How much do we know about the User-Item Matrix?: Deep Feature Extraction for Recommendation

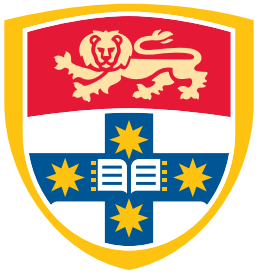
TAEJUN LIM
SID: [REDACTION]

Supervisor: Dr. Caren Han & Dr. Josiah Poon

A thesis submitted in fulfilment of
the requirements for the degree of
Master of Philosophy (Science)

School of Computer Science
The University of Sydney
Australia

3 November 2022



THE UNIVERSITY OF SYDNEY

Student Plagiarism: Compliance Statement

I certify that:

I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure;

I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to the University commencing proceedings against me for potential student misconduct under Chapter 8 of the University of Sydney By-Law 1999 (as amended);

This Work is substantially my own, and to the extent that any part of this Work is not my own I have indicated that it is not my own by Acknowledging the Source of that part or those parts of the Work.

Name: Taejun Lim

Signature: [REDACTION] **Date**: 20/06/2022

Abstract

Collaborative filtering-based recommender systems typically operate on a high-dimensional sparse user-item matrix. Matrix completion is one of the most common formulations where rows and columns represent users and items, and predicting user's ratings in items corresponds to filling in the missing entries of the matrix. In practice, it is a very challenging task to predict one's interest based on millions of other users having each seen a small subset of thousands of items.

We considered how to extract the key features of users and items in the rating matrix to capture their features in a low-dimensional vector and how to create embeddings that well represent the characteristics of users and items by exploring what kind of user/item information to use in the matrix. However, recent studies have focused on utilising side information, such as user's age or movie's genre, but it is not always available and is hard to extract. More importantly, there has been no recent research on how to efficiently extract the important latent features from a sparse data matrix with no side information (1st problem). The next (2nd) problem is that most matrix completion techniques have mainly focused on semantic similarity between users and items with data structure transformation from a rating matrix to a user/item similarity matrix or a graph, neglecting the position of each element (user, item and rating) in the matrix. However, we think that a position is one of the fundamental points in matrix completion, since a specific point to be filled is presented based on the positions of its row and column in the matrix.

In order to address the first (1st) problem, we aim to generalise and represent a high-dimensional sparse user-item matrix entry into a low-dimensional space with a small number of important features, and propose a Global-Local Kernel-based matrix completion framework, named GLocal-K, which is divided into two major stages. First, we pre-train an autoencoder with the local kernelised weight matrix, which transforms the data from one space into the feature space by using a 2d-RBF kernel. Then, the pre-trained autoencoder is fine-tuned with the rating matrix, produced by a convolution-based global kernel, which captures the characteristics of each item. GLocal-K outperforms the state-of-the-art baselines on three collaborative filtering benchmarks. However, it cannot show its superior feature extraction ability when the data is very large or too extremely sparse.

For the aforementioned second (2nd) problem and the GLocal-K's limitation, we propose a novel position-enhanced user/item representation training model for recommendation, SUPER-Rec. We first

capture the rating position in a matrix using relative positional rating encoding and store the position-enhanced rating information and its user-item relationship to a fixed dimension of embedding that is not affected by the matrix size. Then, we apply the trained position-enhanced user and item representations to the simplest traditional machine learning models to highlight the pure novelty of the SUPER-Rec representation. We contribute to the first formal introduction and quantitative analysis of the position-enhanced user/item representation in the recommendation domain and produce a principled discussion about SUPER-Rec with the incredibly excellent RMSE/MAE/NDCG/AUC results (i.e., both rating and ranking prediction accuracy) by an enormous margin compared with various state-of-the-art matrix completion models on both explicit and implicit feedback datasets. For example, SUPER-Rec showed the 28.2% RMSE error decrease in ML-1M compared to the best baseline, while the error decrease by 0.3% to 4.1% was prevalent among all the baselines.

Acknowledgements

It is a genuine pleasure to express my deep sense of thanks and gratitude to my supervisors, Dr. Caren Han and Dr. Josiah Poon, University of Sydney. Their constant encouragement, keen interest and above all the overwhelming attitude to help the students had been solely and mainly responsible for completing my dissertation. Their timely support, meticulous scrutiny and scholarly advice have helped me to greatly accomplish my task.

Specially, I would like to thank Dr. Caren Han. It was like fate to get to know her for the first time, and I am very fortunate to be able to do research as a student of the professor. She took the best care of me so that I sometimes doubted whether I could receive this much attention and love, and I will never forget this tremendous grace for the rest of my life. She gave me not only knowledge related to research, but also wisdom for life and how to properly relate to others, and all of these things were applied in my life as well as my studies so that I was able to grow up as a better person. Had it not been for her, I would have had to live without knowing all these things and as a person who has lots of flaws. I am always very grateful for her unconditional love and care that never asks for anything in return, and I never take any of these for granted. Because I believe that doing well makes her shine, I promise that I will devote myself diligently every day, always thinking about the relationship between problems and solutions, and causes and effects when doing research. Once again, I express my sincere thanks to her.

Personally, I would like to say thank you to my parents for their full support. Thanks to them always looking at me and giving me advice from an objective point of view, I was able to live my life in the right direction. It is very vivid that they smiled brightly at me and greeted me with a happy voice, and I miss them so much. I think the only thing I can do for them is to do my best in my position, so I will learn steadily and enthusiastically under my excellent supervisor, Dr. Caren Han.

I would like to say thank you to David. As he treats me kindly and warmly, I feel like I have gained a new strong support. I also learned from him how to thoroughly and accurately identify problems and come up with solutions in research. Overall, I feel he is very similar to my father. But since my father is my role model, I also want to be a person like him in the future.

Lastly, I thank my USYDNLP group friends. I would like to tell them that I was very grateful for being friendly with me from the time we first met until now, and being attentive to me, even though my English is not good. I hope that we will continue to have a good relationship in the future.

CONTENTS

List of Figures

List of Tables

CHAPTER 1

Introduction

Recommender systems aim to filter useful information and contents of user's potential interests and provide users with the most attractive and relevant items in the era of big data and thus have achieved significant success in social media and e-commerce. Among different recommendation approaches, Collaborative Filtering(CF) methods aim to discover the similarities in the user's past behaviour and make predictions to the user based on a similar preference with other users. To achieve the goal of CF, Matrix Completion(MC) is one of the most common formulations that uses a user-item rating matrix where rows and columns represent users and items, and predicts user's interactions (ratings or actions) in items corresponding to filling in the missing entries. (Candès and Recht, 2009; Bobadilla et al., 2013)

In practice, the matrix used for collaborative filtering is extremely sparse since it has ratings for only a limited number of user-item pairs. Traditional recommender systems focus on generalising sparsely observed matrix entries to a low dimensional feature space by using an autoencoder(AE) (Zhang et al., 2020). AEs would help the system better understand users and items by learning the non-linear user-item relationship efficiently, and encoding complex abstractions into data representations. GC-MC (Berg et al., 2018) proposed a graph-based AE framework for matrix completion, which produces latent features of user and item nodes through a form of message passing on the bipartite interaction graph. These latent user and item representations are used to reconstruct the rating links via a bilinear decoder. Such link prediction with a bipartite graph extends the model with structural and external side information. Recent studies (Rashed et al., 2019; Strahl et al., 2020; Ugla et al., 2020) focused on utilising side information such as opinion information or attributes of users. However, in most real-world settings (e.g., platforms and websites), there is no (or insufficient) side information available about users.

Instead of considering side information, we focus on improving the feature extraction performance for a high-dimensional user-item rating matrix into a low-dimensional latent feature space by applying two types of kernels, one (local kernel) of which is known to give optimal separating surfaces for the data

transformation and the other (global kernel) of which is from convolutional neural network (CNN) architectures. We propose a Global-Local Kernel-based matrix completion framework, called GLocal-K, which includes two stages: 1) pre-training the auto-encoder using a local kernelised weight matrix, and 2) fine-tuning with the global kernel-based rating matrix. We demonstrate the RMSE results on three benchmark datasets under the low-resource setting where no side information is available.

Moreover, existing MC-based recommendation models generally factorise the user-item rating matrix into two classes of latent features (embeddings) for users and items, respectively. There are different types of user/item embedding-based models such as graph-based, feature-based and inductive-based embeddings. They commonly tried to transform or compress the matrix structure to learn the latent features of users and items. However, those studies missed one of the fundamental points in the MC-based recommendation models. Predicting users' interest in items corresponds to filling in the specific points of missing entries, which can be presented based on the positions of row and column in the matrix. We consider that for accurate matrix completion we should recognise which user and item the target entry to be predicted is associated with and where the user and item is located in the row and column of the matrix. So, we focus on exploring the best way to capture and apply the location/position information in the matrix. With this in mind, we propose a position-enhanced user/item representation learning model for recommender systems, called SUPER-Rec, which covers the surrounding neighbour item information (i.e. positions and ratings). The SUPER-Rec recommendation consists of 3 main stages: The first stage, User-Item Matrix Positioning, defines the position-fixed surrounding item context and forms the training corpus for SUPER-Rec. Then, SUPER-Rec Training is conducted training a decoupled SUPER-Rec item representation utilising the fixed surrounding item position correlated with user feedback taste. The final stage, Matrix Completion with SUPER-Rec, conducts the recommendation based on the trained SUPER-Rec item representation and its derived user representation.

1.1 Contribution

In order to solve the data sparsity problem, which is the most common problem in CF and the problem that there is no recent research that effectively extracts features in a general situation given only sparse data, we propose a Global-Local Kernel-based matrix completion framework, named GLocal-K. The main contributions of the GLocal-K are summarised as below:

- We introduce a global and local kernel-based autoencoder model (GLocal-K), which mainly pays attention to extract the latent features of users and items.
- We propose a new way to integrate pre-training and fine-tuning for recommender systems.
- Without using any extra information, our GLocal-K achieves the smallest RMSEs on three widely-used benchmarks, even beating models augmented by side information.

Furthermore, in order to address the fundamental problem in matrix completion that has not been properly addressed and solved, we focus on the most important essence of a matrix: position/location, and then propose a position-enhanced user/item representation learning model for recommender systems, called SUPER-Rec. The main contributions of the SUPER-Rec are summarised as below:

- We introduce a new position-enhanced user/item representation for recommendation (SUPER-Rec) that leverages the positions and ratings of surrounding neighbour items.
- We propose a stand-alone and generally applicable position-enhanced user/item embedding model, which can produce the outstanding rating prediction results on most recommendation benchmark datasets (e.g. high or low density; small or large user/item/rating size; explicit or implicit user's action).
- The SUPER-Rec achieves the best RMSE/MAE/NDCG/AUC results by a significantly large margin compared with various state-of-the-art matrix completion models on eight widely-used recommendation benchmark datasets. The pretrained SUPER-Rec embedding for each dataset will be released to the public and let researchers/developers use it as an input embedding for their deeper and complex recommendation models.

1.2 Thesis Structure

The dissertation deals with collaborative filtering tasks on extremely sparse datasets using matrix completion techniques, but without additional side information and data structure transformation (e.g. similarity matrix, graph). We first introduce a global and local kernel-based matrix completion framework to efficiently extract the latent features of users and items from the user-item matrix with no side information, and present a position-enhanced user/item representation learning model for recommender systems to be able to represent all entries in the user-item matrix based the users given and the items rated. Next, we describe the model architectures and the role and principle of components in the proposed models. Then, we discuss various model evaluations to validate the recommendation performance

and explore the additional model capability followed by the objective and contributions (e.g. robustness under the extremely low resources, wide applicability to any size and sparsity of explicit rating/implicit action(click)-based data).

Chapter 2 provides the background of the collaborative filtering-based recommendation approaches by sorting them based on whether neural networks are used and then which type of architecture the model is. Besides, it also discusses about the feature extraction techniques and representation learning approaches in collaborative filtering.

Chapter 3 gives an overview of Global-Local Kernel-based matrix completion framework for recommender systems (GLocal-K) along with the in-depth explanation of local and global kernels in two-stage training process, and provides an overview of evaluation setup by introducing the benchmark datasets, baselines and implementation details, and lastly overviews the overall performance evaluation, cold-start recommendation, various component effectiveness analysis and hyperparameter test results in detail.

Chapter 4 gives an overview of SUrrounding Position-Enhanced Representation for Recommendation (SUPER-Rec), consisting of three main stages: user-item matrix positioning for training item corpus formation; SUPER-Rec training based on the surrounding item position and user feedback taste; matrix completion with SUPER-Rec for recommendation, and provides an overview of evaluation setup, including both explicit and implicit feedback datasets, baselines, implementation details and evaluation metrics for rating and ranking prediction accuracy, and lastly overviews the performance evaluation on explicit, implicit and large-scale datasets and with other simple ML-based classification models, SUPER-Rec training variants testing, sparse training data testing and various embedding representation analysis for exploring the relationship between the data statistics and the optimal embedding dimension.

Chapter 5 concludes this dissertation with observations from evaluation result and analysis. The future work of this thesis is carried out at last.

CHAPTER 2

Literature Review

This chapter provides an overview of earlier and recent trends in collaborative filtering-based recommender systems and feature extraction and representation learning approaches in collaborative filtering. Firstly, we discuss about collaborative filtering-based recommender systems in Section 2.1, where the systems are sorted by whether neural networks are used, and then are introduced based on general neural collaborative filtering models and particularly autoencoder- and graph-based recommender models. Then, we discuss about feature extraction approaches in collaborative filtering in Section 2.2 and focus on kernelised feature extraction techniques to deal with the sparse/large matrix issue in recommender systems. Lastly, we discuss about representation learning approaches in collaborative filtering in Section 2.3, where we overview the specific item and user representation approaches separately.

2.1 Collaborative filtering based recommender systems

Collaborative filtering (CF) is one of the most widely used techniques in recommender systems. CF is based on the assumption that if a user A and user B show similar rating patterns, they will reveal similar behaviors on other items. CF techniques focus on predicting the interests of a user by collecting preferences from a large number of users. In practice, CF-based recommendation has proved a great success in recommender systems, and has applied to a wide range of applications such as e-commerce, information retrieval and so on. Nevertheless, there are several challenges for CF-based tasks. Data used in CF algorithms is extremely sparse where ratings are derived from a very limited number of user-item pairs. It is also required to have the capacity to scale the increasing numbers of users and items as well as to prevent other problems such as cold start and grey sheep, that indicates users whose preferences always agree or disagree with any group of users. It would be worthy to explore how widespread CF models are designed to come up with a solution for the fundamental problems of CF. This section provides a brief overview of existing CF-based recommender systems, largely classifying them based

on their model architectures/types. We demonstrate the introduced CF-based neural recommendation models via a distinct characteristic-based breakdown approach in Figure 2.7.

2.1.1 Traditional collaborative filtering models

Early CF directly uses rating data to calculate the similarity between users (user-based CF) or items (item-based CF), and then makes predictions about the preference of users to items (Yang et al., 2016; Breese et al., 1998; Delgado and Ishii, 1999; Nakamura and Abe, 1998).

User-based CF calculates the similarities between the target user and all the other users, and selects the users with high similarities as the neighbour users. Then, it calculates the weighted average using neighbour users' ratings as weights for a specific item to predict the target user's rating. The system can also predict the rankings of all items for the target user based on the predicted ratings. As is expected, item-based CF calculates the similarities among items. It is basically assumed that items with similar ratings will have common characteristics. Neighbour items that are chosen based on similarity scores are used to calculate the predicted ratings of a target item by calculating the weighted average. In addition, there are several methods for similarity measure: 1) cosine similarity that measures the cosine value between two user/item vectors within the range of 0-1; 2) pearson correlation coefficient (PCC) measures the degree of linear correlation between two variables. Different from the cosine measure, PCC selects co-rated items or users to calculate similarities.

	User 1	User 2	User 3	User 4
Item 1	?	4	3	2
Item 2	4	3	5	1
Item 3	1	5	4	2
Item 4	3	1	3	3
Item 5	3	5	3	2

1) Select the neighbour items.
2) Calculate the weighted average.

	User 1	User 2	User 3	User 4
Item 1	2	4	3	2
Item 2	4	3	5	4
Item 3	1	5	4	2
Item 4	3	2	3	5
Item 5	3	5	3	2

FIGURE 2.1. An example of item-based CF recommendation process. First, select the neighbour items (i.e. Item 3, 5) based on the similarity measure (but here it is skipped) and predict the target item rating (i.e. Item 1), given user 1, by calculating the weighted average.

Furthermore, matrix factorisation (MF) techniques are widely adopted to implement CF-based recommendation tasks, in which a user-item rating matrix is decomposed into two low-rank matrices, the user feature and item feature matrices, respectively (Koren et al., 2009). For example, when a set of users

and items are denoted by U and I, R is the rating matrix of size $|U|\times|I|$. The purpose of MF is to find two matrices P and Q with L latent features as shown in Eq. 2.1:

$$\tilde{R} = P \times Q \approx R \tag{2.1}$$

The user feature vector p_u indicates how much user u is interested in each feature, and the item feature vector q_i measures the degree of each feature for item i. Based on these two vectors, the rating of user u on item i can be calculated by Eq. 2.2:

$$r_{u,i} = p_u^T q_i \tag{2.2}$$

An example of MF for CF-based recommendation is shown in Fig. 2.2, where the predicted (user-item rating) matrix $\tilde{R}$ is completed by multiplying two latent matrices P and Q with five latent features, and is compared with the original rating matrix R of five users and four items.

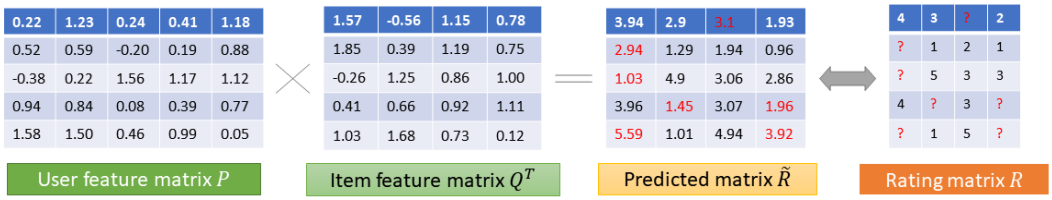


FIGURE 2.2. Matrix factorisation example for CF-based recommendation, in which both users and items are represented in a same size of feature vectors ($|p_u| = |q_i|$), and the ratings ($\tilde{R}$) of users for items are predicted by multiplying the user (P) and item (Q) feature matrices.

However, the limitations of the linear nature of MF are revealed when dealing with large and complex data that requires a thorough understanding ability, so neural network-based MF models are introduced.

2.1.2 Neural collaborative filtering models

While deep learning has advanced in a variety of fields including computer vision, speech recognition and natural language processing, it has also brought new changes to recommender systems. The application of deep learning in recommender systems has a strong advantage in training the representation of large-scale and complex data from its latent feature capturing ability.

The general objective of representation learning in CF is to learn user and item embedding matrices. (Salakhutdinov et al., 2007) firstly introduced to model users and items from a user-item rating matrix, using Restricted Boltzmann Machines (RBM) for neural CF. It is expected to show how a class of two-layer undirected graph models (RBMs) can be used to model tabular data (e.g. user-item rating matrix).

DCF (Li et al., 2015) proposes a general deep architecture that incorporates probabilistic MF and marginalized denoising auto-encoders (mDA) to learn effective latent representations via deep learning. Similarly, SDAE (Wang et al., 2015) proposes to integrate stacked denoising auto-encoders (SDA) and collaborative topic regression (CTR). However, both are hybrid recommender models, exploiting auxiliary information such as user profile and item content information with the CF-based approach. In addition, while SDAE (Wang et al., 2015) requires a large number of hyperparameters for training, using an expectation-maximisation (EM) algorithm, DCF (Li et al., 2015) computes the parameters in a closed form and is thus highly efficient and scalable, however they are close with each other in terms of the architecture.

Besides, CDAE (Wu et al., 2016) also adopts the idea of denoising auto-encoders (DAE) for top-N recommendation, which learns the correlations between interacted items on a set of corrupted user preference, containing information about whether an item is preferred or not. Similar to the standard DAE, the model proposed by (Wu et al., 2016) is represented as one hidden layer neural architecture, but the key difference is that the input layer includes a user latent vector as well as the latent vectors of the items observed by the user.

FISM (Kabbur et al., 2013) takes advantage of both latent factor and neighborhood approaches. It learns an item-item similarity matrix as the product of two low-dimensional latent factor matrices, and the recommendation for a specific unobserved item is performed by aggregating the products of two different item latent vectors, where one has been rated by a user and the other has never observed. The factored representation of the item-item similarity matrix is estimated via a structural equation modeling approach, which leads to better estimation quality as the number of factors increases. However, in practice, simultaneously integrating different historical items into one user behavior was not effective, as they had different contributions to the preference of a user.

NAIS (He et al., 2018) proposes a neural item similarity model, which can differentiate which interacted items are more important in the historical behavior of a user. It was built on the previous work (Kabbur et al., 2013), while preserving its high efficiency but making a stronger representation power. It is

achieved by adopting the recently advanced representation learning approach, the attention mechanism. However, the standard attention network does not work well due to the large variance on the history lengths of users (Bahdanau et al., 2014) and (Chen et al., 2017). To address this problem, it suggests a variant of the softmax function, that reduces the punishment on the attention weights of active users with a long history and accordingly decreases the variance of attention weights.

DeepICF (Xue et al., 2019) presents a neural item-based CF model, which exploits nonlinear neural networks to consider the interaction among all item pairs, thereby going beyond second-order interaction modeling. It aims to effectively capture higher-order item relations. It first models second-order item interactions through an element-wise product on each item embedding pair in the low-level of the neural architecture such as (Kabbur et al., 2013) and (He et al., 2018). In second-order interaction modeling, there are two main methods. One method is to use the same weight to combine pairwise item interactions and the other method is to use an attention mechanism to differentiate the importance of interacted item pairs. Inspired by (He and Chua, 2017), it then stacks a multi-layer perceptron (MLP) to learn higher-order item relations in a nonlinear way.

