates the architecture of a one-layer GraphRec model and each component will be discussed in details in the following subsections.

4.1 Non-linear Co-Embeddings

Given the set of user $U := \{u_1, u_2, ..., u_N\}$ and the set of items $I := \{i_1, i_2, ..., i_M\}$, we define a latent embedding of the users' profiles as the function $P := \phi_U : \mathbb{R}^{N\times(N+L)} \rightarrow \mathbb{R}^{N\times K}$, which is inputted with a concatenation of user-specific features, concretely the indicator variable of the r id $v_u \in \{0, 1\}^N$, with $v_{u,n} = 1$, if $u = n$, and $v_{u,n} = 0$ otherwise; and the additional user profile features $C_u \in \mathbb{R}^{N\times L}$. Similarly, the latent item embedding is a function $Q := \phi_I : \mathbb{R}^{M\times(M+J)} \rightarrow \mathbb{R}^{M\times K}$, that uses the indicator variable of the item id $v_i \in \{0, 1\}^M$, with $v_{i,m} = 1$, if $i = m$, and $v_{i,m} = 0$ otherwise; and the additional item content features $C_i \in \mathbb{R}^{M\times J}$.

$$P_u = \phi_U(x_u; \theta_U), \qquad Q_i = \phi_I(x_i; \theta_I) \tag{1}$$

$$x_u = [v_u, c_u], \qquad x_i = [v_i, c_i] \tag{2}$$

The final output of the embeddings are the latent user features $P_u \in \mathbb{R}^{N \times K}$, and the latent item features $Q_i \in \mathbb{R}^{M \times K}$. Notice that the embedding functions ϕ_u and ϕ_i are nonlinear functions parameterized with θ_U and θ_I. Similar to the matrix factorization techniques, the rating of a specific user-item interaction can then be predicted by learning the higher order non-linear interaction between user and item latent features through the inner-product of the learned vectors

$$\hat{r}_{ui} = P_u^T Q_i. \tag{3}$$

An important difference between our model and the standard matrix factorization is that latent embedding vectors are not directly expressed as parameters. Instead, they are the activation values of neural network layers. Therefore, in contrast to the factorization case, our parameters are not P, Q, but the weights θ_U, θ_I of the neural networks ϕ_U, ϕ_I that output P, Q. All the network model's parameters are learned by minimizing a sparsely regularized squared error loss on the training set R_{train} of observed ratings:

$$\mathcal{L}(\Theta) = \sum_{(u,i) \in R_{\text{train}}} (r_{ui} - \hat{r}_{ui})^2 + \lambda_u ||P_u||^2 + \lambda_i ||Q_i||^2 \tag{4}$$

where $\lambda_u ||P_u||^2$ and $\lambda_i ||Q_i||^2$ are sparse-L2 (activity) regularization [2] components applied directly to the output of the embedding functions instead of applying dense L2 regularization to their weights θ_U and θ_I. These sparse L2 regularization components will induce a similar effect to the inverse of the frequency based regularization [9] such that only the weights that contribute to the output latent values will get penalized, hence, the more frequent the training instance, the more penalty it receives. These components are also suitable for sparse feature vectors since weights multiplied by zero inputs will receive no penalty. This effect can also be achieved using dropout, however, sparse-L2 gave a better performance in all experiments.

In all of our comparative experiments, we used a single-layered embedding function with CReLu [12] activation and sparse-L2 regularization because it was found to be the best performing setting. Comparisons between different settings for the embedding functions are discussed in Section 5.

