Proxy-based Item Representation for Attribute and Context-aware Recommendation

Jinseok Seol
jamie@europa.snu.ac.kr
Seoul National University
Seoul, Republic of Korea

Minseok Gang
alstjr3754@europa.snu.ac.kr
Seoul National University
Seoul, Republic of Korea

Sang-goo Lee
sglee@europa.snu.ac.kr,sglee@intellisys.co.kr
Seoul National University, IntelliSys Co., Ltd.
Seoul, Republic of Korea

Jaehui Park
jaehui@uos.ac.kr
University of Seoul
Seoul, Republic of Korea

ABSTRACT

Neural network approaches in recommender systems have shown remarkable success by representing a large set of items as a learnable vector embedding table. However, infrequent items may suffer from inadequate training opportunities, making it difficult to learn meaningful representations. We examine that in attribute and context-aware settings, the poorly learned embeddings of infrequent items impair the recommendation accuracy. To address such an issue, we propose a proxy-based item representation that allows each item to be expressed as a weighted sum of learnable proxy embeddings. Here, the proxy weight is determined by the attributes and context of each item and may incorporate bias terms in case of frequent items to further reflect collaborative signals. The proxy-based method calculates the item representations compositionally, ensuring each representation resides inside a well-trained simplex and, thus, acquires guaranteed quality. Additionally, that the proxy embeddings are shared across all items allows the infrequent items to borrow training signals of frequent items in a unified model structure and end-to-end manner. Our proposed method is a plug-and-play model that can replace the item encoding layer of any neural network-based recommendation model, while consistently improving the recommendation performance with much smaller parameter usage. Experiments conducted on real-world recommendation benchmark datasets demonstrate that our proposed model outperforms state-of-the-art models in terms of recommendation accuracy by up to 17% while using only 10% of the parameters.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Proxy-based Item Representation, Parameter-efficient Recommendation, Attribute and Context-aware Sequential Recommendation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WSDM '24, March 4–8, 2024, Mérida, Mexico
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0371-3/24/03…$15.00
https://doi.org/10.1145/3616855.3635824

ACM Reference Format:
Jinseok Seol, Minseok Gang, Sang-goo Lee, and Jaehui Park. 2024. Proxy-based Item Representation for Attribute and Context-aware Recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining (WSDM '24), March 4–8, 2024, Merida, Mexico.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3616855.3635824

1 INTRODUCTION

Recommender systems have evolved from collaborative filtering (CF) [36] to machine learning [14, 28, 34] and deep learning-based models [2, 43]. In model-based CF [20], users and items are represented as latent vectors via learnable embedding matrix, which is implemented as a full look-up table, where each row corresponds to a unique user or item vector. Such vector representation approach can transform the recommendation problem into a set of vector arithmetic, assuming that the latent vectors are well-trained through the training data distribution [41]. To train the representations effectively and achieve higher recommendation performance, recent studies have introduced deep learning-based models [16, 42, 48] and sequential recommendation models [18, 30, 32, 40] as well as attribute [15, 54] and context-aware [14, 33] models.

The concept of item embedding matrix also appears in the Natural Language Processing (NLP) field to train the distributional semantics of words [25]. However, in contrast to the data distributions in NLP, recommendation datasets exhibit distinct and unique characteristics, which incur two significant limitations that need to be addressed. Firstly, the item frequency in recommendation datasets follows a long-tail distribution [51]. The item embedding matrix, which is unaware of such data distribution, cannot accurately reflect the proper training signals for the infrequent items [26, 27], leading to degraded recommendation performance [1]. Secondly, the number of items may increase indefinitely. For a real-world recommendation scenario, a large set of new items is consistently added to the system [11, 31] and the item embedding matrix, where most of the model parameters are stored [1], must scale up accordingly to accommodate them. For these problems being the case, it is necessary to devise a new methodology that can effectively replace the item embedding matrix and resolve such limitations.

We further investigate the problem of infrequent item embeddings thoroughly. First of all, during the training process of a recommendation model, items appear in three different scenarios: a user profile that is represented as a list of interacted items, a target item for the system to predict, and a set of negative items that the user has not interacted obtained via negative sampling. It is evident that the parameter update for a specific item embedding, except for that of negative sampled items, is directly proportional to the item's occurrence frequency. Consequently, infrequent items do not receive as many training opportunities as frequent items do and, thus, their embeddings hardly converge to their optimal states throughout the training. We have conducted an empirical analysis of such phenomena under attribute and context-aware settings [18, 33]. Specifically, we have replaced a portion of infrequent items with a shared unknown token and measured the model performance. In Table 1, when the infrequent items are replaced by the unknown token, which can be seen as a removal process, the recommendation accuracy in terms of NDCG@10 increases. Such result clearly demonstrates that the poorly learned infrequent item embeddings harm the overall performance as mentioned earlier. However, such improvement comes at the cost of losing global coverage of recommended items, which accounts for item diversity, and highlights that the insufficient training signals for infrequent items need to be addressed.