2.1.3 Autoencoder-based collaborative filtering models

AutoRec (Sedhain et al., 2015) introduces an autoencoder (AE)-based model for CF. As the proposed item-based AE model takes the rating records of each item from all users as input, it learns each item latent representation with an encoder network, and then feeds the learned item representations into a decoder network to predict missing ratings. Note that it can also be designed as a user-based AE by treating the historical records of each user as input.

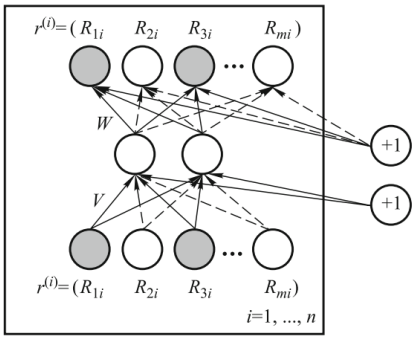


FIGURE 2.3. The structure of AutoRec.

The characteristics of the AutoRec are clearly viewed as compared to the RBM-based model (Salakhutdinov et al., 2007). While RBM-CF proposes a generative, probabilistic model based on RBM, AutoRec is a kind of discriminative model. As for the objective, RBM-CF estimates parameters by maximising log-likelihood, but AutoRec tries to minimise RMSE. Lastly, for efficient model learning, RBM-CF requires a contrastive divergence algorithm, whereas AutoRec focuses on how back-propagation can be much faster. Moreover, compared to MF approaches, which embed both users and items into a shared latent space, AutoRec is required to embed only items or users into a latent space. While MF-based models learn the linear latent representation, AutoRec learns the nonlinear latent representation via the activation function.

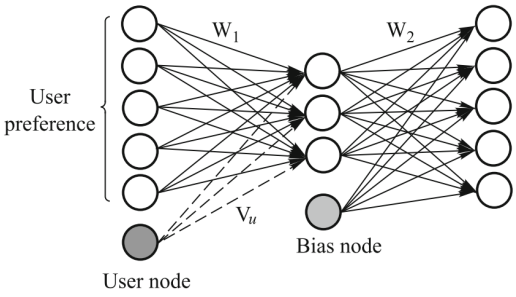


FIGURE 2.4. The structure of CDAE.

CDAE (Wu et al., 2016) presents a CF-based AE framework for top-N recommendation, but it employs the denoising technique (Vincent et al., 2008), which extends the classical AE by training to reconstruct the original input from the intentionally corrupted input. It aims to encourage the hidden layer to discover more robust features and to keep from learning the identity function.

REAP (Zhuang et al., 2017) proposes a collaborative ranking learning framework with user-item pairs, which tries to adopt AE to simultaneously learn user and item latent representations, which are later used to reconstruct the approximated rating matrix. The pairwise ranked loss is considered between the original matrix and the approximated matrix, which imposes information preservation in the approximated matrix. Specifically, it utilises a stacked AE (Hinton et al., 2006; Bengio et al., 2006; Vincent et al., 2010) to initialise the weight matrices of an encoder and a decoder, and focuses on learning better representations only on rating data, while most of the related works require the additional information and do not make full use of the rating data. To get the full benefit from the rating data, it tries to incorporate the pairwise ranked loss defined by user-item pairs into the representation learning framework.

2.1.4 Graph-based collaborative filtering models

The interaction histories of users have an effect on CF for boosting the representation power, and they can be represented as a user-item graph structure. With the success of graph neural network(GNN) (Kipf and Welling, 2016) in modeling the graph-structured data, many works have arisen and proposed how to model the graph structure for neural graph-based representation learning.

SpectralCF (Zheng et al., 2018) firstly introduced CF-based method, which directly learns latent user and item factors from the spectral domains via a spectral convolution operation. It is found that the rich connectivity information in the spectral domain of a user-item bipartite graph is effective in detecting deep user-item connections, and thereby alleviates the cold-start problem for CF-based recommendation. Then, a deep recommender model with multiple layers of the spectral convolution operation is introduced. Inspired by GCN (Kipf and Welling, 2016) and ChebNet (Defferrard et al., 2016) for the node/graph classification problem, it proposes to leverage the broad information in the spectral domain based on a spectral graph theory. Particularly, to overcome the difficulties of directly learning from the spectral domain, it first employs a polynomial approximation, which dynamically adjusts the importance of each frequency domain.

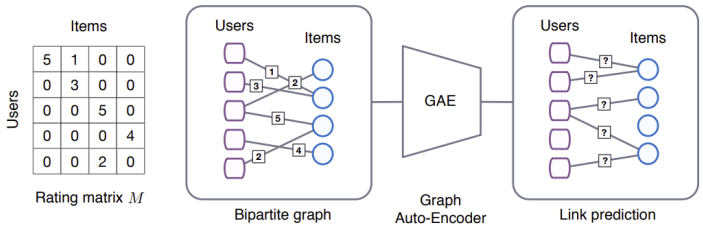


FIGURE 2.5. The structure of GCMC.

GCMC (Berg et al., 2018) considers matrix completion for recommender systems as a link prediction problem on graphs by expressing rating data as a user-item bipartite graph with labeled edges denoted as observed ratings, as shown in Figure 2.5. The benefit of formulating matrix completion as a link prediction task is especially obvious when external information is incorporated into the rating data. Therefore, the cold-start problem can be mitigated. It proposes a graph-based AE framework, which learns latent user and item representations through message passing on the user-item bipartite graph. The trained user and item representations are used to reconstruct rating links in the bipartite graph through a bilinear decoder.

NGCF (Wang et al., 2019a) proposes to integrate the graph structure into the embedding process. It presents a GNN-based recommendation framework, which exploits the user-item bipartite graph via an embedding propagation approach, based on which, the embeddings of interacted user-item pairs are allowed to harvest the collaborative signal. This eventually leads to the expressive modeling in high-order connectivity, injecting the collaborative signal into the embedding process in an explicit way. Moreover, in the proposed framework, there are three components: (1) an embedding layer that initialises the embeddings of users and items; (2) multiple embedding propagation layers that refine the embeddings by injecting high-order connectivity relations; and (3) a prediction layer that aggregates the refined embeddings from each propagation layer.

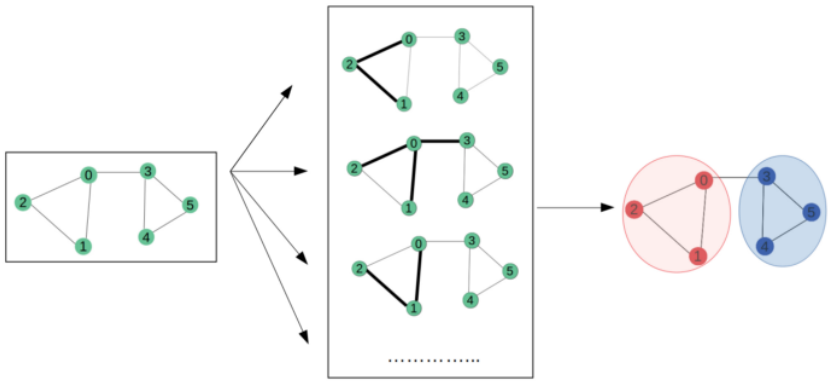


FIGURE 2.6. Illustration of graph convolution for each node with the connected neighbour nodes and neighbourhood aggregation in Graph Convolutional Networks (GCN).

Meanwhile, Graph convolutional networks (GCN)-based recommender models show superior performance compared to the traditional models. GCNs are designed by stacking multiple layers which iteratively perform the following two steps at each layer: neighborhood aggregation with a graph convolution operation (Fig 2.6); a nonlinear transformation. However, many GCN-based recommender models suffer from not only additional complexity from the nonlinear transformation but also the over-smoothing problem in the graph convolution operation as layers are deeper. LR-GCCF (Chen et al., 2020) revisits the current GCN-based models and proposes a simplified graph-based CF model with a linear residual graph convolutional approach. In this proposed model, there are two main characteristics: At each layer of the feature propagation step, it employs a linear embedding propagation, instead of the nonlinear embedding propagation; To alleviate the over-smoothing problem, it adopts the residual preference learning approach at each layer to better preserve the previous layer information.

In addition, LightGCN (He et al., 2020) also aims to simplify the design of GCN, and proposes a graph-based CF model, including only the most essential component of GCN, neighborhood aggregation. The model largely consists of two main components: 1) light graph convolution and 2) layer combination. In light graph convolution, it discards feature transformation and nonlinear activation, which are the standard operations in GCN but increase the difficulty for model training. In layer aggregation, it constructs the final node embedding via the weighted sum of its embeddings from all layers. To sum up, after associating each user/item with an ID embedding, it propagates the embeddings on the user-item bipartite graph, and then combines the trained embeddings from all layers via weighted sum to obtain the final embedding.

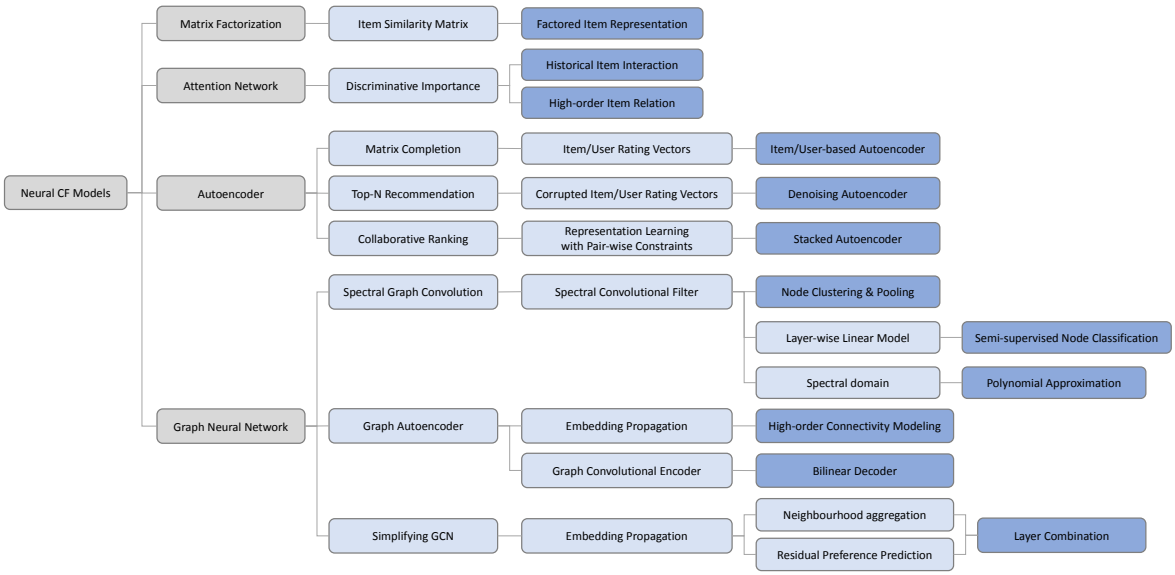


FIGURE 2.7. Summary of the thirteen introduced CF-based neural recommendation models with a distinct characteristic-based breakdown approach, where the neural CF models are largely distinguished into four model architectures/types.

2.2 Feature Extraction in Collaborative Filtering

Collaborative filtering (CF)-based recommender systems focus on making a prediction about the interests of a user by collecting preferences from large number of other users. Matrix completion (Candès and Recht, 2009) is one of the most common formulation, where rows and columns of a matrix represent users and items, respectively. The prediction of users' ratings in items corresponds to the completion of

the missing entries in a high-dimensional user-item rating matrix. In practice, the matrix used for CF is extremely sparse since it has ratings for only a limited number of user-item pairs.

Traditional recommender systems for matrix completion tasks focus on generalising sparsely observed matrix entries to a low dimensional feature space by using an auto-encoder (AE) (Zhang et al., 2020). AEs would help the system better understand users and items by learning the non-linear user-item relationship efficiently, and encoding complex abstractions into data representations.

I-AutoRec (Sedhain et al., 2015) designed an item-based AE, which takes high-dimensional matrix entries, projects them into a low-dimensional latent hidden space, and then reconstructs the entries in the output space to predict missing ratings. Inspired by this, GC-MC (Berg et al., 2018) proposed a graph-based AE framework for matrix completion, which produces the latent features of user and item nodes through a form of message passing on the bipartite interaction graph. These latent user and item representations are used to reconstruct the rating links via a bilinear decoder. Such link prediction with a bipartite graph extends the model with structural and external side information. Recent studies (Rashed et al., 2019; Strahl et al., 2020; Ugla et al., 2020) focused on utilising side information, such as opinions or attributes of users. However, in most real-world settings (e.g. platforms and websites), there is no (or insufficient) side information available about users.

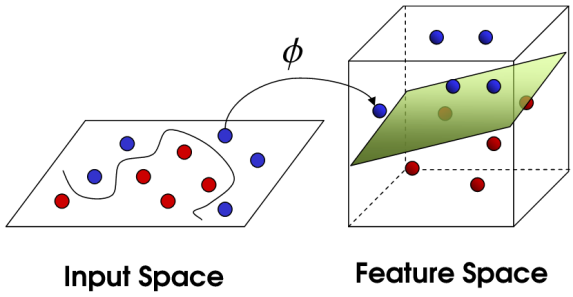


FIGURE 2.8. Illustration of the kernel function in support vector classification.

2.2.1 Kernelised feature extraction techniques

To deal with the sparse/large matrix issue, kernelized models have been recently proposed in the area of recommendation systems. Before we get into the models, matrix factorization (MF) has shown great potential in the CF-based recommendation. Conventional MF models usually assume that the correlated data pairs are distributed on a linear hyperplane (Funk, 2006; Paterek, 2007; Hofmann, 2004). Kernel functions are used widely in support vector machines (SVMs) to classify linearly non-separable

data (Schölkopf et al., 2002), as illustrated in Figure 2.8. The research (Liu et al., 2016) incorporates kernel functions for MF (Lawrence and Urtasun, 2009; Zhou et al., 2012), which embeds the low-rank feature matrices into a higher dimensional space, enabling to learn nonlinear correlations upon the rating data in original space. This model can exemplify the benefits of kernelisation in recommender systems, but it cannot solve the sparse and large matrix issue.

SparseFC (Muller et al., 2018) introduces the concept of kernelised weight matrices, which reduces the number of learnable parameters as weight matrices are sparsified by kernel functions, and also decreases the computational cost in terms of the multiply-accumulate operation. So, this model can solve computationally-expensive and overfitting problem (large matrix issue), but cannot solve the sparse matrix problem. Even though a few models have been proposed, there are still no ideal models which can deal with the sparse/large matrix issue.

To address those problems, we propose to apply multiple types of kernels that have strong ability in extracting the latent features of a high-dimensional user-item rating matrix in a low-dimensional feature space without any additional side information. The first kernel, named *local kernel*, is known to give optimal separating surfaces by its ability to perform the data transformation from high-dimensional space, and is widely used with support vector machines (SVMs). The second kernel, named *global kernel* is from convolutional neural network (CNN) architectures. Based on the rationale that the more kernels with a deeper depth, the higher their feature extraction ability, integrating these two kernels to have best of both worlds successfully extract the low-dimensional feature space. With this in mind, we propose a Global-Local Kernel-based matrix completion framework, called GLocal-K, which includes two stages: 1) pre-training the auto-encoder using a local kernelised weight matrix, and 2) fine-tuning with the global kernel-based rating matrix. The details about GLocal-K will be described in Chapter 3.

2.3 Feature Representation Learning in Collaborative Filtering

User/item-based Collaborative Filtering(CF) has achieved great success in the field of recommendation systems, with the essence of modeling the user-item interaction based on the partially observed interaction matrix. Especially, item-side observation is shown to convey rich information for user modeling and recommendation prediction (Wu et al., 2019). Some existing recommendation approaches demonstrate similar intention of ours in terms of addressing item feature modeling. For instance, early works such as SLIM (Ning and Karypis, 2011) and FISM (Kabbur et al., 2013) learn the latent item factors

via decomposing the interaction matrix with a trainable coefficient matrix and an item-item similarity matrix respectively for top-N recommendation. Besides, there are many auto-encoder (AE)-based recommender systems that learn the item embedding in a low-dimensional hidden space as a latent variable, and reconstruct it in output space to predict the item's ratings from users (Sedhain et al., 2015; Strub et al., 2016; Muller et al., 2018). In order to reinforce the item feature modeling, GLocal-K (Han et al., 2021) further proposes convolutional global kernel to the AE-based matrix completion framework that effectively captures the characteristics of each item.

FIGURE 2.9. Illustration of the item co-occurrence matrix with the original user-item history matrix.

2.3.1 Neighbourhood-based item feature representation approaches

A recent success has been made in exploiting both latent factor and word embedding models for item representation learning in the recommender systems. On the one hand, there are some approaches, such as CoFactor (Liang et al., 2016), RME (Tran et al., 2018), CDIE (Wang et al., 2019b) and CRML (Wu et al., 2020) that implicitly factorize the item-context(neighbourhood) co-occurrence matrix (Fig. 2.9) composed of the Pointwise Mutual Information (PMI) of the respective item and context pairs shifted by a global constant, which is proved to be equivalent to applying the skip-gram model with negative sampling for modeling the latent item embedding (Levy and Goldberg, 2014). However, those item representations are integrated into the matrix completion model and thus is limited to the specific model architecture. On the other hand, several works apply the Word2Vec model for learning item embeddings directly with the semantically ordered item sequence, which can be decoupled from a separate embedding like our SUPER-Rec (Guàrdia-Sebaoun et al., 2015; Wang and Caverlee, 2019; Huang et al., 2021). Nevertheless, same to the other approaches above, the item representation learning is contextualized in an inconsistent manner since the defined item-context varies among users due to their historical interaction with items, e.g. (Liang et al., 2016) and (Tran et al., 2018), or/and the context is sequentialized based on time/session series, e.g. (Wang et al., 2019b) and (Wang and Caverlee, 2019). None of them utilizes

the original fixed position information from the user-item interaction matrix as in our SUPER-Rec for enhancing the item feature representation, which is of great importance since the interaction matrix for completion is composed of rows and columns that naturally entails a fixed context pattern for modeling item-item correlations.

2.3.2 Item interaction-based user feature representation approaches

Especially, many of the aforementioned approaches utilize the item representations to form an effective user representation accordingly, which resolves the limitation of user-specific embeddings. For instance, FISM (Kabbur et al., 2013) combines the rated items from the latent item representation of users to compute user embedding. Those AE-based models such as AutoRec (Sedhain et al., 2015) infer the user rating vectors based on the latent item embeddings modeled in the hidden space. LatentTrajectory (Guàrdia-Sebaoun et al., 2015) uses Word2Vec model to learn the item representation contextualized by sequentially ordered items in the user trajectory. The user representation is then modeled based on the mean of the translations needed to move from one item to the next one in the trajectory. RRLC (Wang and Caverlee, 2019) also applies Word2Vec model and learns the item representation from an timely ordered sequence of interacted items of the user, which is then used to represent the users based on their nearest neighbours measured by the similarity of item representations. Similar to these works, we will utilize our SUPER-Rec item representation to form the user representation and demonstrate its effectiveness together with the SUPER-Rec.

2.4 Summary

In this chapter, we discussed about various approaches used over the past few years in the research field of collaborative filtering-based recommender systems. Initial approaches were more aligned with user/item similarity measure or linear correlation analysis with matrix factorisation. After deep learning has been applied to recommender systems, neural network-based collaborative filtering approaches have arisen with various model architectures based on different feature representation methods. Then, feature extraction approaches in collaborative filtering were discussed. Meanwhile, we focused on the kernelised feature extraction techniques to address the sparse and large user-item matrix issue in collaborative filtering tasks. Lastly, we discussed about representation learning approaches from a separate view for items and users in collaborative filtering.

CHAPTER 3

Global and Local Kernel-based Feature Extraction

Figure 3.1 depicts the architecture of our proposed **GLocal-K** model, which applies two types of kernels in two stages respectively: pre-training (with the local kernelised weight matrix) and fine-tuning (with the global-kernel based matrix)[1]. Note that we pre-train our model to make dense connections denser and sparse connections sparser using a finite support kernel, and fine-tune the pre-trained model, extracting the features of the original rating matrix that is produced from a convolution kernel by reducing the data dimension and producing a less redundant but small number of important feature sets. In this research, we mainly focus on a matrix completion task, which is conducted on a rating matrix $R \in \mathbb{R}^{m \times n}$ with m items and n users. Each item $i \in I = \{1, 2, ..., m\}$ is represented by a vector $r_i = (R_{i1}, R_{i2}, ..., R_{in}) \in \mathbb{R}^n$.

3.1 GLocal-K: Global-Local Kernel-based matrix completion framework

3.1.1 Pre-training with Local Kernel

Auto-Encoder Pre-training

We first deploy and train an item-based autoencoder, inspired by (Sedhain et al., 2015), which takes each item vector r_i as input, and outputs the reconstructed vector r'_i to predict the missing ratings. The model is represented as follows:

$$r'_i = f(W^{(d)} \cdot g(W^{(e)}\, r_i + b) + b'), \tag{3.1}$$

where $W^{(e)} \in \mathbb{R}^{h \times m}$ and $W^{(d)} \in \mathbb{R}^{m \times h}$ are weight matrices, $b \in \mathbb{R}^h$ and $b' \in \mathbb{R}^m$ are bias vectors, and $f(\cdot)$ and $g(\cdot)$ are non-linear activation functions. The autoencoder (AE) deploys an auto-associative

[1]The idea of our pre-training and fine-tuning is different from transfer learning.