4.2 Graph-Based Features

In some situations, user and item side information is scarce, hence, attribute-aware and content-aware models will under-perform. To address this problem, we need to define a set of internal features that can approximate the user and item profiles without relying on any external side information. Recent advances in graph embedding and node classification techniques [13, 15] have shown that decomposing the Laplacian matrix and using its vectors as node features, greatly increases the prediction accuracy because it implicitly encapsulates many useful properties of the interaction graph. Laplacian matrix of an interaction graph can be simply calculated only once by taking the difference between the degree and the adjacency square matrices $L = D - A$. However, to migrate and utilize such features in the recommender systems and user-item interaction graphs, some changes will be required. The users-items interaction graph is a weighted bipartite graph with two node sets U and I, hence, the degree and adjacency matrices are no longer square matrices. To solve this problem, we define a set of graph-based features that imitate the users' and items' Laplacian information as follows:

$$l_u = [d_u, a_u], \quad l_u \in \mathbb{R}^{M+1}, \quad d_u \in \mathbb{R}, \quad a_u \in \mathbb{R}^M \tag{5}$$

$$l_i = [d_i, a_i], \quad l_i \in \mathbb{R}^{N+1}, \quad d_i \in \mathbb{R}, \quad a_i \in \mathbb{R}^N \tag{6}$$

where l_u is the Laplacian feature vector for user u which is constructed by concatenating his weighted items interaction (adjacency) vector a_u and his frequency (degree) of rating d_u. Similarly we construct item i's Laplacian feature vector l_i by concatenating a_i and d_i. The frequency of ratings is calculated by counting the number of interactions instead of summing the interactions weights for simplicity. The adjacency vectors a_u and a_i are weighted vectors using the actual rate values of the interactions instead of being a binary encoded vectors of the implicit feedback.

Since the values of the adjacency vectors and the degrees have different scales, we scale all values to be between 0 and 1, such that a 0 value in the adjacency vector will indicate that the user or item have never been interacted with (missing value), while 1 means that the user or item have been interacted with and the maximum rate was given. The normalization of degrees is done by dividing over the maximum degree in the train set while adjacency vectors are normalized as follows:

$$a_{ui} = \begin{cases} \frac{(r_{ui} + \epsilon)}{\text{max\_rate} + \epsilon}, & \text{if } (u,i) \in R \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where max_rate is the maximum possible rate allowed in the dataset, while ϵ is a constant value for adjusting the rates scale to be > 0, hence, all missing entries will have 0 value and the whole vector can be treated as a sparse one. It is worth noting that all user and item Laplacian features can be computed only once using the training set.

To utilize the extracted Laplacian features in the proposed non-linear co-embedding model, we simply concatenate the features to the rest of the user and item feature vectors defined in Equation (2).

$$x_u = [v_u, c_u, l_u], \qquad x_i = [v_i, c_i, l_i] \tag{8}$$

5 EXPERIMENTS

We conduct multiple experiments that aim to answer the following research questions:

- **RQ1** Do non-linear activation functions help in learning rich latent representations?
- **RQ2** Which kind of regularization is best suited for the non-linear co-embeddings?
- **RQ3** How many hidden layers are needed for GraphRec to learn expressive latent features?
- **RQ4** How well does GraphRec perform in comparison with other state-of-the-art models for rating predictions?
- **RQ5** What is the impact of adding the graph-based features?

In the following subsections, we present our experimental settings followed by answering the research questions.

5.1 Datasets

In following sections, we used three publicly available real-world datasets:

(1) MovieLens 100K[1]: A well-known movie rating dataset that has been widely used for rating prediction, This version contains 100,000 ratings.
(2) MovieLens 1M: This is a bigger version of the MovieLens 100K that contains 1,000,209 ratings.
(3) Amazon Instant Videos (AIV) [6]: This dataset contains 135,188 ratings on Amazon instant videos, and it is extremely sparse compared with the other two datasets. Each item has a textual description as its external content-based features.

Detailed statistics of the datasets are shown in Table 1.

5.2 Experimental Protocol

For the MovieLens datasets, we followed the same experimental protocol employed in [8, 18]. We used 5-fold cross-validation for hyper-parameters tuning using different percentages (50% and 90%) of ratings for training. Finally, We used the average root mean squared error (RMSE) on the test set of 5 different runs as the evaluation metric.