Previous studies that aim to resolve the aforementioned issues can be divided into two branches. Firstly, to mitigate the negative impact of infrequent item embeddings on the model performance while reducing the parameter usage, [7] employed a shared embedding through clustering and [27] reduced the embedding dimension of infrequent items. However, these methods are not suitable for datasets where content information (*i.e.*, item attributes and contexts) is crucial (*e.g.*, Fashion domain, showing large performance gap when content information is not utilized). Secondly, [5, 44, 55] proposed methods that allow the embeddings of infrequent items to mimic those of frequent items using their content information. However, they have separate model structures (*e.g.*, mimicking network) and learning strategies (*e.g.*, warm-up stage), which makes it difficult to train in an end-to-end fashion, resulting in unstable hyper-parameter tuning.

In this paper, we propose a novel *proxy-based item representation* model that represents each item as a weighted sum of learnable proxy embeddings by leveraging content information, while incorporating the learning process of both frequent and infrequent items into a single framework in an end-to-end manner. Generally, a *proxy* refers to a model or a vector learned in place of the original training objective to improve the training efficiency or performance in deep learning [3, 26]. We reduce the item embedding matrix into two linearly combinable components: the proxy weighting network and the proxy embedding. Our model represents an item as the weighted sum of learnable proxy embeddings, where the weights are primarily determined by content information. As opposed to the infrequent items, frequent items possess abundant collaborative signals. To further incorporate such signals whenever available, we add learnable bias terms to the proxy weights. By such means, the model can learn hybrid collaborative signals in a single representation space that encompasses not only item ID-to-ID but also ID-to-content and content-to-content relations.

The core concept of our proposed model is illustrated in Figure 1. We apply a softmax function on the proxy weights to enforce the weights sum up to 1. This approach ensures each item representation resides inside a simplex where vertices are well-trained proxy embeddings. Moreover, since the gradient is formed in a unit of proxy embeddings that are shared across all items, the training becomes stable and fast while allowing infrequent items to borrow the training signal of frequent items. Additionally, that newly added item can be computed compositionally via proxy embeddings based on their content information prevents the indefinite increase of parameters, resulting in parameter efficiency. As a result, our method alleviates the inadequate parameter updates of those items that exist in long-tail distribution, leading to performance improvement with much smaller parameter usage.

Table 1: Performance and diversity comparison of infrequent item removal ratio on Fashion

Model	Removal	#params (M)	NDCG@10	Diversity
SASRec++	0%	45.8	37.9%	15.0%
	25%	35.1	39.1%	14.9%
	50%	24.5	39.2%	14.8%
	75%	13.9	40.2%	13.1%
	90%	7.5	41.6%	11.6%
with **PIR**	0%	25.1	42.6%	16.2%
	25%	19.7	42.5%	15.3%
	50%	14.4	42.5%	16.2%
	75%	9.1	42.4%	16.2%
	90%	5.9	42.4%	16.0%

We conduct experiments on real-world recommendation benchmark datasets, namely Amazon Review datasets [24] and MovieLens dataset. Each dataset has its distinct data distribution and item attributes, along with context and sequential information. Our proposed method is a plug-and-play model that can replace the item encoding layer of any neural network-based recommendation model, and the experimental results show that leveraging our model consistently improves the performance across all datasets.

2 RELATED WORK

Our model is inspired by three research branches: attribute and context-aware sequential recommendation (ACSR) models [8, 29, 30, 54], parameter-efficient recommendation models [17, 23, 27], and proxy learning models [10, 19] involving clustering techniques in self-supervised learning [3, 21].

The most important feature in the recommendation is the user-item interaction history. However, in certain datasets where interaction data is sparse or item attributes are crucial, leveraging content and contextual information becomes vital [14, 37, 44, 54]. For example, the current state-of-the-art ACSR model CARCA [33] achieves a performance increase of up to 50% in terms of recommendation accuracy when compared to other baseline models that do not reflect all of the following: the attribute, the context, or the sequential information. Furthermore, in the cold-start scenarios where the interaction data are extremely rare, [5, 44, 55] proposed two-structured methods that aim to mimic the embeddings of frequent items by employing content information to boost model performance.