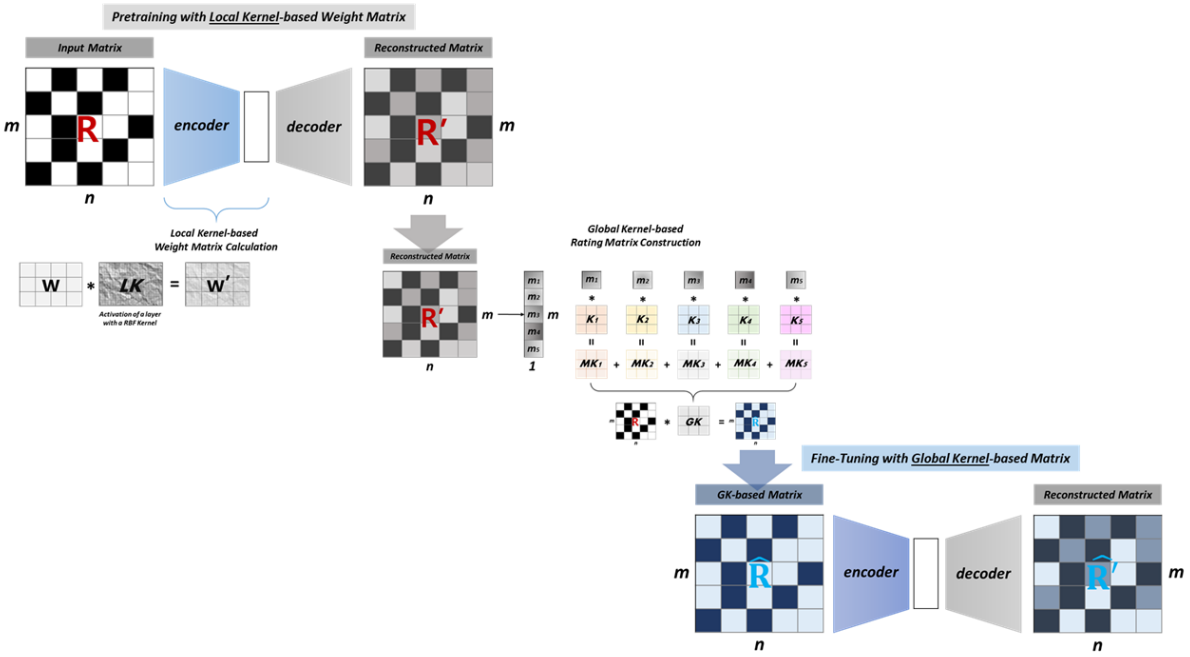


FIGURE 3.1. The GLocal-K architecture for matrix completion. (1) We pre-train the AE with the local kernelised weight matrix. (2) Then, fine-tune the trained AE with the global kernel-based matrix. The fine-tuned AE produces the matrix completion result.

neural network with a single h-dimensional hidden layer. In order to emphasise the dense and sparse connection, we reparameterise weight matrices in the AE through a radial-basis-function (RBF) kernel, which is known as *Kernel Trick* (Giannakopoulos et al., 2008).

Local Kernelised Weight Matrix

In Eq. (3.1), the weight matrices $W^{(e)}$ and $W^{(d)}$ are reparameterised by a 2d-RBF kernel, named *local kernelised weight matrix*. As is illustrated in Figure 3.2, **W'** indicates the local kernelised weight matrix, which is reparameterised by a 2d-RBF kernel, and ***LK*** corresponds to the local RBF kernel.

FIGURE 3.2. Weight reparameterisation via local kernelised weight matrix.

The RBF kernel can be defined as follows:

$$K_{ij}(U, V) = \max(0,\ 1 - ||u_i - v_j||_2^2), \tag{3.2}$$

where $K(\cdot)$ is a RBF kernel function, which computes the similarity between two sets of vectors U, V. Here, $u_i \in U$ and $v_j \in V$. The kernel function can represent the output as a kernel matrix ***LK*** (see Figure 3.2), in which each element maps to 1 for identical vectors and approaches 0 for very distant vectors between u_i and v_j. Then, we compute a local kernelised weight matrix as follows:

$$W'_{ij} = W_{ij} \cdot K_{ij}(U, V), \tag{3.3}$$

where W' is computed by the Hadamard-product of weight and kernel matrices to obtain a sparsified weight matrix. The distance between each vector of U and V determines the connection of neurons in neural networks by making some elements of the weight zeros when two vectors from U and V are far away from each other, and the degree of sparsity is dynamically varied as vectors are being changed at each step of training. As a result, applying the kernel trick to weight matrices enables regularising weight matrices and learning generalisable representations.

3.1.2 Fine-tuning with Global Kernel

Global kernel-based Rating Matrix

We fine-tune the pre-trained autoencoder with the rating matrix, produced by the global convolutional kernel. Prior to fine-tuning, we firstly describe how the global kernel is constructed and applied to build the global kernel-based rating matrix.

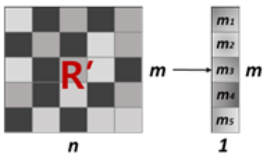


FIGURE 3.3. Item-based average pooling.

As shown in Figure 3.3, the decoder output of the pre-trained model is the matrix that includes initial predicted ratings in the missing entries, and is passed to the item-based average pooling process, by which we summarise each item information in the rating matrix as a single value.

$$\mu_i = \text{avgpool}(r'_i) \tag{3.4}$$

Eq. (3.4) shows that the reconstructed item vector r'_i of the matrix R' from the pre-trained decoder is passed to the average pooling process, and is interpreted as item-based summarisation.

Let $M = \{\mu_1, \mu_2, ..., \mu_m\} \in \mathbb{R}^m$ be the pooling result, which plays a role as the weights of multiple kernels $K = \{k_1, k_2, ..., k_m\} \in \mathbb{R}^{m \times t^2}$. Here, m is the number of items in the matrix and t is defined as the one-side length of a square-shaped global convolution kernel.

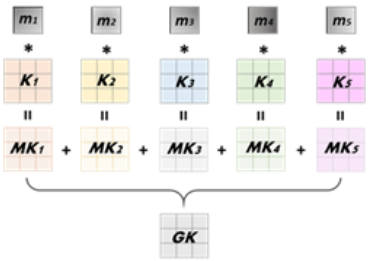


FIGURE 3.4. Global convolution kernel construction.

$$GK = \sum_{i=1}^{m} \mu_i \cdot k_i \tag{3.5}$$

As shown in Figure 3.6, these kernels are aggregated via inner product with the pooling results, and the aggregation result can be dynamically determined by different kernels and different rating matrices so that it can be regarded as the rating-dependent mechanism. Then, the aggregated kernel $GK \in \mathbb{R}^{t \times t}$ is used as a global convolution kernel. Eq. (3.5) demonstrates the procedure for constructing the global convolution kernel GK.

FIGURE 3.5. Convolution operation between the rating matrix and the global kernel.

$$\hat{R} = R \otimes GK \tag{3.6}$$

We apply a global kernel-based convolution operation to the user-item rating matrix to extract the features of the raw rating matrix via the proposed global kernel (see Figure 3.6). In Eq. (3.6), $\hat{R}$ is the global kernel-based rating matrix, which is generated via convolution between the raw rating matrix and the global kernel and is further employed as input for fine-tuning, and $\otimes$ denotes a convolution operation.

Auto-Encoder Fine-tuning

We then explore how the fine-tuning process works.

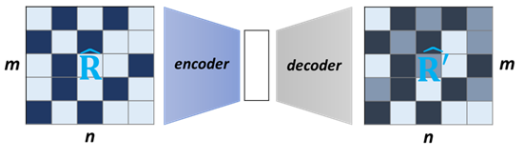


FIGURE 3.6. Fine-tuning process.

The global kernel-based rating matrix $\hat{R}$ is used as input for fine-tuning. It takes all weight matrices of the pre-trained AE model and makes an adjustment of the model based on the global kernel-based rating matrix, as depicted in Figure 3.6. The reconstructed result from the fine-tuned AE corresponds to the final predicted ratings for matrix completion in recommender systems.

3.2 Evaluation setup

3.2.1 Datasets

We conduct experiments on three widely-used matrix completion benchmark datasets: MovieLens-100K (ML-100K), MovieLens-1M (ML-1M) and Douban of density 0.0630, 0.0447 and 0.0152, respectively. These datasets comprise of (100,000 / 1,000,209 / 136,897) ratings of (1,682 / 3,706 / 3,000) movies by (943 / 6,040 / 3,000) users on a scale of $r \in \{1, 2, 3, 4, 5\}$. For ML-100K, we use the canonical u1.base/u1.test data for training and test. For ML-1M, we randomly split into 90:10 train/test sets. For Douban, we use the preprocessed subsets and splits provided by Monti et al. (2017).

TABLE 3.1. Summary statistics of datasets.

Dataset	Users	Items	Ratings	Density
ML-100K	943	1,682	100,000	0.0630
ML-1M	6,040	3,706	1,000,209	0.0447
Douban	3,000	3,000	136,891	0.0152

3.2.2 Baselines

We compare the RMSE with the nine(9) recommendation baselines:

(1) **LLORMA(Lee et al., 2016)** is a matrix factorization model using local low rank sub-matrices factorization. Instead of assuming that a user-item interaction matrix is globally approximated by low-rank matrices, LLORMA assumes that the user-item matrix can serve as a low-rank matrix via certain row-column combinations. Therefore, it proposes to construct a number of low-rank approximations of the matrix, each of which makes accurate approximation in a particular region of the matrix.

(2) **I-AutoRec(Sedhain et al., 2015)** is an autoencoder-based model, which takes high-dimensional matrix entries, projects them into a low-dimensional latent hidden space, and then reconstructs the entries in the output space to predict missing ratings. It considers either user or item embeddings in the encoder, so there are two kinds of autoencoder models: item-based and user-based autoencoder.

(3) **CF-NADE(Zheng et al., 2016)** proposes a neural auto-regressive architecture for collaborative filtering by replacing the role of the restricted Boltzmann machine (RBM) with the neural auto-regressive distribution estimator (NADE) for rating reconstruction, which shares parameters between different ratings of the same item to sufficiently optimise the parameters associated with rare ratings.

(4) **GC-MC(Berg et al., 2018)** is a graph-based autoencoder framework that produces the latent features of users and items via differentiable message passing on the bipartite user-item interaction graph for rating link reconstruction. GC-MC can efficiently combine interaction data with side information in the form of node features as **GC-MC+Extra(Berg et al., 2018)**, which are not fed directly into the graph convolution layer, but are included into the dense hidden layer after graph convolution to alleviate the bottleneck problem of information flow on the graph.

(5) **GraphRec(Rashed et al., 2019)** is a matrix factorization model utilizing graph-based features that are extracted from the Laplacian of the user-item co-occurrence graph to capture user and item profiles. By proposing a simple model that co-embeds users and items into a joint latent space, GraphRec can be applied to a variety of settings and can leverage additional external side information through its numeral vector representation as **GraphRec+Extra(Rashed et al., 2019)**.

(6) **GRAEM(Strahl et al., 2020)** formulates a probabilistic generative model and uses expectation maximization to extend graph-regularised alternating least squares (GRALS) based on the additional side information graph.

(7) **SparseFC(Muller et al., 2018)** proposes an autoencoder-based matrix completion framework whose weight matrices are sparsely reparameterised via interaction with the finite support kernel, which provides the model to regularise weight matrices.

(8) **IGMC(Zhang and Chen, 2020)** is a graph-based matrix completion framework which applies a graph-level GNN to the enclosing 1-hop subgraph to encode its structure, and maps it to the corresponding rating to predict the missing entries of the matrix.

(9) **MG-GAT(Ugla et al., 2020)** uses an attention mechanism to dynamically aggregate neighbor information of each user or item on a neighbour importance graph to learn latent user/item representations.

3.2.3 Implementation Details

We use two 500-dimensional hidden layers for autoencoder and 5-dimensional vectors u_i, v_j for the local RBF kernel. For fine-tuning, we employ a single convolution layer with a 3x3 global convolution kernel. Inspired by (Sedhain et al., 2015), we train our model using the L-BFGS-B optimiser to minimise regularised squared errors, where L_2 regularisation is applied with different regularisation parameters λ_2, λ_s for weight and kernel matrices, respectively. Based on validation results, we choose the following settings for (ML-100K / ML-1M / Douban): for L-BFGS-B optimiser, $maxiter_p$ = (5 / 50 / 5), $maxiter_f$ = (5 / 10 / 5)[2], and for L_2 regularisation, λ_2 = (20 / 70 / 10), λ_s = (0.6e-2 / 1.8e-2 / 2.2e-2). We repeat each experiment five times and report the average RMSE results.

3.3 Result

3.3.1 Overall Performance

We first evaluated our GLocal-K model on ML-100K (u1.base/u1.test split)/-1M datasets as relatively high-density data and compare with the baseline models. The RMSE test results are provided in Table 3.2. It can be easily observed from both GC-MC and GraphRec that incorporate side information improves the recommendation performance, e.g., the error rates of GC-MC+Extra. and GraphRec+Extra. are reduced by 0.005 and 0.007 respectively on ML-100K via side information inclusion. GC-MC transforms a user-item rating matrix into a bipartite user-item interaction graph, and regards a matrix

[2] *maxiter* is maximum number of iterations (p=pre-training, f=fine-tuning).

TABLE 3.2. RMSE test results on three benchmark datasets. The column *Extra.* represents whether the model utilises any side information. All RMSE results are from the respective papers cited in the first column, and the best results are highlighted in bold.

Model	**Extra.**	**ML-100K**	**ML-1M**	**Douban**
LLORMA(Lee et al., 2016)	-	-	0.833	-
I-AutoRec(Sedhain et al., 2015)	-	-	0.831	-
CF-NADE(Zheng et al., 2016)	-	-	0.829	-
GC-MC(Berg et al., 2018)	-	0.910	0.832	-
GC-MC+Extra.(Berg et al., 2018)	O	0.905	-	0.734
GraphRec(Rashed et al., 2019)	-	0.904	0.843	-
GraphRec+Extra.(Rashed et al., 2019)	O	0.897	0.842	-
GRAEM(Strahl et al., 2020)	O	0.917	-	0.732
SparseFC(Muller et al., 2018)	-	0.895	0.824	0.730
IGMC(Zhang and Chen, 2020)	-	0.905	0.857	**0.721**
MG-GAT(Ugla et al., 2020)	O	**0.890**	-	0.727
GLocal-K (ours)	-	**0.890**	**0.822**	**0.721**

completion task as link prediction between user and item nodes. It learns user and item latent representations by passing the message of users (items) to the directly interacted/linked items (users), and then predicts the ratings based on learned user and item representations. Similar to GC-MC, IGMC also learns graph-structural relations on the bipartite user-item interaction graph derived from the rating matrix using GNN, but outperforms GC-MC+Extra. by focusing on 1-hop subgraph structures, which can more precisely understand the relation between a user and an item by representing each node embedding in relation to the subgraph it belongs to. GRAEM focuses on the additional side information graph and MG-GAT uses auxiliary information to represent user-user and item-item graph relations. Different from those models above, the first three models in the table use only the rating matrix structure and achieve better results on ML-1M. Our proposed GLocal-K also draws on the rating matrix structure and uses no extra information, outperforming all the baseline models above on three datasets, including those with additional side information, which illustrates the efficacy of combining the global-local kernels for recommendation tasks. Moreover, SparseFC also achieves higher accuracy than those baseline models on three datasets except for MG-GAT, showing the benefit of proper kernel-approximations into the weight matrix. Our GLocal-K surpasses SparseFC, further illustrating the effectiveness of a global kernel that learns to refine and extract the relevant information from the sparse data matrix.

3.3.2 Cold-start Recommendation

To validate the robustness of our proposed GLocal-K under substantially less user-item interaction information, we varied the training ratio from 1.0 to 0.2 and compared the RMSE test results with SparseFC on ML-100K and Douban in Figure 3.7 and Figure 3.8.

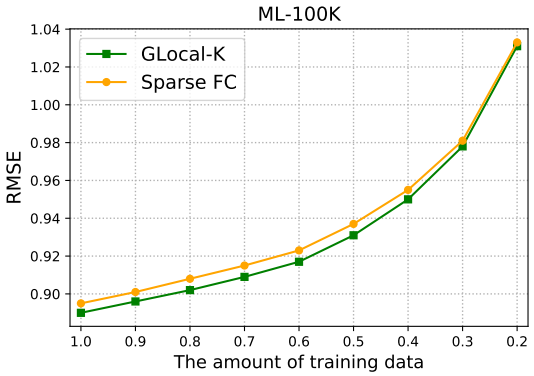


FIGURE 3.7. Performance comparison w.r.t. different sparsity levels on ML-100K.

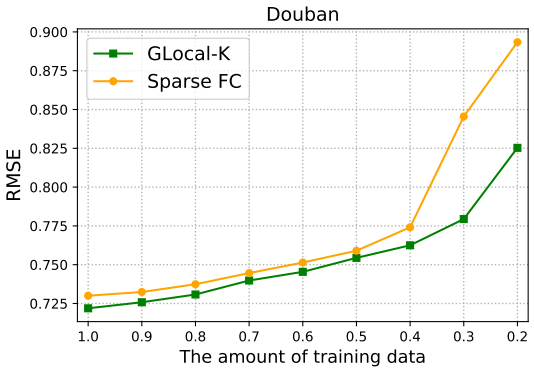


FIGURE 3.8. Performance comparison w.r.t. different sparsity levels on Douban.

It can be seen that both models on the two datasets demonstrate a similar overall trend: the error rate increases as the training size decreases, which complies with the conventional expectation. More specifically, with training ratios of 1.0 to 0.4, both models present the RMSE results within 0.880 to 0.960 on ML-100K and within 0.710 to 0.775 on Douban, and GLocal-K outperforms SparseFC by a merely constant gap on both ML-100K and Douban. This slight gap can be considered insignificant, but new models are continuously being introduced to improve the performance even a little bit, since it determines the ranking of baseline models. So, this merely constant gap illustrates the superior effectiveness of cooperation via global and local kernels of GLocal-K. In addition, when training size reduces from

0.4 to 0.2 on Douban, the error rate of SparseFC deviates from the previous curve and goes up dramatically while GLocal-K still rises at a stable rate as on ML-100K. This implies that the global kernel can deal with scarce data via high-level feature extraction.

3.3.3 Effect of Pre-training

We conducted experiments to find the optimal number of epochs for pre-training on ML-100K, ML-1M and Douban. The RMSE results for the three datasets using pre-training epochs from 0 (i.e., no pre-training) to 60 are provided in Figure 3.9, Figure 3.10 and Figure 3.11.

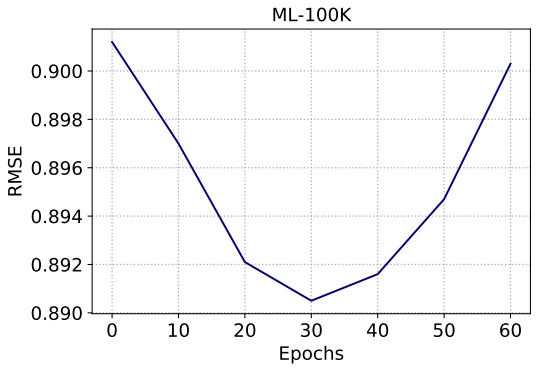


FIGURE 3.9. Performance comparison w.r.t. pre-training epochs on ML-100K.

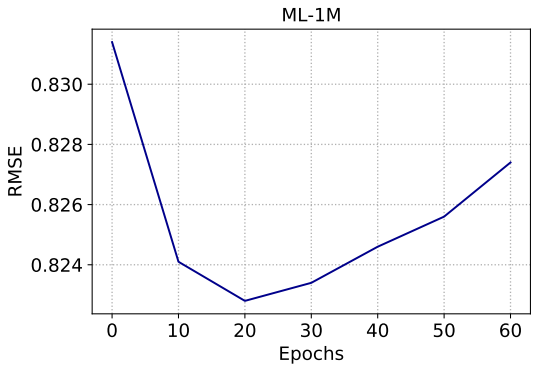


FIGURE 3.10. Performance comparison w.r.t.pre-training epochs on ML-1M.

Similar bowl-shaped curves are presented across all three datasets. The RMSE first keeps decreasing from about 0.901 to 0.891 on ML-100K, from about 0.831 to 0.823 on ML-1M and from about 0.732 to 0.722 on Douban, as the pre-training epoch increases from 0. This indicates that pre-training benefits GLocal-K to achieve better performance on all three datasets. Then, the RMSE starts to go up again (= starts to get worse) after reaching its optimum at 30 epochs for ML-100K and 20 epochs for both

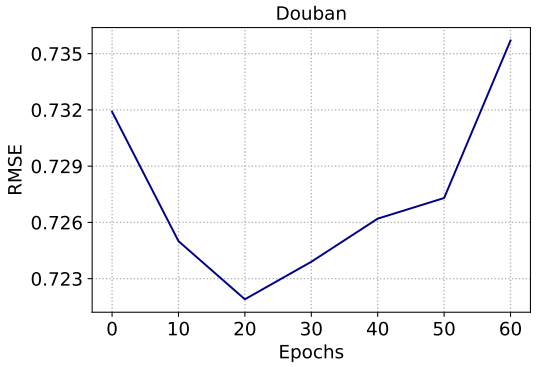


FIGURE 3.11. Performance comparison w.r.t. pre-training epochs on Douban.

ML-1M and Douban. We can also find that the earlier the optimum is reached, the steeper the slope of RMSE is. This implies that there is only difference in optimising speed among the three datasets, but they have the same effect of pre-training on GLocal-K. Referring to the summarised dataset statistics in Table 3.1, we surmise that having more item numbers with lower density may lead to less pre-training for optimal performance.

3.3.4 Effect of Global Convolution Kernel

We conducted several in-depth experiments to explore the effectiveness of convolution using a global kernel. We first tried to vary the kernel sizes and the convolution layers. The RMSE results on the three datasets are presented in Table 3.3 and Table 3.4.

TABLE 3.3. Performance comparison of RMSE test results w.r.t. different convolution kernel sizes. The best results are highlighted in bold.

Kernel size	ML-100K	ML-1M	Douban
3x3	**0.890**	**0.822**	**0.721**
5x5	0.891	0.823	0.723
7x7	0.891	0.823	0.723

It can be seen from Table 3.3 that using 3x3 sized kernel achieves the best performance on all three datasets and the error rate goes up as the size increases to 5x5 or 7x7. It implies that focusing on more local features with smaller kernel size might be more effective for extracting generalizable patterns over the whole data matrix. Moreover, Table 3.4 shows an incremental performance degradation when the convolution layer increases from 1 to 3, indicating a single convolution layer is enough and optimal for feature extraction.

TABLE 3.4. Performance comparison of RMSE test results w.r.t. different numbers of convolution layers. The best results are highlighted in bold.