On the other hand, for the AIV dataset, we employed the same experimental protocol introduced in the original paper [6]. The data is split into a training set (80%), a validation set (10%) and a test set (10%). For evaluation, this process is repeated five times, and the average test RMSE is reported.

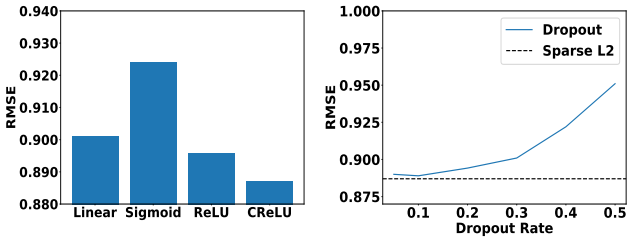
The optimal hyper-parameters have been estimated via grid search for GraphRec and other models. For GraphRec we tested the embedding dimensions of [2, 3, 5, 10, 20. 30, 40, 50, 70, 100], the learning rates of [0.001, 0.0005, 0.0003, 0.0002, 0.0001, 0.00005, 0.00003, 0.00002, 0.00001, 0.000005] and the regularization weights of [0.1, 0.09, 0.08, 0.07, 0.06, 0.05, 0.04, 0.03, 0.02, 0.01]. We use the Adam optimizer for training. More details about the selected hyper-parameters and the experiments reproducibility are discussed in the last subsection.

5.3 Linear vs Non-linear Co-Embeddings (RQ1)

In this experiment, we compare the performance of four different activation functions for the co-embedding component using the MovieLens 100k dataset, 50 latent factors and 90% of ratings for training.

Figure 2 (a) shows that the recent non-linear activation functions such as ReLU and CReLU [12], outperform the linear counterpart as they are able to capture more expressive latent features from the entities. Results also show that CReLU outperforms the traditional ReLU and Sigmoid activation functions as it can be activated in both positive and negative direction while maintaining the same degree of non-saturated non-linearity similar to ReLU activation.

[1]https://grouplens.org/datasets/movielens/



(a) Activation Functions (b) Drouput vs Sparse-L2

Figure 2: Performance comparison on MovieLens 100K: (a) Different activation functions; (b) Different dropout rates and the best Sparse L2 regularization

5.4 Dropout vs Sparse-L2 Regularization (RQ2)

Input vectors of the co-embedding functions are often very sparse, hence, a suitable regularization technique will be required that is immune to data sparsity. Two candidate techniques that fulfill this requirement are Dropout and the sparse-L2 regularization (activity regularization). Both techniques can be applied in the case of sparse input feature vectors, however, they work differently. Dropout creates an implicit ensemble inside the layers while the sparse-L2 regularization penalizes the weights of the layers in a sparse frequency-based fashion. To compare the two techniques, we applied them on MovieLens 100k using 90% of the ratings for training. Results in Figure 2(b) show that the sparse-L2 regularization has a very similar performance to dropout with small ratios (0.05 and 0.1), however, the sparse-L2 is preferable because it has a more stable performance with less RMSE fluctuations on the validation sets.

5.5 Impact of Co-Embedding Layers (RQ3)

In order to find the best number of layers for the co-embedding component, we compared the performance of different single-layered and double-layered architectures with varying sizes of hidden units. This experiment was conducted on the MovieLens 100k dataset using 90% of ratings for training.

Figure 3 shows that a single-layered architecture is sufficient to capture the higher order interaction between the input features and adding a further layer does not provide any lift.

5.6 Performance comparison with state-of-the-art models for rating prediction (RQ4)

In this experiment, we compare the performance of GraphRec with and without the graph-features on all three datasets against multiple state-of-the-art matrix factorization, attribute-aware and content-aware models.

5.6.1 *Baselines.*

(1) NMF [5]: Non-negative matrix factorization model for rating prediction.
(2) PMF [10]: Probabilistic matrix factorization model that only uses ratings for collaborative filtering.