# Conv. layers	ML-100K	ML-1M	Douban
1	**0.890**	**0.822**	**0.721**
2	0.831	0.827	0.725
3	0.897	0.848	0.732

In addition, we also explored two variants of kernel aggregation mechanisms: (1) integrating the multiple kernels based on the weights that are made by pooling the reconstructed ratings and (2) aggregating the kernels via pure element-wise average.

TABLE 3.5. Performance comparison of RMSE test results w.r.t. different kernel aggregation mechanisms. The best results are highlighted in bold.

Agg. mechanism	ML-100K	ML-1M	Douban
Weight-based	**0.890**	**0.822**	**0.721**
Element-wise	0.894	**0.822**	0.730

As shown in Table 3.5, weight-based kernel aggregation reduces RMSE by 0.004 and 0.009 on ML-100K and Douban while achieving similar performance on ML-1M. Overall, it can be seen that using feature-indicative weights to aggregate the kernels is more effective than purely applying element-wise averages.

TABLE 3.6. Summary statistics of Flixster and YahooMusic.

Dataset	Users	Items	Ratings	Density	Rating type
Flixster	3,000	3,000	26,173	0.0029	0.5, 1.0, ..., 5.0
YahooMusic	3,000	3,000	5,335	0.0006	1, 2, 3, ..., 100

3.3.5 Extremely Sparse Dataset Analysis

We demonstrate that our GLocal-K contributes to improving feature extraction ability for matrix structure under low-resource settings. We further evaluate whether our GLocal-K can reveal its high efficacy under much more extreme settings. We select two widely-used benchmark datasets with much lower density in collaborative filtering-based recommendation tasks: Flixster of density 0.29% and YahooMusic of density 0.06%. Note that the lowest density among the three benchmark datasets is 1.52% from

Douban, which is about five or twenty-five times higher than the other two. As shown in Table 3.6, these datasets consist of (26,173 / 5,335) ratings of (3,000 / 3,000) items by (3,000 / 3,000) users on a scale of $\{0.5, 1.0, 1.5, \ldots, 5.0\}$ and $\{1, 2, 3, \ldots, 100\}$, respectively.

We compared the RMSE test results with the baseline models, which are used in the aforementioned overall performance evaluation (refer to Table 3.2). The RMSE results on Flixtser and YahooMusic are provided in Table 3.7. While our proposed GLocal-K outperforms all the baseline models on ML-100K, ML-1M and Douban, as shown in Table 3.7, GLocal-K highly increases RMSE by 0.108 and 14.9 on Flixster and YahooMusic, compared to the best performance result on each dataset. Similar to GLocal-K, SparseFC is a kernel-embedded autoencoder that aims at efficient feature extraction from rating matrix structure, but it also shows worse RMSE performance under much more extreme settings. The graph-based CF models, including GCMC, IGMC and MG-GAT, outperform the aforementioned two models, which implies that it is more effective to represent relations between users and items via graph neural networks, if their interactions are highly scarce.

TABLE 3.7. RMSE test results on two extremely sparse datasets. All RMSE results are from the respective papers cited in the first column, and the best results are highlighted in bold.

Model	**Flixster**	**YahooMusic**
SparseFC (Muller et al., 2018)	0.981	34.0
GCMC (Berg et al., 2018)	0.917	20.5
IGMC (Zhang and Chen, 2020)	**0.872**	19.1
MG-GAT (Ugla et al., 2020)	0.876	**18.9**
GLocal-K (Han et al., 2021)	0.980	33.8

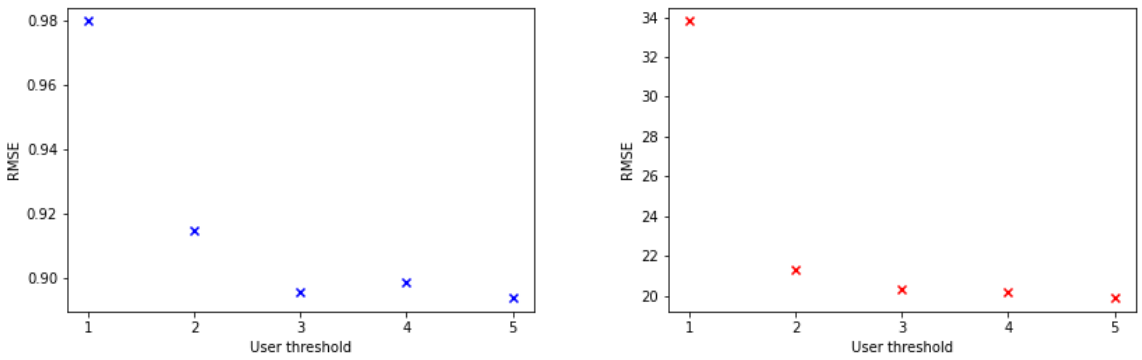
We further analysed the method to alleviate the worse recommendation performance of GLocal-K on highly sparse rating datasets. As an item-based autoencoder model, GLocal-K feeds an item rating vector into the hidden layers of the model and encodes it into a low-dimensional feature vector. We found that the density of each item rating vector from the row of a user-item rating matrix affects the model's feature extraction ability more than the density of the whole matrix.

We suggest applying threshold to users' interactions with items, which removes the users who interacted with items less than the threshold provided. As shown in Table 3.8, we varied the user threshold from 1 (not applicable) to 5, and as the user threshold increases, users are continuously removed but items are rarely removed on both Flixster and YahooMusic datasets. There is a small amount of variation on Flixster across all user thresholds, but RMSE has substantially changed.

TABLE 3.8. Summarisation of matrix statistics w.r.t different user thresholds.

Dataset	Flixster			YahooMusic		
User threshold	**#users**	**#items**	**density**	**#users**	**#items**	**density**
1 (N/A)	2,341	2,956	0.0038	1,357	1,363	0.0029
2	1,979	2,956	0.0044	720	1,335	0.0049
3	1,694	2,955	0.0050	454	1,306	0.0074
4	1,480	2,955	0.0056	318	1,272	0.0093
5	1,327	2,952	0.0061	265	1,250	0.0107

As illustrated in Figure 3.12, compared to the first introduction of the user threshold (= 2), the RMSE result is sharply reduced by about 0.06 and is even similar to the RMSE of GCMC. The trend of gradually reducing RMSE is observed afterwards. Meanwhile, there is a density difference about 3.6 times between the user thresholds of 1 (= not applicable) and 5 on YahooMusic. However, compared to the first introduced user threshold (= 2) with 1, while the rating matrix density is only increases by 0.002, RMSE is dramatically reduced by about 13, which is also at the similar level to the RMSE of GCMC. The RMSE result gradually decreases as the user threshold increases. From the analysis of applying the user threshold to extremely sparse datasets, we surmise that our proposed GLocal-K can achieve similar performance to the state-of-the-art baselines under much lower-resource settings if the users who rarely interacted with items are reduced.



(A) Flixster (B) YahooMusic

FIGURE 3.12. RMSE test result comparison w.r.t. different user thresholds on Flixster and YahooMusic datasets.

3.3.6 Effect of Integrating Global and Local Kernels

We highlighted the efficacy of integrating global and local kernels for efficient feature extraction. To validate it, we evaluate the GLocal-K performance without either of the two kernels on all benchmark datasets, using MAE and NDCG as well as RMSE as evaluation metrics. MAE measures the model's performance by calculating the absolute error between predicted and true rating matrices, and NDCG calculates how more relevant items are recommended earlier to each user.

TABLE 3.9. Performance comparison with the two variants of GLocal-K: (1) without the local kernel and (2) without the global kernel.

Dataset	Model variants	RMSE (↓)	MAE (↓)	NDCG (↑)
ML-100K	**GLocal-K**	**0.890**	**0.695**	**0.906**
	w/o local kernel	0.894	0.700	0.904
	w/o global kernel	0.896	0.701	0.903

Dataset	Model variants	RMSE (↓)	MAE (↓)	NDCG (↑)
ML-1M	**GLocal-K**	**0.822**	**0.641**	**0.929**
	w/o local kernel	0.827	0.644	0.928
	w/o global kernel	0.823	0.643	**0.929**

Dataset	Model variants	RMSE (↓)	MAE (↓)	NDCG (↑)
Douban	**GLocal-K**	**0.721**	**0.562**	**0.943**
	w/o local kernel	0.724	0.563	0.942
	w/o global kernel	0.731	0.571	0.940

As shown in Table 3.9, three model variants demonstrate the same ranking patterns across all evaluation metrics on each dataset. For example, GLocal-K won the first place, followed by GLocal-K without the local kernel and GLocal-K without the global kernel, across all metrics on ML-100K. This implies that the model variants can be clearly distinguished by their performance on each dataset. However, ranking patterns among the model variants are not equivalent across all three datasets. While GLocal-K achieves the best performance on all three datasets, GLocal-K without the local kernel outperforms the other variant on only ML-100K and Douban datasets, indicating that the relative effectiveness between two kernels can be varied depending on which dataset is used. While the NDCG results are very similar among the model variants, there are relatively outstanding differences among them in RMSE. We prove the superior effectiveness of cooperation between global and local kernels by showing that GLocal-K consistently outperforms other model variants in all evaluation metrics on all benchmark datasets.

3.3.7 Matrix Compression Analysis for Global Kernel Construction

Our GLocal-K constructs the global kernel by calculating average pooling for each item rating vector in the reconstructed rating matrix from the pre-trained autoencoder. To validate that the average pooling technique is the optimal way of compressing the prediction information of the pre-trained model, we evaluate the performance of GLocal-K with different pooling techniques on three benchmark datasets, using three evaluation metrics; RMSE, MAE and NDCG.

TABLE 3.10. Performance comparison w.r.t different reconstructed information pooling methods based on three evaluation metrics. Avg.(Average) pooling is the representative method, which is used by our proposed GLocal-K.

Dataset	Method	RMSE (↓)	MAE (↓)	NDCG (↑)
ML-100K	**Avg. pooling**	**0.890**	**0.695**	**0.906**
	Max pooling	0.892	0.698	0.903
	1D convolution	0.899	0.706	0.902
ML-1M	**Avg. pooling**	**0.822**	**0.641**	**0.929**
	Max pooling	0.823	0.642	0.928
	1D convolution	0.829	0.647	0.927
Douban	**Avg. pooling**	**0.721**	**0.562**	**0.943**
	Max pooling	0.725	0.566	0.942
	1D convolution	0.727	0.568	0.941

We introduce max pooling and 1D convolution techniques as other compression methods. Max pooling compresses an item rating vector into a single value by selecting the maximum value in the vector space. 1D convolution extracts a single value from an item rating vector by calculating convolution with a learnable weight vector, which is updated to enable more efficient single value extraction. As shown in Table 3.10, the average pooling technique achieves the best recommendation performance, followed by max pooling and 1D convolution, across all metric measures for each dataset. The NDCG results are very similar with each of the pooling techniques on all datasets. The RMSE and MAE results of max pooling are comparable to those of average pooling on ML-100K and ML-1M datasets but not on Douban. We can find the difference in density between MovieLens datasets (ML-100K/-1M) and Douban, referring to the dataset statistics (Table 3.1), and then we can conjecture that the types of pooling are relatively insignificant if the density of a dataset becomes high.

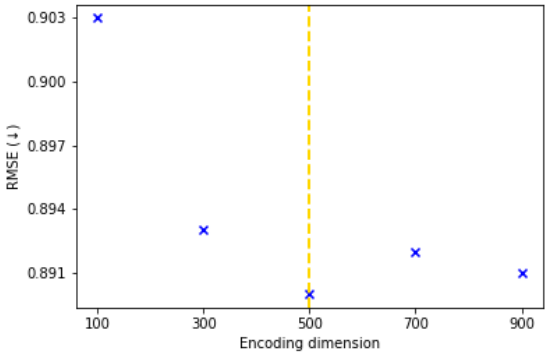
3.3.8 Encoding Dimension Analysis

We explore the optimal encoding dimension in the hidden layers of GLocal-K by comparing the performance results with respect to different encoding dimensions in order to efficiently represent item latent features. We evaluate the GLocal-K performance using the dimension of encoding vectors in every size of 200 starting from 100 to 900. The test results of ML-100K, ML-1M and Douban are provided in Figure 3.13, Figure 3.14 and Figure 3.15, respectively, where the results of RMSE, MAE and NDCG are presented. A yellow dotted line in each subfigure indicates the optimal dimension which produces the best performance among all possible encoding dimensions.

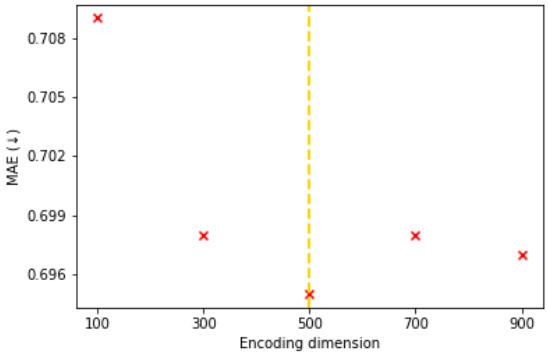
The GLocal-K model was performed best in all metric measures with a dimension of 500 on ML-100K, ML-1M and Douban, showing the similar performance trends with each other based on the variation of encoding dimension. When the equivalent measure results are shown in different dimension sizes such as MAE evaluation in Figure 3.14, MAE and NDCG evaluation in Figure 3.15, we regard the smallest among them as the optimal dimension size due to the advantages in memory space and computation complexity. The model always achieves the best metric measure results with the hidden encoding vector of size 500 across all three datasets. This demonstrates that it can represent the latent features of rating information into a fixed dimension of hidden vector, even though the shape (e.g., # of users, # of items) or the density of a dataset has varied. We can summarise the analysis above as GLocal-K is not required to consider the size of encoding vector dimension in hidden layers over some degree (e.g., 500) to find the optimal one by taking full advantage of both global and local kernels for fine-grained feature extraction.

3.4 Summary

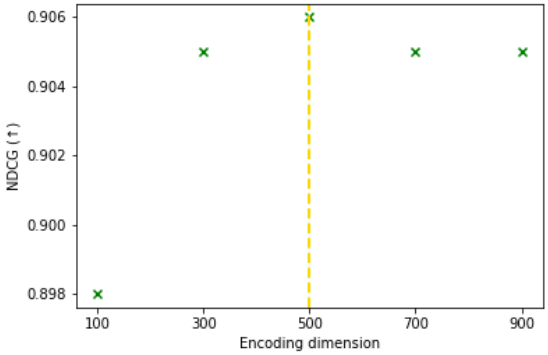
In this chapter, we introduced Global-Local Kernel-based matrix factorisation framework (GLocal-K) for recommender systems, which takes full advantage of both a local kernel at the pre-training stage and a global kernel at the fine-tuning stage for capturing and refining the important characteristic features of the sparse rating matrix under an extremely low resource setting. We demonstrated the RMSE results on three benchmark datasets: MovieLens-100k/-1M and Douban, outperforming the state-of-the-art baseline models. In particular, we highlighted the effectiveness of our global kernel for exerting scarce data by evaluating the cold-start recommendation. From several further experiments, we demonstrated the importance of each component within GLocal-K and explored the optimal settings of components to reveal the highest efficacy of integrating global and local kernels.



(A) RMSE ↓

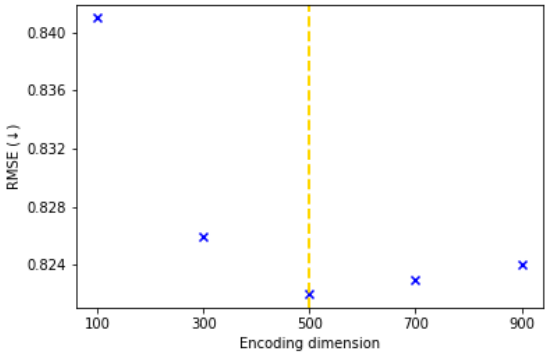


(B) MAE ↓

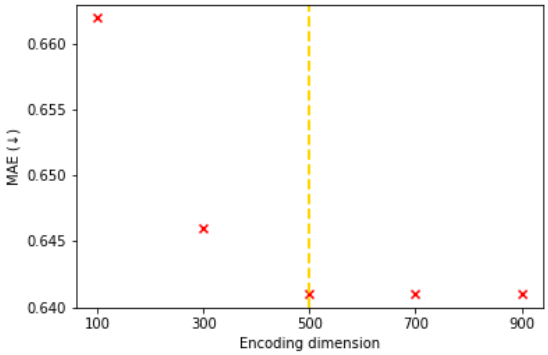


(C) NDCG ↑

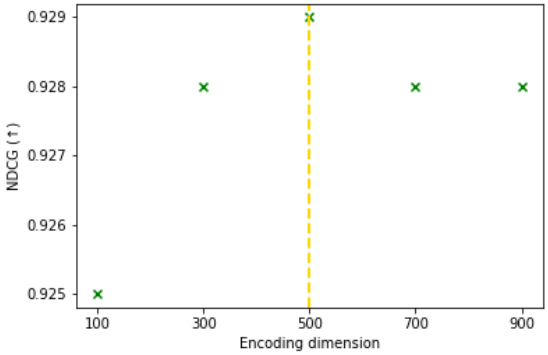
FIGURE 3.13. Performance comparison w.r.t. different dimensions of the hidden encoding vector in autoencoder starting from 100 to 900 in every size of 200 based on three evaluation metrics (RMSE / MAE / NDCG) on ML-100K.



(A) RMSE ↓

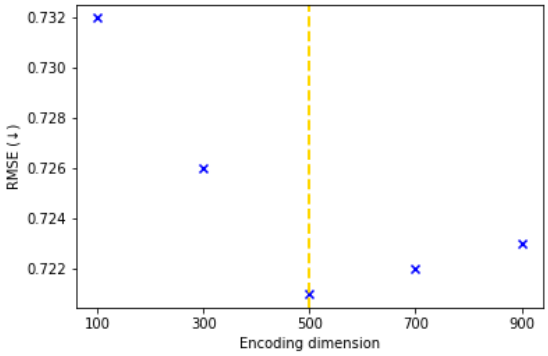


(B) MAE ↓

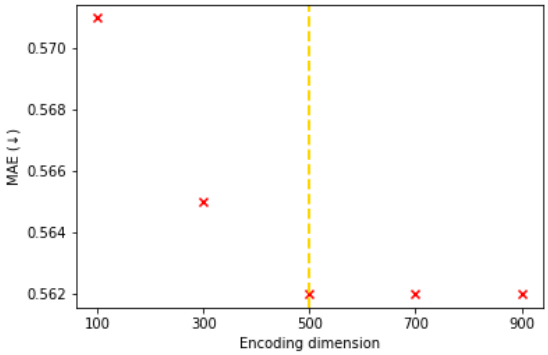


(C) NDCG ↑

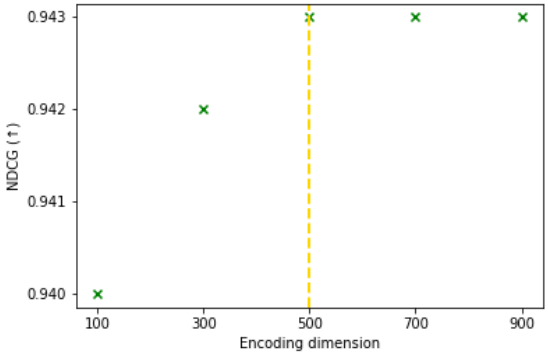
FIGURE 3.14. Performance comparison w.r.t. different dimensions of the hidden encoding vector in autoencoder starting from 100 to 900 in every size of 200 based on three evaluation metrics (RMSE / MAE / NDCG) on ML-1M.



(A) RMSE ↓



(B) MAE ↓



(C) NDCG ↑

FIGURE 3.15. Performance comparison w.r.t. different dimensions of the hidden encoding vector in autoencoder starting from 100 to 900 in every size of 200 based on three evaluation metrics (RMSE / MAE / NDCG) on Douban.

CHAPTER 4

Position-Enhanced Feature Representation based on Surrounding Neighbor Information

---

Data for recommendation tasks are generally composed of a set of users and items, and the feedback that users give to items. It is normally stored as a user-item rating matrix $D \in \mathbb{R}^{M \times N}$ where each row $i \in M$ and each column $j \in N$ correspond to the i-th user and the j-th item while their indexed value represents the corresponding feedback $D_{i,j}$ that the i-th user gives to the j-th item, which can be either explicit feedback such as movie rating ranging from 1-5, or implicit feedback such as the user click behavior denoted as 0 or 1. For each user at the row of the user-item matrix D, each item can be uniquely identified by its fixed position at the column shared among all users, and user's taste is indicated by the given feedback for the item. We propose a surrounding position-enhanced feature representation for recommendation systems, SUPER-Rec, which utilizes only the user-item rating matrix D for capturing the latent item features by learning to model user's taste over an item based on how its surrounding positioned items are liked by the user. Figure 4.1 articulates the three main stages for SUPER-Rec recommendation: 1) User-item Positioning, 2) SUPER-Rec Training, and 3) Matrix Completion with SUPER-Rec. We describe each stage in details as follows.