Table 1: Datasets Statistics

Dataset	Items	Users	Ratings	Density (%)	User Features	Item Features
MovieLens 100k	1,682	943	100,000	6.30	Age, gender, and occupation	Genres and year
MovieLens 1M	3,952	6,040	1,000,209	4.46	Age, gender, and occupation	Genres and year
Amazon Instant Video	29,757	15,149	135,188	0.030	-	Text Description

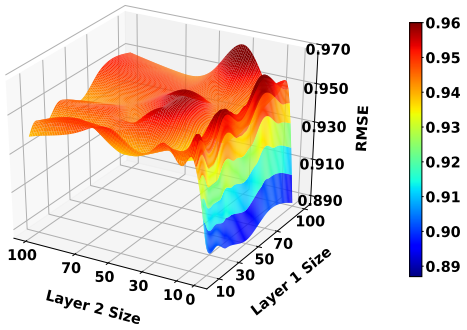


Figure 3: Performance comparison on MovieLens 100K dataset between different single-layered and double-layered architectures of the co-embedding component.

(3) SVD [7]: The popular singular value decomposition model for ratings prediction.
(4) SVD++ [7]: This is the improved version of SVD that utilizes the user's implicit feedback information.
(5) FM [11]: The traditional factorization machines model that utilizes the first and second order interaction between features for regression and classification problems.
(6) mSDA-CF [8]: This attribute-aware model is a combination between traditional matrix factorization with marginalized stacked denoising auto-encoders.
(7) CTR [14]: Collaborative Topic Regression for rating prediction that combines PMF with LDA for topic modeling. This model relies mainly on the textual description of items.
(8) CDL [16]: Collaborative Deep Learning model that utilizes the textual description of items similar to CTR and it relies on stacked denoising auto-encoders for text analysis.
(9) ConvMF [6]: A state-of-the-art convolutional matrix factorization model for rating prediction using items' textual descriptions.
(10) ConvMF++ [6]: A pre-trained version of ConvMF that uses pre-trained word embeddings.
(11) AutoSVD [18]: A state-of-the-art attribute-aware recommender model that combines SVD with contractive auto encoders.
(12) AutoSVD++ [18]: Similar to AutoSVD, this version uses SVD++ instead of SVD to ultilze the implicit feedback information.
(13) NFM [3]: State-of-the-art non-linear factorization machines that capture higher order non-linear interactions between input features by stacking a multi-layered feed-forward neural network on top of factorization machines.

For all models except NFM and FM, we directly report their results from [6, 8, 18] for the same experimental protocol and data splits. While for NFM and FM we used grid-search for parameter tuning as no results were reported on the same datasets.

On the AIV dataset, we only used CTR, CDL, ConvMF and ConvMF++ for comparison as content-aware models because they are able to leverage the items' textual descriptions. Hyper-parameter details of NFM and FM are discussed in the experiments reproducibility section.

Model	RMSE	
	90%	50%
Without External Side Features		
NMF [5]	0.958	0.997
PMF [10]	0.952	0.977
SVD [7]	0.911	0.936
SVD++ [7]	0.913	0.938
FM [11]	0.909	0.935
NFM [3]	0.910	0.935
GraphRec (w/o Graph Feat.)	0.898	0.932
GraphRec (w/ Graph Feat.)	**0.887**	**0.918**
With External Side Features		
FM [11]	0.903	0.925
mSDA-CF [8]	-	0.931
AutoSVD [18]	0.901	0.925
AutoSVD++ [18]	0.904	0.926
GraphRec (w/o Graph Feat.)	0.899	0.923
GraphRec (w/ Graph Feat.)	**0.883**	**0.911**

Table 2: Average RMSE on MovieLens 100k using 90% and 50% training data percentages.

5.6.2 Results. Results in tables 2, 3, and 4, show that GraphRec without external or graph features consistently outperforms all other models that similarly do not utilize any external information. All of these results are statistically significant compared to NFM with a p-value less than 0.01 using a paired t-test. Once the graph-based features are added to GraphRec in MovieLens 100k and 1M, it outperforms almost all models including multiple state-of-the-art attribute-based and content-based models even without yet

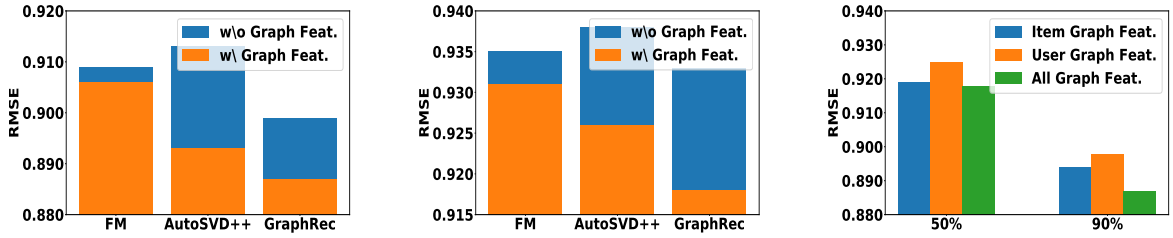


Figure 4: Experimental analysis of the graph-based features on the MovieLens 100K: (a) Impact on different models using 90% of ratings; (b) Impact on different models using 50% of ratings; (c) Impact of different graph features on GraphRec's performance

Model	RMSE	
	90%	50%
Without External Side Features		
NMF [5]	0.915	0.927
PMF [10]	0.883	0.890
SVD [7]	0.876	0.889
SVD++ [7]	0.855	0.884
FM [11]	0.856	0.873
NFM [3]	0.858	0.881
GraphRec (w/o Graph Feat.)	0.845	0.869
GraphRec (w/ Graph Feat.)	**0.843**	**0.862**
With External Side Features		
FM [11]	0.855	0.870
mSDA-CF [8]	-	0.861
ConvMF [6]	-	0.890
AutoSVD [18]	0.864	0.877
AutoSVD++ [18]	0.848	0.875
GraphRec (w/o Graph Feat.)	0.845	0.864
GraphRec (w/ Graph Feat.)	**0.842**	**0.860**

Table 3: Average RMSE on MovieLens 1M using 90% and 50% training data percentages.

using the available external features. This shows that graph-based features can be used as a substitute for the external features when they are not available. Adding the external features to GraphRec improves its accuracy even more which also indicates that graph-based and the external side features capture different aspects of the user's and item's profiles.

On the other hand, results on the AIV dataset show that the graph-based features did not improve the accuracy of GraphRec, however, the two versions of GraphRec still outperform all other models. This negative effect can be contributed to the fact that the dataset is extremely sparse compared to the other two datasets and the Laplacian feature vectors will be almost all zeros, hence, it does not contain much helpful information about the entities.

It is worth noting that we only report the accuracy without using any external features on the AIV dataset because using the binary vector of word occurrences did not improve the accuracy.

Model	RMSE
Without External Side Features	
PMF [10]	1.412
FM [11]	1.236
NFM [3]	1.093
GraphRec (w/o Graph Feat.)	**1.037**
GraphRec (w/ Graph Feat.)	1.085
With External Side Features	
CTR [14]	1.550
CDL [16]	1.359
ConvMF [6]	1.134
ConvMF++ [6]	1.128

Table 4: Average test RMSE on AIV using 80% of the data for training, 10% for validation and 10% for testing.