4.1 Surrounding Position-Enhanced Representation for Recommendation

4.1.1 User-item Matrix Positioning

To begin with, we first define the target items with their surrounding neighbours so as to prepare the training corpus for SUPER-Rec. As is shown in the first stage in Figure 4.1, we reformulate the aforementioned user-item rating matrix $D \in \mathbb{R}^{M \times N}$ as $D \in \mathbb{R}^{N \times M}$ for better illustration, where each row $j \in N$ represents the item and each column $i \in M$ indicates the user. We regard each item with feedback available in D as one of the target items. Assume that a target item is positioned at the row j in D

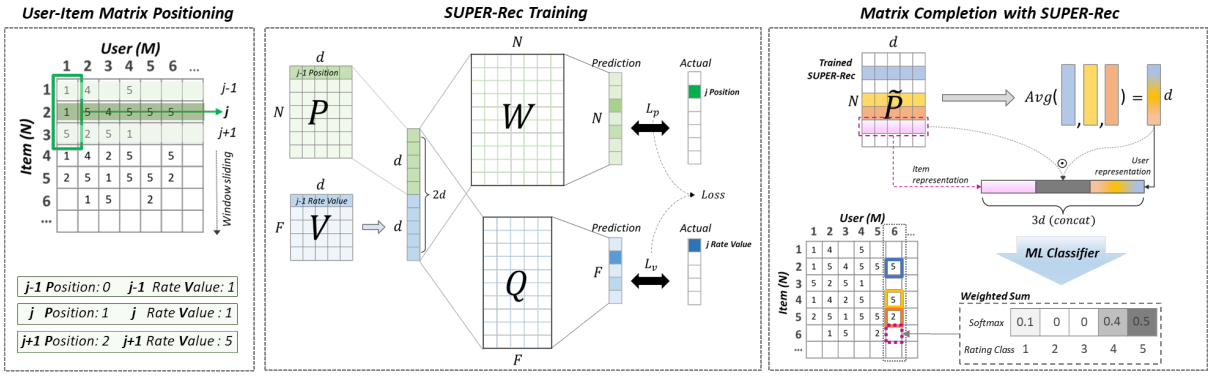


FIGURE 4.1. The SUrrounding Position-Enhanced Representation for Recommendation (SUPER-Rec) architecture

and is given a feedback $D_{j,i}$ from user i, being either explicit or implicit. Then, we define the scope of surrounding neighbour items based on the bilateral context of the target item as $\{j+k \mid k \in \{\mathcal{K} \cup -\mathcal{K}\}\}$, where $\mathcal{K} \in \{1, 2, ..., K\}$ and K is the window size that indicates the single-side length of the context. Likewise, the corresponding item feedback from user i in this surrounding context can be represented as $\{D_{j+k,i} \mid k \in \{\mathcal{K} \cup -\mathcal{K}\}\}$. We set k to $-j \leq k < N - j$ to constrain the surroundings in the scope of D. Moreover, all users and items are assigned to the row and column of the matrix D, and neighbours with no feedback can be also represented with a zero value for feedback and their position. So, it has not been considered whether surrounding neighbours are insufficient.

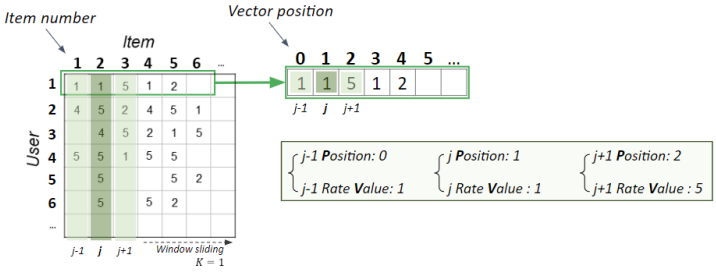


FIGURE 4.2. The training corpus C preparation by extracting the position and rate value of surrounding neighbour items $(j-1, j+1)$ within window size 1. Each item index of a matrix corresponds to an item number, but each index of a vector is assigned by a vector position starting at 0.

The example demonstrated in Figure 4.2 takes the target item at row $j = 2$ for user $i = 1$ as with position index of 1 and feedback of 1, indicating the explicit feedback ranging from level 1-5. With window size $K = 1$, the bilateral surrounding context can be defined as item neighbours with position index of $j - 1 = 0$ and $j + 1 = 2$ while their corresponding item feedback would be 1 and 5. By this way, we can form the training corpus C for SUPER-Rec, consisting of each target item with its

surrounding neighbour items. During the training, we will utilize these position-aware neighbour items and their feedback of user taste as the condition for inferring the identity(position) and feedback of the target item.

4.1.2 SUPER-Rec Training

In order to learn the SUPER-Rec representation for items using the prepared corpus C, we first define an item position embedding matrix P and a feedback embedding matrix V, where $P = (p_1, ..., p_N) \in \mathbb{R}^{N \times d}$ and $V = (v_1, ..., v_F) \in \mathbb{R}^{F \times d}$. Each embedding vector p or v uniquely corresponds to a specific item or feedback type. N and F refer to the total number of items and feedback types, e.g. F = 5 for 1-5 ranged user rating and F = 2 for the binary user interaction data such as user clicks, whereas d denotes the embedding dimension. We then define two transformation operators $f(\cdot)$ and $g(\cdot)$ that learn to transform the combined position and feedback embedding of a surrounding neighbour item into both item space $E \in \mathbb{R}^N$ and feedback space $T \in \mathbb{R}^F$ so as to predict the position and feedback of the target item accordingly.

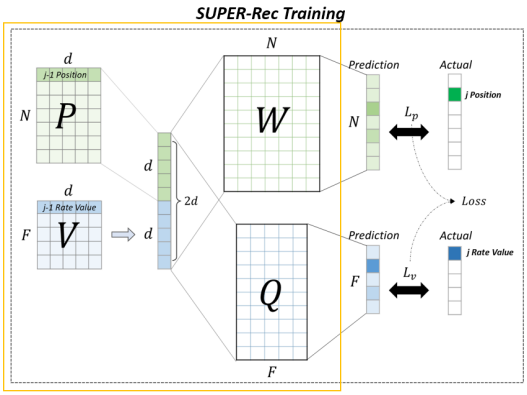


FIGURE 4.3. First stage of SUPER-Rec training process: 1) concatenate the position and rating embedding vectors 2) project the concatenated vector into two transformation matrices (W, Q).

Concretely, the transformation process is shown in Eq.(4.1) and (4.2). Given an target item j with a surrounding neighbour item $j + k$ for user i, we first concatenate ($\oplus$) the position embedding vector p_{j+k} and the feedback embedding vector $v_{D_{j+k,i}}$ of the surrounding neighbour item. Then, we project the concatenated embedding ($\in \mathbb{R}^{2d}$) into the item space and feedback space via multiplying with the learnable weight matrices $W \in \mathbb{R}^{2d \times N}$ and $Q \in \mathbb{R}^{2d \times F}$ respectively, resulting in the two projected vectors for target item position and feedback prediction. An illustrative example of predicting for the

target item j utilizing one of its surrounding neighbour item $j-1$ is demonstrated in the second stage in Figure 4.1.

$$f(j+k,i) = W^{\top}[p_{j+k} \oplus v_{D_{j+k,i}}] \tag{4.1}$$

$$g(j+k,i) = Q^{\top}[p_{j+k} \oplus v_{D_{j+k,i}}] \tag{4.2}$$

$$i \in M, j \in N, k \in \{\mathcal{K} \cup -\mathcal{K}\}$$

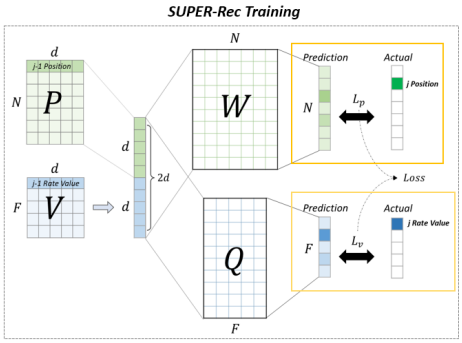


FIGURE 4.4. Second stage of SUPER-Rec training process: After the input vector is projected into item space and rating space, calculate 1) the loss between the predicted target item and the actual target item and 2) the loss between the predicted target rating and the actual target rating.

During the training, we jointly optimise based on two objectives. The first objective is to minimize the item position prediction loss L_p for predicting each target item position as j' against the ground truth position j, given the user i. As is indicated in Eq.(4.3), it calculates the summed cross-entropy loss over predictions of all the surrounding neighbour items $j+k$. Here $\sigma_j(\cdot)$ is the negative-log softmax probability of the predicted target item position being the ground truth j based on the information of surrounding neighbour item $j+k$, as is derived in Eq.(4.4). ϵ is used to avoid the Log-0 problem.

$$L_p(j' = j|i) = \sum_{k \in \{\mathcal{K} \cup -\mathcal{K}\}} \sigma_j\big(f(j+k,i)\big) \tag{4.3}$$

$$\sigma_j\big(f(j+k,\,i)\big) = -\log\left(\frac{\exp\big(f_j(j+k,\,i)\big)}{\sum_{a \in N}\exp\big(f_a(j+k,\,i)\big)} + \epsilon\right) \tag{4.4}$$

The second objective is to minimize the feedback prediction loss L_v, which is calculated in a similar way but is instead based on the feedback predictions of the target item, as is shown in Eq.(4.5) for loss calculation and Eq.(4.6) for deriving the negative-log softmax probability of the predicted feedback being the ground truth target item feedback r.

$$L_v(D'_{j,i} = r) = \sum_{k \in \{\mathcal{K} \cup -\mathcal{K}\}} \sigma_r\big(g(j+k,\, i)\big) \tag{4.5}$$

$$\sigma_r\big(g(j+k,\, i)\big) = -\log\left(\frac{\exp\big(g_r(j+k,\, i)\big)}{\sum_{b \in F} \exp\big(g_b(j+k,\, i)\big)} + \epsilon\right) \tag{4.6}$$

Algorithm 1 SUPER-Rec training algorithm

Require: $\{j,\, i\} \leftarrow$ {target item, given user}
Ensure: $\mathcal{L} \leftarrow$ Joint cross-entropy loss for the j-th target item
Define $P \in \mathbb{R}^{N \times d} \leftarrow$ Item position embedding matrix
$V \in \mathbb{R}^{F \times d} \leftarrow$ Feedback embedding matrix
$W \in \mathbb{R}^{2d \times N}, Q \in \mathbb{R}^{2d \times F} \leftarrow$ Two transformation weight matrices
Initialize $L_p = 0$, $L_v = 0$, $\alpha \in (0,\, 1)$
$r \leftarrow D_{j,i}$
for $k \in \{-K, -K+1, ..., -1, 1, 2, ..., K\}$ **do**
  if $k < -j$ or $k \geq N - j$ **then**
    A surrounding neighbour item is out of range of item position, so skip the below statements and continue for-loop.
  end if
  Calculate $f(j+k,\, i)$ for the projected vector in item space using Eq.(4.1).
  Calculate $\sigma_j\big(f(j+k,\, i)\big)$ for the log softmax probability of the predicted target item position using Eq.(4.4).
  $L_p \leftarrow L_p + \sigma_j\big(f(j+k,\, i)\big)$ *(refer to Eq.(4.3))*
  Calculate $g(j+k,\, i)$ for the projected vector in feedback space using Eq.(4.2).
  Calculate $\sigma_r\big(g(j+k,\, i)\big)$ for the log softmax probability of the predicted target item feedback using Eq.(4.6).
  $L_v \leftarrow L_v + \sigma_r\big(g(j+k,\, i)\big)$ *(refer to Eq.(4.5))*
end for
$\mathcal{L} \leftarrow (1-\alpha) \cdot L_p + \alpha \cdot L_v$
return $\mathcal{L}$

We jointly optimise these two objectives based on their weighted sum. As is denoted in Eq.(4.7) for our joint loss $\mathcal{L}$, α is a hyperparameter that balances between penalties of the item position prediction and the feedback prediction. As this prediction proceeds, the item position embedding matrix P, the transformation weight matrices W and Q are updated accordingly whereas the feedback embedding matrix V is fixed throughout the training process. After the training is finished, we extract the trained position embedding matrix $\tilde{P} \in \mathbb{R}^{N \times d}$ as our SUPER-Rec for item feature representation, from which

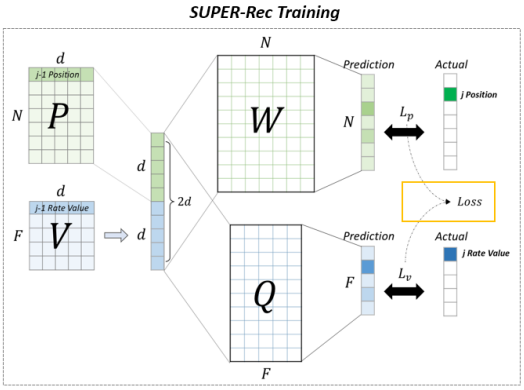


FIGURE 4.5. Third stage of SUPER-Rec training process: calculate the final joint loss via the weighted sum of item prediction loss and rating prediction loss.

each row vector $\tilde{p_j}$ can uniquely represent each item $j \in N$. Note that there are many choices to calculate the loss for multiple objectives such as joint learning, alternate learning and multi-task learning, but we empirically find out there is no big difference to our SUPER-Rec optimisation. In addition, we describe the details for the overall training procedure of our proposed SUPER-Rec representation model in Algorithm 1.

$$\mathcal{L} = (1 - \alpha) \cdot L_p(j' = j|i) + \alpha \cdot L_v(D'_{j,i} = r) \tag{4.7}$$

4.1.3 Matrix Completion with SUPER-Rec

The trained SUPER-Rec representation $\tilde{p_j}, j \in N$ for items captures the position-enhanced latent item features correlated to users' tastes. Based on it, we can further construct user embedding vectors $z_i, i \in M$. The intuition here is that the unique preference of a user for providing the feedback to items can be inferred via their historical feedback-giving patterns over all their interacted items represented by SUPER-Rec. Concretely, as is demonstrated in the third stage in Figure 4.1, for a user i, we first weigh each item j based on the feedback $D_{j,i}$ given by this user and calculate the weighted average of all their interacted items represented by SUPER-Rec accordingly. In case of the aforementioned implicit feedback, we directly calculate the average of SUPER-Rec of all the interacted items, i.e. $D_{j,i}$ = 1. It is worth to note that here we do not use original feedback types, e.g. 1-5, but instead their power-raised values as the weights for the summation so that more preferred items would result in much higher weight and thus the user's preference can be better emphasized. In addition, since we are not to recommend

only the most favourable items to users but instead to model how users react to all items that they did not give feedback yet, we adopt users' historical feedback from all items no matter the feedback is positive or not.

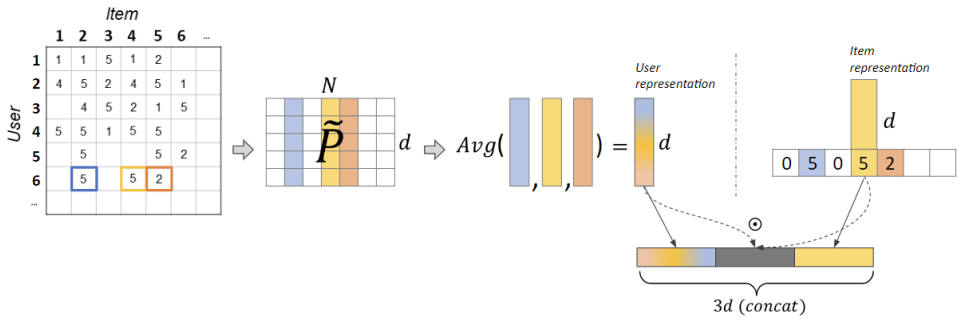


FIGURE 4.6. Input representation with three different representations for matrix completion: 1) user representation via weighted sum between user's historical item representations and their ratings, 2) item representation and 3) user-item relation representation via element-wise product between the item and user representations.

With the SUPER-Rec based item and user representations, our end goal of recommendation is to predict all the unobserved item ratings in the user-item rating matrix via matrix factorisation (MF). In order to demonstrate that our SUPER-Rec item representation and its derived user representation alone can exert superior feature representation power that leads to excellent recommendation performance in any neural network model, we apply it to a matrix factorisation model with only simple neural network structure: Feed-Forward Neural Networks-based MF (NNMF), which adopts a multi-layer perceptron. As is denoted in Eq.(4.8) for the NNMF model $h(\cdot)$, given user-item pair (i, j), it first conducts an element-wise product ($\odot$) between the item representation $\tilde{p}_j$ and the user representation z_i to derive the user-item relation representation, which is then concatenated ($||$) with $\tilde{p}_j$ and z_i. Followed by (Dziugaite and Roy, 2015), we then employ a three-layer neural network with *relu* activation denoted as nn. We conduct a log softmax layer σ over the output of $h(\cdot)$ as in Eq.(4.9), where σ_r represents the negative-log probability of the predicted feedback being $r \in F$.

$$h(j, i; \theta) = nn[\tilde{p}_j || \tilde{p}_j \odot z_i || z_i] \tag{4.8}$$

$$\sigma_r\big(h(j, i; \theta)\big) = -\log\left(\frac{\exp\big(h_r(j, i; \theta)\big)}{\sum_{b \in F} \exp\big(h_b(j, i; \theta)\big)} + \epsilon\right) \tag{4.9}$$

Our final loss function $\mathcal{L}_{\text{MF}}$ for training is formulated in Eq.(4.10), in which we adopt $I[\cdot]$ to calculate the loss from feedback prediction being the ground truth $D_{j,i}$, the value of which equals to 1 only when r equals to the ground truth feedback $D_{j,i}$ and 0 otherwise. The matrix $\Omega \in \{0,1\}^{N\times M}$ serves as a mask for unobserved item feedback so that ones occur when only for items with available feedback in the original rating matrix while zeros refer to unobserved item feedback. Hence, the training optimisation will only be conducted over observed feedback in the rating matrix, updating all the trainable parameters θ, including $\bigcup_{l=1,2,3}\{\omega_l, \beta_l\}$ of the neural network nn and the trained item position embedding matrix $\tilde{P}$, while considering L2 regularisation for $\{\omega_l | l = 1,2,3\} \cup \{\tilde{p}_j | j \in N\}$. λ is a scaling factor for the two regularization terms. We train the model based on the loss L_{MF} with the training set while selecting the best model based on the validation set.

$$\mathcal{L}_{\text{MF}} = \sum_{j,i;\Omega_{j,i}=1} \sum_{r\in F} I[D_{j,i} = r] \cdot \sigma_r\big(h(j,i;\theta)\big) + \frac{1}{2}\lambda \cdot \Big(\sum_{l=1,2,3} ||\omega_l||^2 + \sum_{j\in N} ||\tilde{p}_j||^2 \Big) \tag{4.10}$$

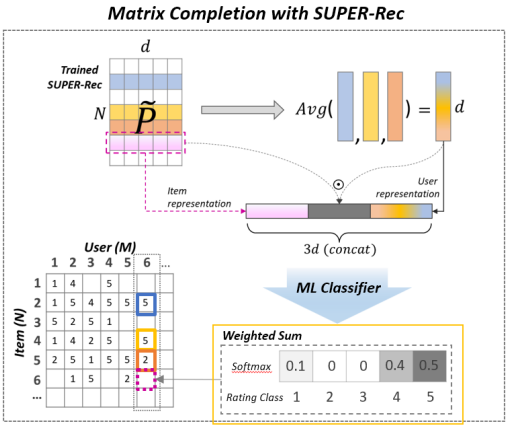


FIGURE 4.7. Matrix completion with SUPER-Rec by using the weighted sum of probability distribution from a ML classifier and the rating type (i.e. {1,2,3,4,5}).

The trained SUPER-Rec with NNMF empowers the inductive inference for any user i, either being an existing one or a new one, provided with all the historical rated feedback over the N items. As is shown in Eq.(4.11), to infer an unobserved feedback over an item $t \in N$ for a user i, we apply the softmax over the output of the trained SUPER-Rec with NNMF (parameterized with $\tilde{\theta}$, denoting the trained parameters in θ), which derives the probability distribution over each feedback candidate $r \in F$, being either implicit or explicit. We then use the weighted sum (expected value) $D'_{t,i}$ as our final predicted feedback for recommendation.

$$D'_{t,i} = \sum_{r \in F} r * \frac{\exp\big(h_r(t, i; \tilde{\theta})\big)}{\sum_{b \in F} \exp\big(h_b(t, i; \tilde{\theta})\big)} \tag{4.11}$$

TABLE 4.1. Statistics of six explicit datasets and two implicit datasets used in the experiments. Note that explicit feedbacks are represented with the specific ratings in different ranges (e.g. 1-5, 1-100) and implicit feedbacks are based on the user's action (Clicked or Not-Clicked).

Feedback	Dataset	#Users	#Items	#Ratings	Density	Rating types
Explicit	ML-100K	943	1,682	100,000	6.30%	1, 2, 3, 4, 5
	ML-1M	6,040	3,706	1,000,209	4.47%	1, 2, 3, 4, 5
	ML-10M	71,567	10,681	10,000,054	1.30%	0.5, 1, 1.5, ..., 5
	Douban	3,000	3,000	136,891	1.52%	1, 2, 3, 4, 5
	Flixster	3,000	3,000	26,173	0.29%	0.5, 1, 1.5, ..., 5
	YahooMusic	3,000	3,000	5,335	0.06%	1, 2, 3, ..., 100
Implicit	Amazon-Books	52,643	91,599	2,931,466	0.06%	Clicked/Not-Clicked
	Amazon-Beauty	2,944	57,289	82,904	0.04%	Clicked/Not-Clicked

4.2 Evaluation Setup

4.2.1 Datasets

We conducted experiments on eight(8) widely-used recommendation benchmark datasets with both explicit and implicit feedback, MovieLens-100K (ML-100K) (Miller et al., 2003), MovieLens-1M (ML-1M) (Miller et al., 2003), MovieLens-10M (ML-10M) (Harper and Konstan, 2015), Douban (Ma et al., 2011), Flixster (Jamali and Ester, 2010), YahooMusic (Dror et al., 2012), and two(2) implicit feedback, Amazon-Books and Amazon-Beauty (He and McAuley, 2016; McAuley et al., 2015). Note that the first six datasets contain users' explicit ratings on items, and the latter two datasets provide users' implicit feedback on items, indicating users' actions (clicked or not-clicked).

We directly adopt the canonical u1 train/test splits as in (Monti et al., 2017; Rao et al., 2015) for ML-100K and randomly split the dataset into 0.9/0.1 train/test set for ML-1M and ML-10M, followed by (Muller et al., 2018). For Douban, Flixster and YahooMusic, we use the preprocessed train/test splits provided by (Monti et al., 2017). For the two Amazon datasets, we use the last ten interactions of each user as the test set and keep the rest as the train set, followed by the IDCF (Wu et al., 2021). The raw Amazon datasets are very large and sparse, so IDCF (Wu et al., 2021) filtered out infrequent items and

users with less than five interactions. We leave out 10% of the training data as the validation set for early stopping in training[1]. The statistics of datasets are summarised in Table 4.1.

4.2.2 Baselines

We compare our novel position-enhanced embedding learning model, SUPER-Rec, with the following six(6) baseline models for explicit datasets. We first include models with graph-based user/item embeddings, GC-MC and IGMC.

- **GC-MC (Berg et al., 2018)** is a graph-based matrix completion framework that extracts user, item embeddings using a GNN and reconstructs the rating links via a bilinear decoder.
- **IGMC (Zhang and Chen, 2020)** is also a graph-based model but extracts 1-hop enclosing subgraphs of user-item pairs and uses a GNN to encode the subgraph-based item embeddings.

In addition, feature-based user/item embedding models, SparseFC and GLocal-K, are selected to compare.

- **SparseFC (Muller et al., 2018)** is a user/item-based autoencoder (AE) model, which feeds a high-dimensional user-item matrix, projects it into a low-dimensional latent feature space, and then reconstructs its entries to predict unknown ratings. This model regularises the weight matrices of hidden layers by projecting them into a low-dimensional space using support kernel matrices.
- **GLocal-K (Han et al., 2021)** is an item-based AE architecture model, which takes a user-item rating matrix as input and extracts item embeddings from each row vector of item ratings in the user-item matrix. Moreover, it proposes to apply two types of kernels in two training stages to improve the feature extraction performance: 1) pre-training the AE model using local kernelised weight matrices, and 2) fine-tuning the pre-trained model with the rating matrix, produced via convolution with global kernels.

Finally, the inductive user/item embedding learning model, IDCF, is also considered as baseline. Note that IDCF has two variants; one (IDCF-NN) adopts multi-layer perceptron as a prediction model, and another (IDCF-GC) applies graph convolution networks.

[1]We tested 5, 10, 20% of the training data as the validation set, but there is no significant difference.

- **IDCF (Wu et al., 2021)** proposes an inductive collaborative filtering framework, which is comprised of the two-stage training process: 1) pre-training a matrix factorisation model to obtain pre-trained user embeddings, which are further leveraged as metadata to inductively compute new users' embeddings, and 2) learning the user graph with message-passing layers based on the pre-trained meta embeddings to generalise to compute inductive representations for new users.

For the implicit feedback dataset, we applied four(4) baseline models, two variants of IDCF as above, and two item-based CF models with item embeddings.

- **FISM (Kabbur et al., 2013)** proposes an item-item similarity matrix-based top-N recommendation framework, which learns the matrix as a product of low-rank latent factor matrices and estimates the ratings based on the dot product between the aggregated latent vector of the items that have been rated by a user (user embedding) and unrated item latent vector from different factor matrices.
- **MultVAE (Liang et al., 2018)** proposes a variational autoencoder-based collaborative filtering framework, which adopts multinomial conditional likelihood on the generative model (decoder) for modelling user-item implicit feedback data, and makes predictions by sorting all the items based on the predicted multinomial probability.

4.2.3 Implementation Details

For all benchmark datasets, we used the window size of 1, which included the very next item on the left and right sides as neighbours in the context. Adam optimiser and early stopping technique with a patience parameter of 5 were used for training optimisation. We empirically applied the learning rate of $1e$-4 for ML-1M, $1e$-2 for Amazon-Books and $1e$-3 for all the other datasets. The batch size was set in proportion to the data size: 500 for YahooMusic, 5,000 for Amazon-Books, 10,000 for ML-10M and 1,000 for all the other datasets. We searched for the optimal embedding dimension in every size of 100, starting from 100 for the explicit rating-based datasets. For Amazon-Books and Amazon-Beauty, we tried dimensions within [200, 500, 1000] and [200, 500, 1000, 2000, 4000]. We found that the data with a higher density (or lower sparsity) requires a deeper search to find the optimal embedding dimension. We restricted the number of interactions of users to 100 for ML-10M and Amazon-Books due to the limitation of computational and memory resources. We trained the MF model using the Adam optimiser

and the L2 regularisation technique. We searched for the optimal regularisation parameter within [0.1, 0.3, 0.5, 1.0, 3.0, 5.0] for all datasets. The learning rate was empirically selected as $1e$-5 for ML-100K, ML-1M and Amazon-Beauty, and $1e$-4 for all the other datasets. The batch size was set to 500 for YahooMusic, 5,000 for Amazon-Books and 1,000 for all the other datasets.

4.2.4 Evaluation Metrics

For datasets with explicit feedback, the goal is to predict user's ratings on items, i.e. estimate the missing values in a user-item rating matrix. We use the prediction accuracy metrics, including RMSE (Root Mean Square Error) and MAE (Mean Absolute Error), which mainly focus on comparing the actual and predicted ratings. MAE presents a holistic view of the recommendation prediction without any bias of extrema in error terms, while RMSE disproportionately penalises large errors and is affected by outliers or bad predictions. Both metrics measure whether the predicted value is close to the actual value but do not account for the order of predictions. We apply the Normalised Discounted Cumulative Gain (NDCG), an average score that measures the consistency between the ranking of predicted ratings and the ground truth for each user. Datasets with implicit feedback aim to predict whether a user interacts with an item. Since the dataset is very sparse and has positive instances only, we uniformly sample five items as negative samples for each clicked item followed by (Wu et al., 2021) and adopt AUC and NDCG to measure the global and personalised ranking accuracy, respectively. AUC measures the global consistency between the ranking of all the predicted user-item interactions and the ground truth, which ranks all the 1's before 0's.

TABLE 4.2. Overall performance comparison with baseline models on the ML-100K and ML-1M datasets. The baseline models are ordered in chronological order from top to bottom. IDCF and GLocal-K were published in 2021.

Model	**ML-100K**			**ML-1M**		
	RMSE ↓	MAE ↓	NDCG ↑	RMSE ↓	MAE ↓	NDCG ↑
SparseFC (Muller et al., 2018)	0.895	0.700	0.905	0.824	0.642	0.928
GC-MC (Berg et al., 2018)	0.905	0.714	0.900	0.832	0.658	0.923
IGMC (Zhang and Chen, 2020)	0.905	0.741	0.899	0.857	0.784	0.903
IDCF-NN (Wu et al., 2021)	0.931	0.732	0.896	0.844	0.663	0.922
IDCF-GC (Wu et al., 2021)	0.905	0.714	0.901	0.839	0.652	0.924
GLocal-K (Han et al., 2021)	<u>0.888</u>	<u>0.695</u>	<u>0.906</u>	<u>0.822</u>	<u>0.641</u>	<u>0.929</u>
SUPER-Rec	**0.720**	**0.551**	**0.990**	**0.591**	**0.409**	**0.996**

4.3 Result

4.3.1 Performance Comparison on Explicit Datasets

We first evaluated our proposed **SUPER-Rec** by comparing its test performance against the baseline models on five rating-based benchmark recommendation datasets. We provide the results separately of MovieLens datasets (ML-100K/-1M) in Table 4.2 and the other datasets (Douban, Flixster and YahooMusic) in Table 4.3 based on data density. Overall, as is illustrated in Table 4.2 and Table 4.3, it can be observed that SUPER-Rec outperforms all these baseline models on each of the rating-based datasets by a large margin even with a simple neural network-based predictor by applying the position-enhanced user/item representation training model. This illustrates the general effectiveness of the the proposed representations learned by SUPER-Rec for capturing the user latent interest/taste patterns and implies the great potential of representation learning from the high-dimensional sparse user-item rating matrix.

More specifically, for the first two datasets in Table 4.2, which involve rating values within 1 to 5 and density from 4.47% (ML-1M) to 6.30% (ML-100K), the performance gain over the best baseline model ranges from 0.168 (ML-100K, compared to GLocal-K) to 0.231 (ML-1M, against GLocal-K) for RMSE, from 0.144 (ML-100K, compared to GLocal-K) to 0.232 (ML-1M, against GLocal-K) for MAE and from 0.067 (ML-1M, against GLocal-K) to 0.084 (ML-100K, compared to GLocal-K) for NDCG. Considering the gap between the first and second best performance scores among the baseline models for each metric measure, the performance gap against our SUPER-Rec model is substantially large. This implies that SUPER-Rec brings a tremendous performance gain in recommender systems. This performance gain still persists as data density drops.

TABLE 4.3. Overall performance comparison with baseline models on the Douban, Flixstser and YahooMusic datasets. The baseline models are ordered in chronological order from top to bottom. IDCF and GLocal-K were published in 2021.

Model	**Douban**			**Flixster**			**YahooMusic**		
	RMSE ↓	MAE ↓	NDCG ↑	RMSE ↓	MAE ↓	NDCG ↑	RMSE ↓	MAE ↓	NDCG ↑
SparseFC (Muller et al., 2018)	0.730	0.569	0.941	0.981	0.699	0.924	34.0	22.7	0.848
GC-MC (Berg et al., 2018)	0.734	0.573	0.938	0.917	0.684	0.886	20.5	16.0	0.844
IGMC (Zhang and Chen, 2020)	0.721	0.591	0.939	<u>0.872</u>	0.697	0.927	<u>19.1</u>	16.3	0.865
IDCF-NN (Wu et al., 2021)	0.738	0.576	0.939	0.910	0.693	0.925	21.5	16.9	0.929
IDCF-GC (Wu et al., 2021)	0.733	0.574	0.940	0.896	<u>0.683</u>	<u>0.928</u>	19.3	<u>15.3</u>	<u>0.932</u>
GLocal-K (Han et al., 2021)	<u>0.720</u>	<u>0.562</u>	<u>0.943</u>	0.980	0.699	0.925	33.8	22.6	0.864
SUPER-Rec	**0.619**	**0.461**	**0.998**	**0.827**	**0.632**	**0.986**	**17.1**	**13.7**	**0.972**

Even for more extremely sparse datasets in Table 4.3, there is still large performance improvement over the best baseline models, while the density of a data matrix ranges from 0.06% (YahooMusic) to 1.52% (Douban) and there are a wide range of rating types such as {0.5, 1.0, 1.5, ..., 5.0} in Flixster and {1, 2, 3, ..., 100} in YahooMusic. For Douban, RMSE and MAE identically drop by 0.101 and NDCG increases by 0.055 against GLocal-K, which is the best performed model among the baselines on Douban. Note that dropping the RMSE and MAE results indicates increasing the rating prediction accuracy. For datasets under 1.0% for density, RMSE drops by 0.045 and 2.0 compared to IGMC on Flixster and YahooMusic, respectively. MAE drops by 0.051 and 1.6 and NDCG increases by 0.058 and 0.040 compared to IDCF-GC on the respective same datasets. Compared to the performance gain from the aforementioned MovieLens datasets, more extremely sparse datasets accomplish relatively minor performance improvement, but produce significant results with regard to comparison with the state-of-the-art baselines, which validate the robustness of the feature representation learning approach and its efficacy for recommendation even on the severely sparse user-item rating data.

4.3.2 Comparison of Rating Prediction Model Variant

In this evaluation, we would like to validate the robustness of our SUPER-Rec as a standalone position-enhanced item representation for recommendation systems. In Section 4.1, we applied a simple neural network as a matrix factorisation model but consider that it is also worth testing with other simple and traditional machine learning classifiers. We compare three machine learning techniques with our SUPER-Rec: k-NN (k-Nearest Neighbour), SVM (Support Vector Machine), and NN (Neural networks) on two explicit datasets, Flixster and YahooMusic, with lower rating density and relatively bad rating prediction performance, as shown in Table 4.4.

TABLE 4.4. Performance comparison with three simple machine learning classification models (k-NN vs. SVM vs. NN) with the SUPER-Rec on Flixster and YahooMusic.

Classifier	**Flixster**			**YahooMusic**		
	RMSE ↓	MAE ↓	NDCG ↑	RMSE ↓	MAE ↓	NDCG ↑
k-NN	0.995	0.826	0.959	18.1	14.4	0.861
SVM	0.851	0.638	**0.987**	17.8	**13.7**	**0.973**
NN	**0.827**	**0.632**	0.986	**17.1**	**13.7**	0.972

As described in Section 4.1.3, we concatenated 1) item, 2) user, and 3) user-item relation representation and applied this concatenated representation as the input for each classifier. First, we use k-NN based on the k(=5) nearest samples, and it results in relatively worse performance in both evaluation metrics

for both datasets, even though it is still competitive and higher than most of the baseline models in Table 4.2 and Table 4.3. Secondly, SVM classifies from a view of the global training samples, and it achieves better performance than KNN in both datasets. It sometimes slightly outperforms NN in NDCG, achieving 0.987 in Flixster and 0.973 in YahooMusic.

Based on this evaluation result, our proposed user/item representation SUPER-Rec has an exceptional capability for predicting the rating on a specific position, even with simple traditional machine learning classifiers. It implies a great potential of our SUPER-Rec as an input representation of the deeper and more complicated deep learning-based matrix factorisation models.

4.3.3 Large-scale Rating Dataset Analysis

In order to demonstrate the effectiveness of SUPER-Rec in a real-world scenario, we test it on the large-scale but sparse ML-10M dataset of density 1.30%, involving a large amount of users and items. The detailed statistics of ML-10M is provided in Table 4.1.

The comparison of RMSE performance with the baseline models is provided in Table 4.5. Among them, there are two newly introduced models, which are not presented in Section 4.4.1. First, I-AutoRec (Sedhain et al., 2015) is a collaborative filtering-based autoencoder model, which represents item feature vectors by projecting each item rating history of interacted users into a low-dimensional feature space and then reconstructs the projected ones to predict the missing values in the item rating history, and MRMA (Li et al., 2017) represents user-item ratings by a mixture of low-rank matrix approximation (LRMA) models with different ranks (MRMA), which goes beyond the conventional matrix approximation with fixed ranks.

TABLE 4.5. Performance comparison of RMSE with baseline models on ML-10M.

Model	**RMSE** $\downarrow$
I-AutoRec (Sedhain et al., 2015)	0.782
GC-MC (Berg et al., 2018)	0.777
SparseFC (Muller et al., 2018)	0.769
MRMA (Li et al., 2017)	0.763
SUPER-Rec	**0.651**

Overall performance improvement of around 0.019 can be observed among the baseline models while starting from 0.782 of I-AutoRec to 0.763 of MRMA. While all of them still remain the high RMSE

results over 0.760, SUPER-Rec outperforms them with a significant drop and reaches the lowest RMSE of 0.651. Therefore, the robustness and the efficacy for recommendation of our proposed feature representation is applicable to even the huge and sparse user-item interaction data, so it can directly be used for the real-world recommendation task with millions of users and items.

4.3.4 Performance Comparison on Implicit Datasets

The previous evaluations are based on explicit datasets, which mainly focus on predicting the user's rating. This evaluation aims to evaluate the item representation capability of our SUPER-Rec in the user's action prediction with implicit feedback datasets. We evaluate the SUPER-Rec on two implicit feedback datasets, Amazon-Beauty and Amazon-Books, indicating user's actions (clicked or not-clicked) using AUC and NDCG metrics. Note that both implicit datasets are large and sparse, Amazon-Books (density of 0.06%) and Amazon-Beauty (density of 0.04%).

TABLE 4.6. Performance comparison of AUC and NDCG with baseline models on the two implicit feedback datasets (Amazon-Beauty and Amazon-Books).

Model	**Amazon-Beauty**		**Amazon-Books**	
	AUC ↑	NDCG ↑	AUC ↑	NDCG ↑
FISM (Kabbur et al., 2013)	0.613	0.678	0.792	0.752
MultVAE (Liang et al., 2018)	0.644	0.679	0.701	0.738
IDCF-NN (Wu et al., 2021)	0.774	0.750	0.920	0.939
IDCF-GC (Wu et al., 2021)	0.791	0.791	0.930	0.946
SUPER-Rec	**0.959**	**0.980**	**0.997**	**0.998**

Surprisingly, as shown in Table 4.6, our SUPER-Rec hugely outperforms the baselines in both AUC and NDCG, including IDCF-GC with graph convolution, which achieves the best performance among all the baseline models. It produces the best result on both datasets of different sparsity in all metrics, i.e. AUC of 0.959 and 0.997, and NDCG of 0.980 and 0.998 on Amazon-Beauty and Amazon-Books. SUPER-Rec outperforms the second-best model by a large margin (0.168/0.189), especially on Amazon-Beauty, with larger size (50K users and 90K items) and richer user's action data (3M actions). The results demonstrate the robustness of the proposed representation, not only in the explicit rating nature but also a simple binary implicit action trends.

4.3.5 Impact of Sparsity Ratios

While previous evaluations have demonstrated the effectiveness of SUPER-Rec on different sparse datasets, we further investigated its efficacy in terms of the sensitivity to sparsity of the same data source. We gradually decrease the training data ratio from 1.0 (full training data) to 0.2 and compared the RMSE, MAE and NDCG test results with three baseline models on ML-100K/-1M and Douban in Figure 4.8, Figure 4.9 and Figure 4.10.

It can be seen that SUPER-Rec and the three baseline models perform almost similarly overall in Figure 4.8 and Figure 4.10. In Figure 4.8, both SUPER-Rec and the three baselines show a similar increasing curve with a similar margin in RMSE and MAE, and they also show a similar decreasing curve in NDCG, but the NDCG of SUPER-Rec decreases only by a small value from 1.0 to 0.2 training ratios, compared to the other baselines. So, it can be interpreted that as NDCG is a measurement for correct ranking prediction, the proposed feature representation is more robust against sparse training data in ranking prediction tasks. In Figure 4.10, the three baseline models also represent increasing curve patterns in RMSE and MAE and decreasing curve patterns in NDCG similar to Figure 4.8. However, there is almost no degradation in the SUPER-Rec evaluations across all metrics in Figure 4.10, illustrating that more powerful robustness of SUPER-Rec can be observed in Douban of lower density compared to ML-100K of higher density.

In Figure 4.9, while the three baseline models depict almost similar performance trends overall, SUPER-Rec presents totally different performance trends, outperforming all of these by a large margin for all metrics at all training ratios. There are the largest margins of 0.38 in RMSE, 0.37 in MAE and 0.1 in NDCG between SUPER-Rec and the best performed baselines at the same ratio 0.2. All of them are larger than the gap between SUPER-Rec and the best scores from baselines with full training data in all metrics, that means as training data changes to more extremely low resource, most recommender systems gradually lose their representation power. However, SUPER-Rec can withstand the degradation of its representation power and even improve it with less training data available.

In Figure 4.9, the baseline models show obvious performance degradation in RMSE, MAE and NDCG as the training data gets more sparse (i.e., with smaller training ratio). The RMSE / MAE results rise by about 0.095 / 0.080 on average of the three baselines between 1.0 and 0.2 training ratios, and the NDCG results drop by about 0.022 on average of these three between 1.0 and 0.2. In contrary, SUPER-Rec illustrates the performance improvement in RMSE when training ratio decrease from 1.0 and reaches

the best RMSE and MAE at 0.4. Even though performance degradation is observed when the training ratio continues decreasing to 0.2, it is still better compared to the ratio of 1.0 or 0.8. On the other hand, SUPER-Rec achieves stable performance in NDCG throughout as the data sparsity changes. We can infer that SUPER-Rec is more robust in recommendation performance and less sensitive to data sparsity, compared to the baseline models. Moreover, SUPER-Rec shows more advantages with sparse datasets, validating its effectiveness of feature representation even with less data available.

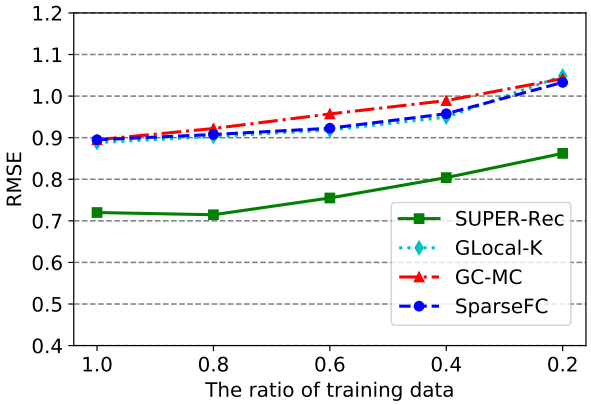
4.3.6 Bilinear Neural Network for Matrix Factorisation

We proved that embedding vectors pre-trained by SUPER-Rec can reach good recommendation performance via simple machine learning-based classification models as well as NNMF. We propose and evaluate a new neural network-based matrix factorisation (MF) model to validate that embedding vectors from SUPER-Rec are of high applicability to a variety of neural network-based MF models.

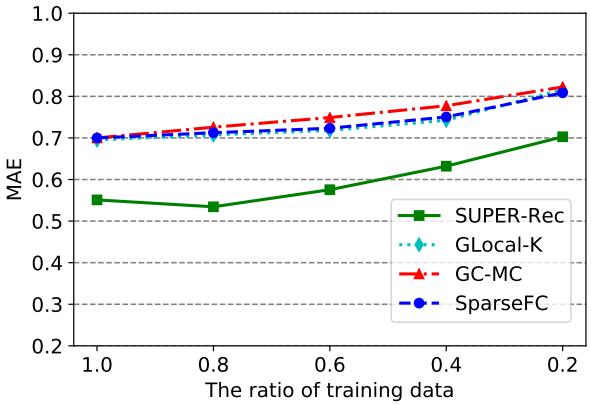
We introduce a bilinear neural network-based MF model (BNMF), which includes a different weight matrix for each rating score, and calculate the probability from each weight matrix bilinearly using user and item vectors. We evaluate how similar BNMF and NNMF produce recommendation performance and whether BNMF outperforms all other baseline models.

We conduct experiments based on the same parameter settings as NNMF to compare the results only depending on different model architectures. In addition, NNMF uses an interaction vector as well as user and item vectors as input by combining the two vectors, and to make the configuration of BNMF very close to that of NNMF, we propose an interaction-enhanced BNMF model (BNMF+). BNMF+ uses an interaction vector to generate an interaction-enhanced user vector by concatenating it with a user vector and feeding the concatenated one into dense layers. We found in comparative experiments that there is no significant performance difference between concatenating with a user vector and with an item vector.