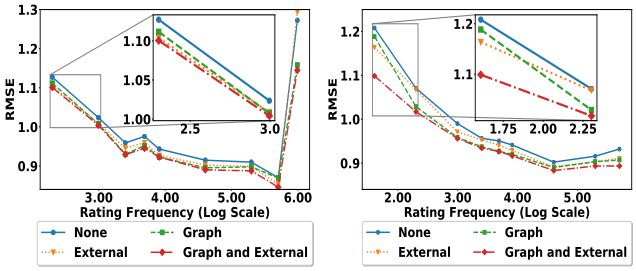
5.7 Impact of Graph-Based Features (RQ5)

To analyze the impact of graph-based features, we conducted three different experiments using MovieLens 100K dataset with 50% and 90% of ratings for training. First, we checked if other models can also leverage the graph-based features for further improvement gains. To do so, we added the graph-based features to FM and AutoSVD++ models then we compared their performance with and without such features. Results in Figures 4 (a) and (b) show that FM, AutoSVD++ and GraphRec can benefit from the graph-based features. However, the improvements for AutoSVD++ and GraphRec are much higher than for the FM model that unlike the first two models, uses a joint latent dimension for all features. These results also show that the proposed graph-based features are a good substitute for the side features when they are missing.

In the second experiment, we compared the impact of users and items graph features separately. Figure 4 (c) shows that item features have a slightly higher impact on the accuracy, however, merging both feature types maximize the performance.

Finally, we analyzed the impact of the different feature combinations over items and users that have different rating frequencies using the 50% data split. Figure 5 shows that the graph-based features have a smaller effect on items and users that have too few ratings (<10) compared to the external side features, however, their impact increases with increasing the number of available ratings. Merging both feature types gives the best performance which indicates that both feature types encapsulate a different kind of information.

Interestingly, results in Figure 5 (a) show that the external side features are sensitive to noisy users that have many ratings such as bots and group accounts, while the graph-based features, on the other hand, are less sensitive to such noisy users because they capture the users' implicit preferences and frequencies of rating.



(a) Users (b) Items

Figure 5: Impact of graph and external side features on the model performance across different rating frequencies.

5.8 Reproducibility of the Experiments

The source code of GraphRec is available at our GitHub repository[2]. The code for FM is available at the LibFM home page[3]. The code for NFM is available at the authors' GitHub repository[4].

Regarding the hyper-parameters we used the followings during the experiments:

- GraphRec (w/o Graph Feat.): The best found hyper-parameters are $\lambda_u = 0.08$, $\lambda_i = 0.06$, embedding sizes = [50, 40], learning rate = 0.0002 and number of iterations = [600, 550] for the MovieLens 100K 50% and 90% splits respectively; $\lambda_u = 0.08$, $\lambda_i = 0.06$, embedding size = 40, learning rate = 0.0002 and number of iterations = [300, 510] for the MovieLens 1M 50% and 90% splits; and $\lambda_u = 0.05$, $\lambda_i = 0.05$, embedding size = 3, learning rate = 0.0002 and number of iterations = 100 for the AIV dataset.
- GraphRec (w/ Graph Feat.): The best hyper-parameters are $\lambda_u = 0.05$, $\lambda_i = 0.02$, embedding sizes = [10, 50], learning rate = 0.00003 and number of iterations = [559, 195] for the MovieLens 100K 50% and 90% splits; $\lambda_u = 0.05$, $\lambda_i = 0.06$, embedding size = 50, learning rate = 0.000005 and number of iterations = 545 for the MovieLens 1M 50% and 90% splits; and $\lambda_u = 0.05$, $\lambda_i = 0.05$, embedding size = 3, learning rate = 0.00003 and number of iterations = 450 for the AIV dataset.
- GraphRec (w/ External Feat. & w/o Graph Feat.): The best found hyper-parameters are $\lambda_u = 0.08$, $\lambda_i = 0.06$, embedding size = 40, learning rate = 0.0002 and number of iterations = [400, 310] for the MovieLens 100K 50% and 90% splits respectively; and $\lambda_u = 0.08$, $\lambda_i = 0.06$, embedding size = 40, learning rate = 0.0002 and number of iterations = [320, 330] for the MovieLens 1M 50% and 90% splits.
- GraphRec (w/ External Feat. & w/ Graph Feat.): The hyper-parameters are $\lambda_u = 0.05$, $\lambda_i = 0.02$, embedding sizes = [20, 50], learning rate = 0.00003 and number of iterations = [321, 195] for the MovieLens 100K 50% and 90% splits; $\lambda_u = 0.05$, $\lambda_i = 0.06$, embedding size = 50, learning rate = 0.000005 and number of iterations = 545 for the MovieLens 1M 50% and 90% splits.
- FM (w/o External Feat.): The hyper-parameters are $\lambda = 0.1$, embedding size = 5, learning rate = 0.001 and number of iterations = 600 for the MovieLens 100K 50% and 90% splits; $\lambda = 0.05$, embedding size = 5, learning rate = 0.001 and number of iterations = [500, 800] for the MovieLens 1M 50% and 90% splits; and $\lambda = 0.5$, embedding size = 10, learning rate = 0.005 and number of iterations = 270 for the AIV dataset.
- FM (w/ External Feat.): The hyper-parameters are $\lambda = 0.2$, embedding size = 4, learning rate = 0.0002 and number of iterations = 300 for the MovieLens 100K 50% and 90% splits; and $\lambda = 0.05$, embedding size = 5, learning rate = 0.001 and number of iterations = 270 for the MovieLens 1M 50% and 90% splits
- NFM:The hyper-parameters are hidden factors = [3, 5], layer sizes=[3, 5], dropout rates of [0.8, 0.5], learn rate = 0.02 and number of epochs = 400 for the MovieLens 100K 50% and 90% splits; hidden factors = 10, layer size=10, dropout rates of [0.8, 0.5], learn rate = 0.02 and number of epochs = 250 for the MovieLens 1M 50% and 90% splits; and hidden factors = 5, layer size=5, dropout rates of [0.8, 0.5], learn rate = 0.02 and number of epochs = 250 for the AIV dataset.