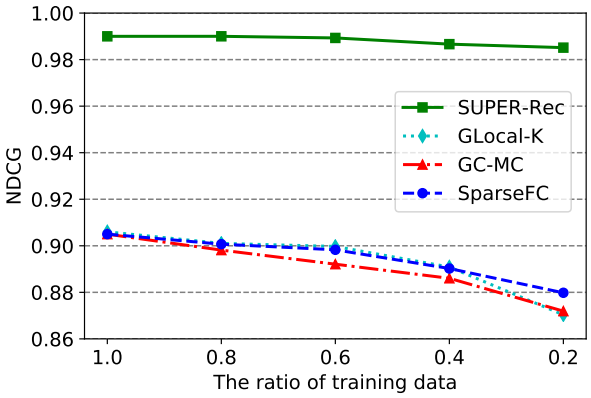
We provide the results separately of MovieLens datasets (ML-100K/-1M) in Table 4.7 and the other datasets (Douban, Flixster and YahooMusic) in Table 4.8 based on data density. The significant performance difference can be observed between the NNMF model and the BNMF and BNMF+ models in Table 4.7 for MovieLens datasets with comparatively high density. Moreover, while BNMF+ outperforms all the other baseline models, the RMSE result of BNMF is worse than that of GLocal-K, the best among the baseline models, found in Table 4.2. The results of the BNMF+ model demonstrate the superior representation power of SUPER-Rec and emphasise the importance of using an interaction



(A) RMSE ↓

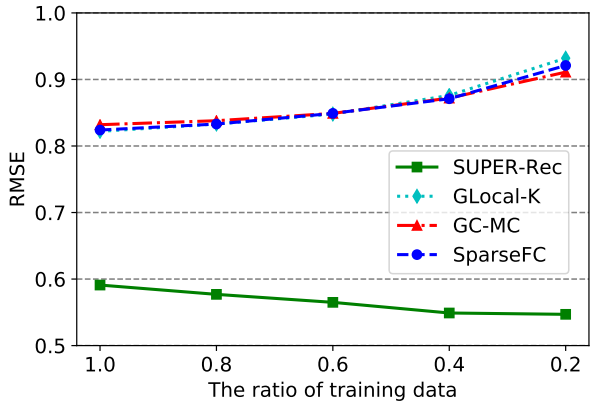


(B) MAE ↓

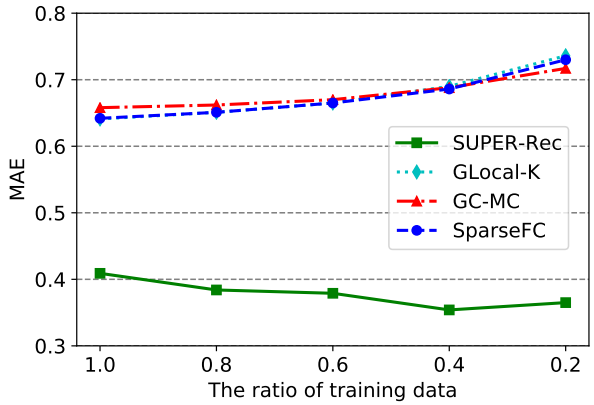


(C) NDCG ↑

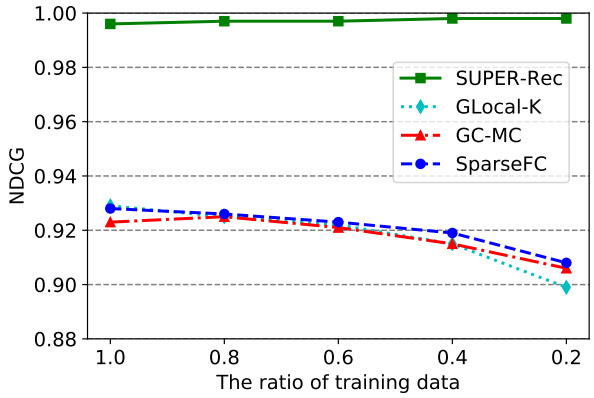
FIGURE 4.8. Performance comparison w.r.t. different sparsity levels ranging from 1.0 to 0.2 at every 0.2 interval between SUPER-Rec and the three baseline models via the three prediction accuracy-based evaluation measurements on ML-100K.



(A) RMSE ↓

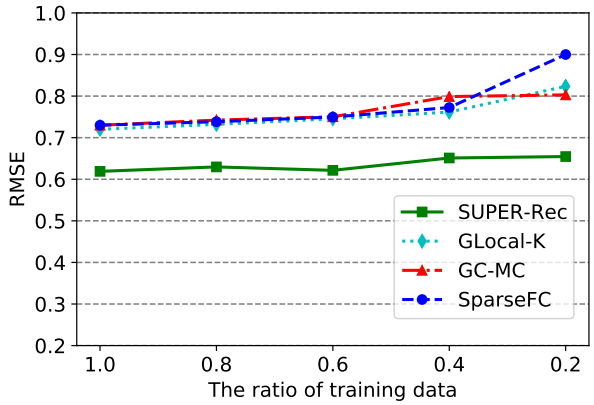


(B) MAE ↓

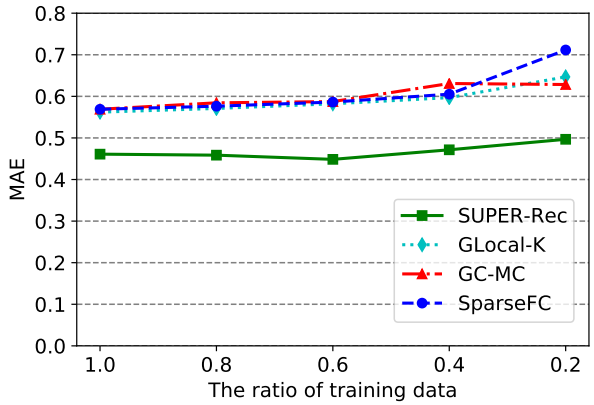


(C) NDCG ↑

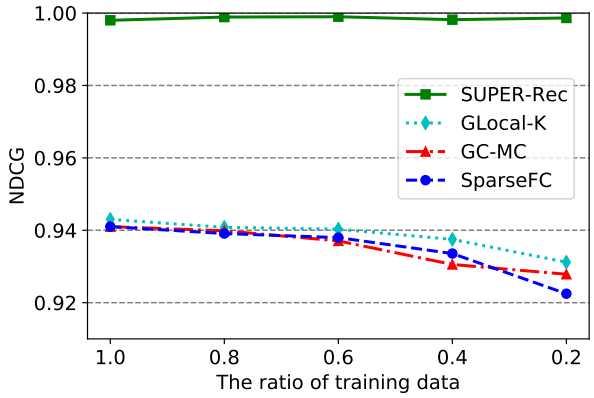
FIGURE 4.9. Performance comparison w.r.t. different sparsity levels ranging from 1.0 to 0.2 at every 0.2 interval between SUPER-Rec and the three baseline models via the three prediction accuracy-based evaluation measurements on ML-1M.



(A) RMSE ↓



(B) MAE ↓



(C) NDCG ↑

FIGURE 4.10. Performance comparison w.r.t. different sparsity levels ranging from 1.0 to 0.2 at every 0.2 interval between SUPER-Rec and the three baseline models via the three prediction accuracy-based evaluation measurements on Douban.

TABLE 4.7. RMSE and NDCG test under different neural network-based MF models on ML-100K and ML-1M. **BNMF** is a newly-introduced bilinear neural network-based MF model, and **BNMF+** indicates that an interaction vector is additionally used within BNMF such as NNMF.

Model	ML-100K		ML-1M	
	RMSE ↓	NDCG ↑	RMSE ↓	NDCG ↑
BNMF	0.927	0.950	0.826	0.978
BNMF+	0.819	0.954	0.804	0.984
NNMF	**0.720**	**0.990**	**0.591**	**0.996**

TABLE 4.8. RMSE and NDCG test under different neural network-based MF models on Douban, Flixster and YahooMusic, which are of relatively low density. **BNMF** is a newly-introduced bilinear neural network-based MF model, and **BNMF+** indicates that an interaction vector is additionally used within BNMF such as NNMF.

Model	Douban		Flixster		YahooMusic	
	RMSE ↓	NDCG ↑	RMSE ↓	NDCG ↑	RMSE ↓	NDCG ↑
BNMF	0.631	0.976	**0.815**	0.977	20.6	0.904
BNMF+	0.742	0.978	0.907	0.941	20.6	0.913
NNMF	**0.619**	**0.998**	0.827	**0.986**	**17.1**	**0.972**

vector in the case of high-density data for better prediction of missing entries in the matrix. On the other hand, NNMF does not always outperform BNMF and BNMF+ on relatively low-density data such that the RMSE of NNMF is worse than that of BNMF on Flixster. For Douban and Flixster, BNMF provides better performance than BNMF+, while it outperforms all other baselines, but for YahooMusic, BNMF produces the same performance as BNMF+ and the comparable results to the baseline models. This implies that using an interaction vector rather makes the performance worse or has no impact on it when data density is relatively low. In conclusion, to determine whether an interaction vector should be used as input depends on how sparse data is, and as the BNMF and BNMF+ models achieve overall higher performance than other baselines, we can lend weight to the argument that embedding vectors from SUPER-Rec should be able to be applied to a wide variety of neural network-based MF models.

4.3.7 Window Size Analysis

We take item representation via direct surrounding items as the representative representation of SUPER-Rec, which can also be treated as the neighbour items selected by window of size 1. In Word2Vec (Mikolov et al., 2013a,b), the effect of word representation learning depends on the context window

size. Larger window sizes tend to capture a wide range of information, which leads embedding similarity to an indicative of the relatedness of the words. Smaller window sizes are more likely to capture about word itself, so the high similarity score between two embeddings indicates that the words can be interchangeable in the same syntactic structure (Lin et al., 2015).

We also explore how the effect of co-occurred neighbour-based item representation learning has varied via changing the window size. We evaluate the performance while increasing the window size from 1 to 5 to find the optimal neighbour number for each dataset. Then, we compare the optimal value with each dataset and analyse the performance trends with respect to the variation of window size.

TABLE 4.9. Performance comparison w.r.t. different context window sizes of 1, 3 and 5 on ML-100K, ML-1M and Douban.

Window Size	**ML-100K**			**ML-1M**			**Douban**		
	RMSE ↓	MAE ↓	NDCG ↑	RMSE ↓	MAE ↓	NDCG ↑	RMSE ↓	MAE ↓	NDCG ↑
1	**0.720**	**0.551**	**0.990**	**0.591**	**0.409**	**0.996**	**0.619**	**0.461**	**0.998**
3	0.811	0.630	0.980	0.637	0.456	0.995	0.704	0.521	**0.998**
5	0.862	0.657	0.972	0.655	0.479	0.994	0.782	0.578	**0.998**

TABLE 4.10. Performance comparison w.r.t. different context window sizes of 1, 3 and 5 on Flixster and YahooMusic.

Window Size	**Flixster**			**YahooMusic**		
	RMSE ↓	MAE ↓	NDCG ↑	RMSE ↓	MAE ↓	NDCG ↑
1	**0.827**	**0.632**	**0.986**	**17.1**	**13.7**	**0.972**
3	0.856	0.661	0.985	17.4	14.0	0.964
5	0.862	0.668	0.984	17.5	14.5	0.961

We illustrate the performance comparison of SUPER-Rec using a context window of size 1, 3 and 5 in Table 4.9 for ML-100K/-1M and Douban and Table 4.10 for Flixster and YahooMusic separately based on data density. As can be seen from those two tables, window of size 1 always achieves the best recommendation results and the performance degrades as the size increases. We find that focusing on a fewer number of neighbours is more effective to represent the item latent features by capturing the most relevant neighbour rating patterns. Moreover, the performance has changed greatly as window size increased under relatively high-density data in Table 4.9. As the density becomes higher, a possibility of being rated in neighbour items increases, so it brings too much information about rating patterns with the co-occurred neighbour items to encode into a low-dimensional feature space. In conclusion, focusing on

modelling the co-occurrence of the direct surrounding items may help render more robust representation and larger context might inevitably introduce some negative noise.

4.3.8 Impact of Embedding Dimension

As the key of the proposed SUPER-Rec lies in a fixed dimension of feature representations, which are pre-extracted from the user-item matrix, we take a more in-depth look into the dimension applied to derive the position-enhanced feature representation on the six rating-based datasets. It can be seen from the overview of performance trends across all datasets that recommendation performance varies differently on these datasets as the dimension of feature representation changes, which implies that different data nature may entail a different amount of latent features needed for conducting the recommendation and thus lead to different optimal dimension sizes for feature representation.

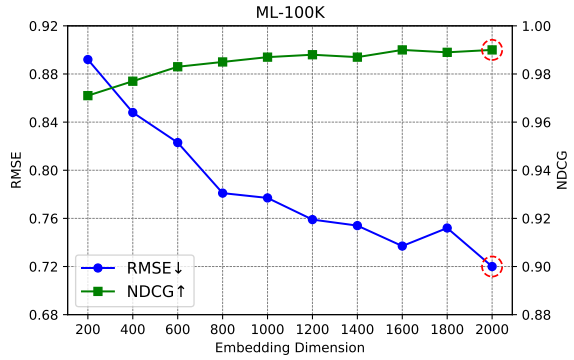


FIGURE 4.11. Performance trend under different sizes of embedding dimension on ML-100K of density 6.30%. The red dotted round lines indicate the best performance results from the optimal dimension size, which is determined by the RMSE result.

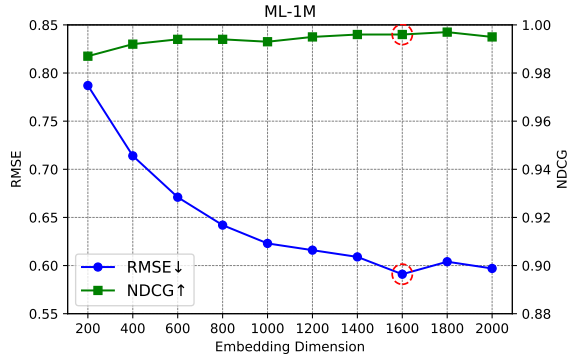


FIGURE 4.12. Performance trend under different sizes of embedding dimension on ML-1M of density 4.47%. The red dotted round lines indicate the best performance results from the optimal dimension size, which is determined by the RMSE result.

The recommendation results of ML-100K/-1M/-10M using different dimension sizes for embedding are shown in Figure 4.11, Figure 4.12 and Figure 4.13. Regarding the RMSE results, ML-100K of density 6.30% and ML-1M of density 4.47% both illustrate an overall pattern that the recommendation performance becomes better as the embedding dimension increases from the range of 200 to 2,000, within which the best performance is produced at the dimension 2,000 and 1,600 respectively. ML-10M of density 1.30% also shows a similar performance trend that the recommendation performance increases in proportion to the embedding dimension, but reaches the optimum more quickly than the aforementioned two datasets at the dimension 1,000. After the optimum is reached, all three datasets consistently keep the optimal value, even though the embedding dimension increases. Note that the red dotted round lines on each figure present the best performance results, which are determined by the RMSE result, and the corresponding dimension is the optimal embedding dimension for each dataset.

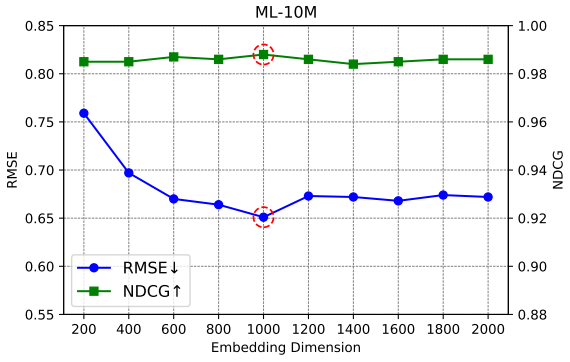


FIGURE 4.13. Performance trend under different sizes of embedding dimension on ML-10M of density 1.30%. The red dotted round lines indicate the best performance results from the optimal dimension size, which is determined by the RMSE result.

In comparison, an opposite overall pattern is illustrated for the other three sparser datasets in Figure 4.14 for Douban of density 1.52%, Figure 4.15 for Flixster of density 0.29% and Figure 4.16 for YahooMusic of density 0.06%, which reach their performance peak at a very small embedding dimension of 100 or 200 within a narrow dimension range of 100 to 1,000, compared to the range of 200 to 2,000 from the aforementioned three datasets. This might be because the sparse data conveys relatively less relation information and patterns available for capturing. However, SUPER-Rec still shows that even without rich patterns available in the given data, the position-enhanced feature representation can possibly lead to good recommendation results. On the other hand, the performance of NDCG remains stable while the dimension changes. This may be attributed to its ranking-based measurement nature, which is comparatively simpler compared to the RMSE for measuring accurate feedback. In fact, the density of Douban is over 1.0% and is even higher than that of ML-10M, so we should identify the factors determining the

optimal dimension size and the overall performance trend under different dimension sizes by analysing the relationship between the data characteristics and the embedding dimension in Section 4.4.10.

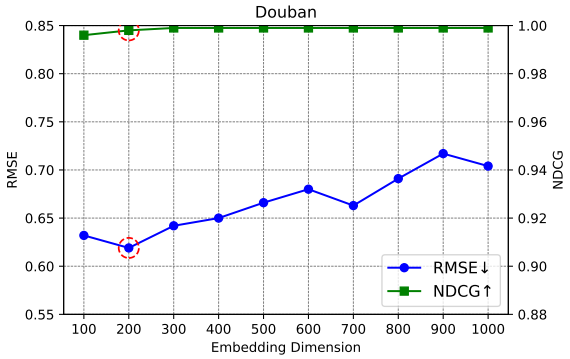


FIGURE 4.14. Performance trend under different sizes of embedding dimension on Douban of density 1.52%. The red dotted round lines indicate the best performance results from the optimal dimension size, which is determined by the RMSE result.

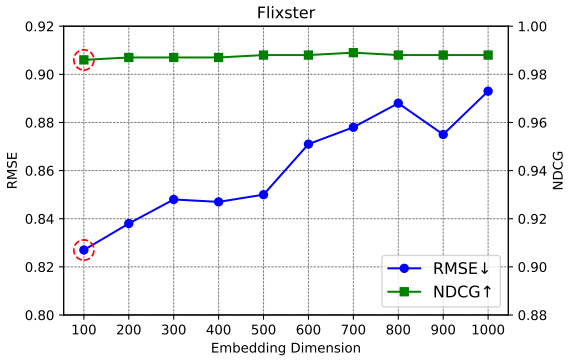


FIGURE 4.15. Performance trend under different sizes of embedding dimension on Flixster of density 0.29%. The red dotted round lines indicate the best performance results from the optimal dimension size, which is determined by the RMSE result.

4.3.9 Comparison of Embedding Training Variants

As mentioned in Section 4.1, we train SUPER-Rec position-enhanced item representation by using a single surrounding neighbour item as an input and a centre item as a target. In this evaluation, we aim to test input variants of the position-enhanced item representation training, which produces better joint learning of position and rating of items. As shown in Table 4.11, *Single* refers to the training model with a single neighbour item as an input and a centre item as a target, while in Table 4.12 the *Double* refers to that with the sum of both left and right neighbour items as an input and a centre item as a target. Those two variant testing would give insight into which input format covers the surrounding neighbour item information for the position-enhanced item embedding training.

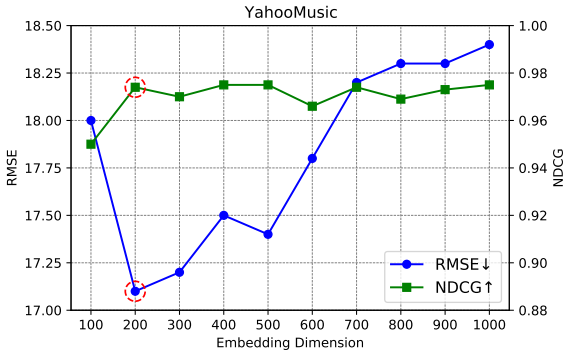


FIGURE 4.16. Performance trend under different sizes of embedding dimension on YahooMusic of density 0.06%. The red dotted round lines indicate the best performance results from the optimal dimension size, which is determined by the RMSE result.

We tested those two variants on the five explicit datasets, shown in Table 4.11 and Table 4.12. The performance gap between the two variants is extremely small, but the type *Single* using a single surrounding neighbour item as input is better than another. This result implies that 1-to-1 neighbour-based training is better than N-to-1 (N=both side, left-right neighbour items).

TABLE 4.11. Performance comparison w.r.t. different embedding methods on the three benchmark datasets: ML-100K, ML-1M and Douban.

Input	**ML-100K**			**ML-1M**			**Douban**		
	RMSE ↓	MAE ↓	NDCG ↑	RMSE ↓	MAE ↓	NDCG ↑	RMSE ↓	MAE ↓	NDCG ↑
Single	**0.720**	**0.551**	**0.990**	**0.591**	**0.409**	**0.996**	**0.619**	**0.461**	**0.998**
Double	0.732	0.568	**0.990**	0.611	0.412	0.995	0.668	0.493	**0.998**

TABLE 4.12. Performance comparison w.r.t. different embedding methods on the two benchmark datasets: Flixster and YahooMusic.

Input	**Flixster**			**YahooMusic**		
	RMSE ↓	MAE ↓	NDCG ↑	RMSE ↓	MAE ↓	NDCG ↑
Single	**0.827**	**0.632**	**0.986**	**17.1**	**13.7**	**0.972**
Double	0.872	0.670	**0.986**	18.8	14.9	0.968

4.3.10 Dimension Pattern Analysis via Dataset Classification

We have explored the optimal dimension for each dataset in Section 4.4.8, which is derived from the assumption that datasets with different compositions will entail a different amount of latent feature information and will be required different dimensions to efficiently represent their information in a

low-dimensional feature space. Moreover, the best recommendation performance for each dataset can be obtained via the optimal dimension found. We further tried to find the relation between data and representation (embedding) dimension, since if we found some relations about those two, when new data is introduced, we can predict its optimal embedding dimension in advance. In reality, it requires a large volume of time and labour to search for the optimal dimension for a given data. However, we cannot find out any relation using the given data statistics only (e.g., the number of users, items and ratings, and density). So, we intend to extract the vital features of data via handcrafted feature engineering, and by classifying datasets in three-dimensional space based on the extracted features, we aim to find out some relations between data and embedding dimension.

The optimal embedding dimension for each dataset is provided in Section 4.4.8, and here we will list the optimal dimensions of six rating-based datasets by a descending order: ML-100K (2,000); ML-1M (1,600); ML-10M (1,000); Douban (200); YahooMusic (200); Flixster (100).

Then, we will list the most vital data features that are capable of presenting clear relations between data and its optimal embedding dimension below. Note that the selected vital features are obtained by a myriad of trial and error:

(1) *# RATINGS PER USER* indicates the average number of ratings for each user.
(2) *# RATINGS PER TRAIN USER* indicates the average number of ratings for each user in the training data.
(3) *# RATINGS PER ITEM* indicates the average number of ratings for each item.
(4) *DENSITY (%)* indicates the density of the whole data.
(5) *TRAIN DENSITY (%)* indicates the density of the training data.
(6) *# USERS / # ITEMS* indicates the ratio of # users and # items. If the value is close to 1, the numbers between the two are very similar.

All six rating-based datasets are assigned in three-dimensional space with x-, y- and z-axis as *# RATINGS PER USER*, *TRAIN DENSITY (%)* and *# USERS / # ITEMS* in Figure 4.17, as *# RATINGS PER TRAIN USER*, *TRAIN DENSITY (%)* and *# USERS / # ITEMS* in Figure 4.18 and as *# RATINGS PER USER*, *DENSITY (%)* and *# USERS / # ITEMS* in Figure 4.19. To represent relations between the data features and the optimal embedding dimensions, we first classify all six datasets into two large and small embedding dimension groups based on the x- and z-axis. It can be observed that as the x value increases and the z value decreases, the dimension of the corresponding dataset becomes higher compared to the

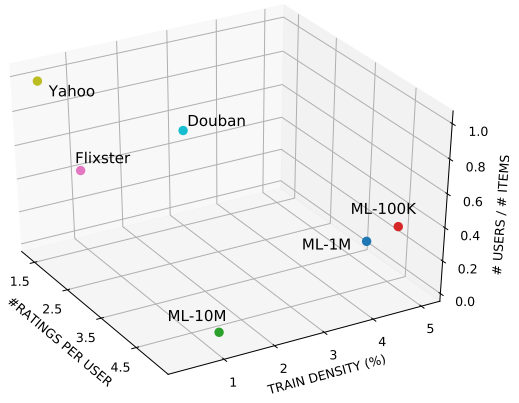


FIGURE 4.17. Dataset classification based on handcrafted feature engineering among the six rating-based benchmark datasets in three-dimensional space (x-axis: # ratings for each user, y-axis: training matrix density(%), z-axis: ratio of # users and # items (when close to 1, the shape of a matrix becomes nearly square).

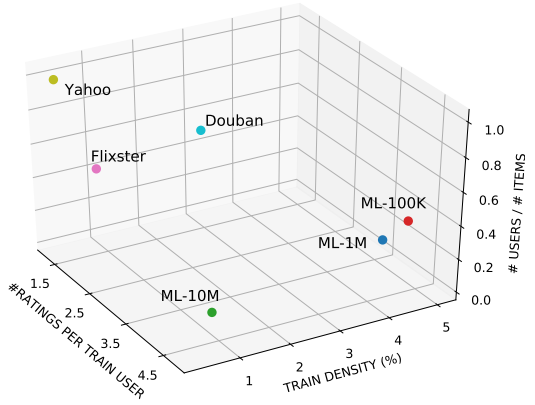


FIGURE 4.18. Dataset classification based on handcrafted feature engineering among the six rating-based benchmark datasets in three-dimensional space (x-axis: # ratings for each training user, y-axis: training matrix density(%), z-axis: ratio of # users and # items (when close to 1, the shape of a matrix becomes nearly square).

other datasets under the opposite conditions. Moreover each group shows that when both y and z values are smaller, the dimension tends to be lower compared to other dimensions in the same group.

All six rating-based datasets are assigned in three-dimensional space with x-, y- and z-axis as *# RATINGS PER TRAIN USER*, *# RATINGS PER USER* and *# USERS / # ITEMS* in Figure 4.20 and as *# RATINGS PER TRAIN USER*, *# RATINGS PER ITEM* and *# USERS / # ITEMS* in Figure 4.21. To demonstrate that the optimal value of embedding dimension might be affected by the latent features of data, we first classify all six datasets based on the x- and z-axis into two groups, where one shows a large embedding

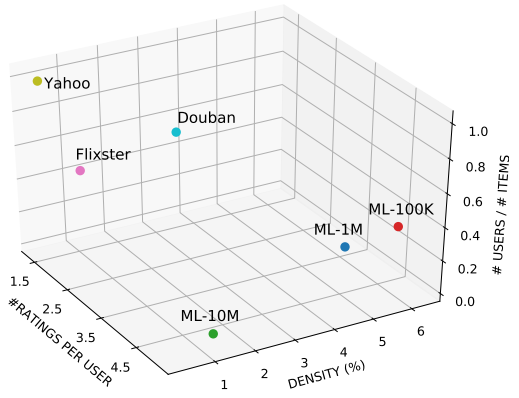


FIGURE 4.19. Dataset classification based on handcrafted feature engineering among the six rating-based benchmark datasets in three-dimensional space (x-axis: # ratings for each user, y-axis: whole matrix density(%), z-axis: ratio of # users and # items (when close to 1, the shape of a matrix becomes nearly square).

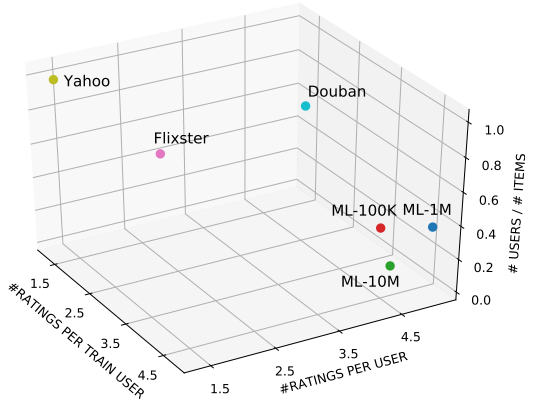


FIGURE 4.20. Dataset classification based on handcrafted feature engineering among the six rating-based benchmark datasets in three-dimensional space (x-axis: # ratings for each training user, y-axis: # ratings for each user, z-axis: ratio of # users and # items (when close to 1, the shape of a matrix becomes nearly square).

dimension with the high x and the low z values, and conversely the other shows a small embedding dimension with the low x and the high z values. Meanwhile, y-axis can also involve in discriminating the datasets by the size of embedding dimension, showing that datasets of a large dimension tend to lean towards a high place in y-axis. However, different from the aforementioned three cases, these suggested feature combinations are not capable of further fine-grained classification in the two separate groups.

In Figure 4.22, all six rating-based datasets are projected in three-dimensional space with x-, y- and z-axis as *# RATINGS PER USER*, *DENSITY (%)* and *# RATINGS PER ITEM*. To show that the optimal

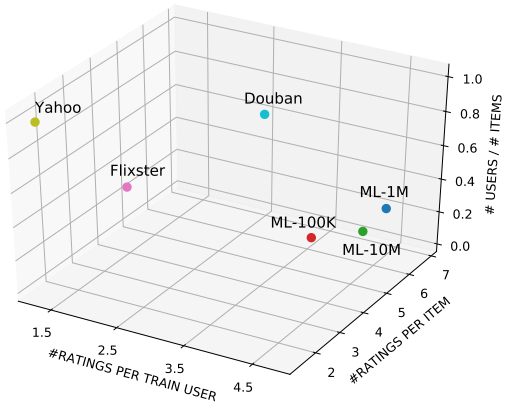


FIGURE 4.21. Dataset classification based on handcrafted feature engineering among the six rating-based benchmark datasets in three-dimensional space (x-axis: # ratings for each training user, y-axis: # ratings for each item, z-axis: ratio of # users and # items (when close to 1, the shape of a matrix becomes nearly square).

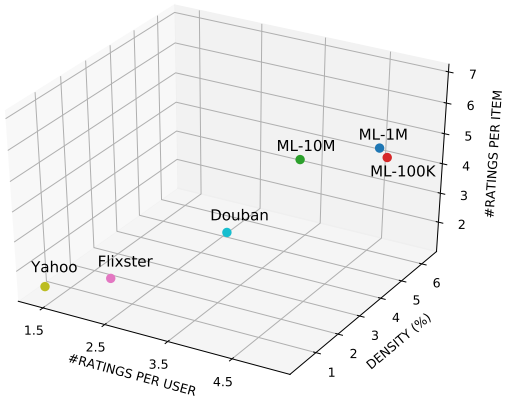


FIGURE 4.22. Dataset classification based on handcrafted feature engineering among the six rating-based benchmark datasets in three-dimensional space (x-axis: # ratings for each user, y-axis: whole matrix density(%), z-axis: # ratings for each item.

embedding dimension for each dataset can be defined by the specific combination of the data features, we classify the datasets into two groups divided by the size of embedding dimension based on either the values on both x- and z-axis or the values on both y- and z-axis. We can find that as the x and z values or the y and z values increase together, the datasets whose optimal embedding dimensions are relatively large are gathered around the corresponding area. It is almost impossible to classify one step further the datasets based on their optimal dimension sizes in each large and small dimension area, though. However, since the dimension difference between the previously classified groups is very clear, we can predict the scope of the optimal dimension of newly incoming data according to its assigned area, which will render us to do efficient optimal search.

To sum up, the optimal embedding dimension for a rating-based dataset demonstrates the correlation with the combination of three types of data features, each of which corresponds to x- or y -or z-axis in three-dimensional space, and is able to be predicted for a new dataset via comparing its feature combination with the dimension-feature correlation.

4.4 Summary

In this chapter, we have introduced a position-enhanced user/item representation SUPER-Rec for the recommendation that leverages the positions and ratings of surrounding neighbour items. The SUPER-Rec significantly outperformed in all RMSE/MAE/NDCG/AUC metrics against state-of-the-art Matrix Completion models on most recommendation benchmark datasets, with high or low density, small or large user/item/rating size, and explicit or implicit user's action. The pre-trained SUPER-Rec embedding for each dataset will be released to the public and let researchers/developers use it as an input embedding for their deeper and complex recommendation model.

CHAPTER 5

Conclusion

In this dissertation, we have focused on how to extract the key features of users and items in the rating matrix to capture their features in a low-dimensional vector and also how to create embeddings that well represent the characteristics of users and items by exploring what kind of user/item information to use in the matrix. First, we have tried to solve the data sparsity problem, which is the most common problem in collaborative filtering and the other problem that there is no recent research on how to efficiently extract the latent features of users and items in a general situation given only sparse data. More importantly, we have pointed out that most matrix completion techniques have mainly focused on semantic similarity between users and items, but have not addressed the fundamental problem in matrix completion, neglecting the position of user/item/rating. We have thought that position/location is the most important nature of a matrix as a specific point can be presented based on the positions of its row and column in the matrix, so we have focused on how to deal with the position/location.

We have discussed a Global and Local Kernel-based matrix completion framework, called GLocal-K in detail, which aims to generalise and represent a high-dimensional sparse user-item matrix entry into a low-dimensional space with a small number of important features by using only given sparse data without relying on side information. We introduced GLocal-K for recommender systems, which takes full advantage of both a local kernel at the pre-training stage and a global kernel at the fine-tuning stage for capturing and refining the important characteristic features of the sparse rating matrix under an extremely low resource setting. GLocal-K achieved the outperformed results even with no side information over all the baseline models. We can infer that combining the local-global kernels improves the feature extraction performance against a high-dimensional sparse rating matrix. Further, we tried to vary the training ratio of ML-100K and Douban to validate the superior effectiveness of cooperation by local and global kernels of GLocal-K under more scarce data sources. Then, we explored the optimal number of epochs for pre-training, where we find that pre-training benefits GLocal-K to achieve better performance on all three datasets, and having more item numbers with lower density may lead to less

pre-training for optimal performance. Overall, it can be seen that applying the local and global kernels in the two-stage training process is effective for feature extraction in a sparse user-item matrix with no side information.

However, we think that the fundamental problem of matrix completion has not been solved, as previous studies have not dealt with the position of a matrix, even though it is the essential nature of a matrix. We tried to explore the best way to capture and apply the position information in the matrix, and finally propose a position-enhanced user/item representation training model for recommendation, SUPER-Rec. The model learns an item representation based on item's interactions with its surrounding neighbour items leveraging their position and rating information. The SUPER-Rec recommendation consists of 3 main stages: User-Item Matrix Positioning to form the item-context training corpus, SUPER-Rec Training to learn an item representation using the fixed surrounding item positions correlated with user feedback taste and Matrix Completion with SUPER-Rec to conduct the recommendation. The model with the SUPER-Rec representation significantly outperforms all baselines on all five rating datasets with a variety of rating ranges by an enormously large margin. Note that we applied a simple neural network as a rating prediction model. The result validates the robustness of the position-enhanced representation and its efficacy for recommendation in various rating ranges of user-item data. Moreover, we validate the robustness of the SUPER-Rec as a stand-alone representation for recommendation systems by comparing three simple machine learning techniques (kNN, SVM and NN) using the SUPER-Rec. We evaluate the SUPER-Rec on the large-scale and sparse rating dataset, and even on two implicit feedback datasets to confirm the item representation capability in the rating prediction with a million of users and the user's action (e.g., click/view) prediction. Lastly, we examined the efficacy of SUPER-Rec in terms of the sensitivity to training data sparsity. The model achieves stable performance overall as the training data gradually decreased, which implies that SUPER-Rec is more robust in recommendation performance and less sensitive to data sparsity than the baseline models. Overall, we can summarise that SUPER-Rec significantly outperformed by an extremely large margin in all RMSE/MAE/NDCG/AUC metrics (i.e., in any rating/ranking prediction accuracy measure) against state-of-the-art matrix completion models on most recommendation benchmark datasets with high or low density, small or large user/item/rating size, and explicit or implicit user's action.

5.1 Future Work

In this research, we have focused on learning a user/item feature representation for a sparse user-item matrix via the kernelised feature extraction technique and the position-enhanced feature learning using surrounding neighbour information. However, we considered the only recommendations for rating prediction in a fixed user-item matrix. Due to the stand-alone and generally applicable characteristics of our position-enhanced representation (SUPER-Rec), we consider as a future work to apply the SUPER-Rec to various fields such as news recommendation and time series predictive analysis, which are widely used in current recommendation systems and require better systems, even though the nature of data is different. News contains a variety of textual information such as titles, articles, bodies and categories, and the text information of news is usually encoded by feeding pre-trained word embeddings to the multi-head self-attention layer, which is a particular case of attention mechanism. Meanwhile, in order to apply our representation learning model to the news recommendation model, we can give a relative position to the user's browsed news and give another representation of the news based on the interaction between the surrounding news. By combining the existing text-based news representation and the position-enhanced news representation in which the interaction between the surrounding neighours is captured, we can give a new insight into news recommender systems surpassing the existing models. Moreover, we can highlight that our position-enhanced representation has a great potential to be applied to any recommendation problem.

Bibliography

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2006. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19.

Rianne van den Berg, Thomas N Kipf, and Max Welling. 2018. Graph convolutional matrix completion. *KDD Deep Learning Day*.

Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems*, 46:109–132.

John S. Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, page 43–52. Morgan Kaufmann Publishers Inc.

Emmanuel J Candès and Benjamin Recht. 2009. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772.

Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 335–344.

Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 27–34.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29.

Joaquin Delgado and Naohiro Ishii. 1999. Memory-based weighted majority prediction. In *SIGIR Workshop Recomm. Syst. Citeseer*, page 85. Citeseer.

Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. 2012. The yahoo! music dataset and kdd-cup'11. In *Proceedings of KDD Cup 2011*, pages 3–18. PMLR.

Gintare Karolina Dziugaite and Daniel M Roy. 2015. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*.

Simon Funk. 2006. Netflix update: Try this at home.

Theodoros Giannakopoulos, Aggelos Pikrakis, and Sergios Theodoridis. 2008. A novel efficient approach for audio segmentation. In *2008 19th International Conference on Pattern Recognition*, pages

1–4. IEEE.

Elie Guàrdia-Sebaoun, Vincent Guigue, and Patrick Gallinari. 2015. Latent trajectory modeling: A light and efficient way to introduce time in recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 281–284.

Soyeon Caren Han, Taejun Lim, Siqu Long, Bernd Burgstaller, and Josiah Poon. 2021. Glocal-k: Global and local kernels for recommender systems. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3063–3067.

Attribution F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4).

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517.

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*, pages 355–364.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. 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*, pages 639–648.

Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. Nais: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2354–2366.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Thomas Hofmann. 2004. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115.

Tianlin Huang, Rujie Zhao, Lvqing Bi, Defu Zhang, and Chao Lu. 2021. Neural embedding singular value decomposition for collaborative filtering. *IEEE Transactions on Neural Networks and Learning Systems*.

Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 135–142.

Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 659–667.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Neil D Lawrence and Raquel Urtasun. 2009. Non-linear matrix factorization with gaussian processes. In *Proceedings of the 26th annual international conference on machine learning*, pages 601–608.

Joonseok Lee, Seungyeon Kim, Guy Lebanon, Yoram Singer, and Samy Bengio. 2016. Llorma: Local low-rank matrix approximation. *Journal of Machine Learning Research*, 17(15):1–24.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems*, 27.

Dongsheng Li, Chao Chen, Wei Liu, Tun Lu, Ning Gu, and Stephen Chu. 2017. Mixture-rank matrix approximation for collaborative filtering. *Advances in Neural Information Processing Systems*, 30.

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*, pages 811–820.

Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. 2016. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM conference on recommender systems*, pages 59–66.

Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*, pages 689–698.

Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. 2015. Unsupervised POS induction with word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Xinyue Liu, Chara Aggarwal, Yu-Feng Li, Xiaugnan Kong, Xinyuan Sun, and Saket Sathe. 2016. Kernelized matrix factorization for collaborative filtering. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 378–386. SIAM.

Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *ICLR Workshop*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Bradley N Miller, Istvan Albert, Shyong K Lam, Joseph A Konstan, and John Riedl. 2003. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266.

Federico Monti, Michael M Bronstein, and Xavier Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *Proceedings of the 31st International Conference on Neural*

Information Processing Systems, pages 3700–3710.

Lorenz Muller, Julien Martel, and Giacomo Indiveri. 2018. Kernelized synaptic weight matrices. In *International Conference on Machine Learning*, pages 3654–3663. PMLR.

Atsuyoshi Nakamura and Naoki Abe. 1998. Collaborative filtering using weighted majority prediction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, page 395–403. Morgan Kaufmann Publishers Inc.

Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th international conference on data mining*, pages 497–506. IEEE.

Arkadiusz Paterek. 2007. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8.

Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. 2015. Collaborative filtering with graph information: Consistency and scalable methods. *Advances in neural information processing systems*, 28.

Ahmed Rashed, Josif Grabocka, and Lars Schmidt-Thieme. 2019. Attribute-aware non-linear co-embeddings of graph features. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 314–321.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798.

Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. 2002. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*, pages 111–112.

Jonathan Strahl, Jaakko Peltonen, Hirsohi Mamitsuka, and Samuel Kaski. 2020. Scalable probabilistic matrix factorization with graph-based priors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5851–5858.

Florian Strub, Romaric Gaudel, and Jérémie Mary. 2016. Hybrid recommender system based on autoencoders. In *Proceedings of the 1st workshop on deep learning for recommender systems*.

Thanh Tran, Kyumin Lee, Yiming Liao, and Dongwon Lee. 2018. Regularizing matrix factorization with user and item embeddings for recommendation. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 687–696.

A Ugla, Dhuha J Kamil, Hassan J Khaudair, et al. 2020. Interpretable recommender system with heterogeneous information: A geometric deep learning perspective. *International Journal of Mechanical and Production Engineering Research and Development (IJMPERD)*, 10(3):2411–2430.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12).

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*, pages 1235–1244.

Jianling Wang and James Caverlee. 2019. Recurrent recommendation with local coherence. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 564–572.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019a. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174.

Yaqing Wang, Chunyan Feng, Caili Guo, Yunfei Chu, and Jenq-Neng Hwang. 2019b. Solving the sparsity problem in recommendations via cross-domain item embedding based on co-clustering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 717–725.

Hao Wu, Qimin Zhou, Rencan Nie, and Jinde Cao. 2020. Effective metric learning with co-occurrence embedding for collaborative recommendations. *Neural Networks*, 124:308–318.

Qitian Wu, Yirui Gao, Xiaofeng Gao, Paul Weng, and Guihai Chen. 2019. Dual sequential prediction models linking sequential recommendation and information dissemination. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 447–457.

Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Junchi Yan, and Hongyuan Zha. 2021. Towards open-world recommendation: An inductive model-based collaborative filtering approach. In *International Conference on Machine Learning*, pages 11329–11339. PMLR.

Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ninth ACM international conference on web search and data mining*, pages 153–162.

Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. 2019. Deep item-based collaborative filtering for top-n recommendation. *ACM Transactions on Information Systems (TOIS)*, 37(3):1–25.

Zhe Yang, Bing Wu, Kan Zheng, Xianbin Wang, and Lei Lei. 2016. A survey of collaborative filtering-based recommender systems for mobile internet applications. *IEEE Access*, 4:3273–3287.

Guijuan Zhang, Yang Liu, and Xiaoning Jin. 2020. A survey of autoencoder-based recommender systems. *Frontiers of Computer Science*, 14(2):430–450.

Muhan Zhang and Yixin Chen. 2020. Inductive matrix completion based on graph neural networks. In *International Conference on Learning Representations*.

Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. 2018. Spectral collaborative filtering. In *Proceedings of the 12th ACM conference on recommender systems*, pages 311–319.

Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A neural autoregressive approach to collaborative filtering. In *International Conference on Machine Learning*, pages 764–773. PMLR.

Tinghui Zhou, Hanhuai Shan, Arindam Banerjee, and Guillermo Sapiro. 2012. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *Proceedings of the 2012 SIAM international Conference on Data mining*, pages 403–414. SIAM.

Fuzhen Zhuang, Dan Luo, Nicholas Jing Yuan, Xing Xie, and Qing He. 2017. Representation learning with pair-wise constraints for collaborative ranking. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 567–575.