[2]https://github.com/ahmedrashed-ml/GraphRec
[3]http://www.libfm.org/
[4]https://github.com/hexiangnan/neural_factorization_machine

6 CONCLUSION AND FUTURE WORK

In this paper, we propose GraphRec a simple non-linear co-embedding model for rating prediction that extracts rich latent representations for users and items. In situations where external attributes are scarce, GraphRec utilizes the Laplacian of the user-item interaction graph to extract generic graph-based attributes. GraphRec seamlessly leverages any available external attributes without any further modifications to the model structure. Experimental results on three real-world recommender datasets show that GraphRec outperforms state-of-the-art attribute-aware and content-aware models with the mere graph-based features.

In future works, we plan to extend this model for context and session-based prediction. We also plan to adapt and evaluate GraphRec for item ranking and implicit feedback relations.

ACKNOWLEDGMENTS

This work is co-funded by the industry project "Data-driven Mobility Services" of ISMLL and Volkswagen Financial Services.(https://www.ismll.uni-hildesheim.de/projekte/dna_en.html)

REFERENCES

[1] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.

[2] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Geoffrey Gordon, David Dunson, and Miroslav DudÃŋk (Eds.), Vol. 15. PMLR, Fort Lauderdale, FL, USA, 315–323. http://proceedings.mlr.press/v15/glorot11a.html

[3] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 355–364.

[4] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999*. Association for Computing Machinery, Inc, 230–237.

[5] Patrik O Hoyer. 2004. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research* 5, Nov (2004), 1457–1469.

[6] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 233–240.

[7] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.

[8] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 811–820.

[9] Aditya Krishna Menon and Charles Elkan. 2010. A log-linear model with latent features for dyadic prediction. In *2010 IEEE International Conference on Data Mining*. IEEE, 364–373.

[10] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[11] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[12] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. 2016. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *International Conference on Machine Learning*. 2217–2225.

[13] N Kipf Thomas and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*.

[14] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 448–456.

[15] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1225–1234.

[16] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 1235–1244.

[17] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 974–983.

[18] Shuai Zhang, Lina Yao, and Xiwei Xu. 2017. Autosvd++: An efficient hybrid collaborative filtering model via contractive auto-encoders. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 957–960.