To address the enormous training parameter issue [1], studies for parameter-efficient recommendation models have been conducted

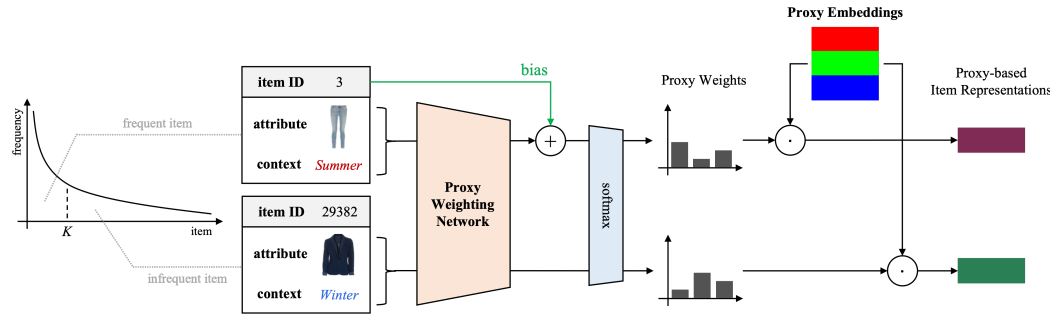

Figure 1: Proxy-based item representation model. Items are represented as a weighted sum of learnable proxy embeddings.

recently [1, 17, 23, 27, 49, 53]. Most studies focus on devising new methodologies to encode each item into a promising vector representation with fewer parameters [7, 17, 27, 53] or reduce the model size without hurting the performance [38, 49]. In the mixed-dimension model [27], the item embedding matrix is split into two parts: frequent and infrequent items, where the embeddings of the latter are factorized into low-rank matrices. Such method has the potential to alleviate the negative impact of infrequent item embeddings on the model performance while reducing the overall parameter usage. In the hash-based method [17], the embedding matrix is compressed through multiple hash functions, representing the item embeddings as the combination of hash values. The clustering-based method of [7] also aims to resolve inherent issues of the embedding matrix by using the shared embeddings. However, such methods does not reflect essential recommendation properties, such as the attributes of items and the recommendation contexts, limiting performance improvement in certain datasets that heavily resort to item content information (*e.g.*, Fashion domain).

The concept of utilizing proxies has been applied and verified in many studies from various fields with similar motivations: infrequently occurring objects borrowing the training signals from frequently occurring objects. Among numerous recommendation models, ProxySR [10] employs user proxy embeddings to augment information for item sequences with a relatively short length in a session-based setting. However, ProxySR explicitly selects one proxy user at a time, which is very different from our proposed model that represents items by combining multiple proxy item embeddings. Concretely, proxies in ProxySR represent prototypical users, while proxies in our model represents cluster centroids of items that serve as necessary components for item representations. Despite such differences, ProxySR successfully demonstrates that the proxy embeddings serve as an external memory [12] of collaborative signals [13, 22] and provide a performance improvement.

In computer vision, proxies are also utilized as cluster centroids to improve training efficiency and performance in contrastive learning, namely self-supervised learning [3, 4, 9, 21, 52] and metric learning for image retrieval [19, 26, 45, 50]. Contrastive learning involves positive and negative sampling, where the complexity of image pairs or triplets becomes enormous [19]. This leads to inconsistent parameter updates since some positive samples may starve by chance. To address such problem, training is performed between proxies rather than image samples. In this setting, a proxy represents a centroid of an unsupervised cluster or a representative vector of a supervised labeled group. The individual images are then trained via their corresponding proxy. Since the number of proxies is much smaller than the number of images, the network converges faster, and the training becomes more stable while improving the overall performance [19]. Regarding the recommendation, the infrequent items often undergo insufficient parameter updates, corresponding to the aforementioned inconsistent parameter update of positive samples in the contrastive learning. Therefore, employing proxies that behave similarly to cluster centroids can alleviate such training opportunity starvation of infrequent items.

3 MODEL

3.1 Problem Definition

Let U and I be the set of all users and all items, respectively. The *user item sequence* S_u of a user $u \in U$ is defined as $S_u = (i_1, i_2, ..., i_{|S_u|})$ where $1 \le t \le |S_u|$ and $i_t \in I$. We denote a subsequence that uses only the first t' items as $S_{u,(t \le t')}$. We define negative items as $I_u^- = I \setminus I_u^+$ where I_u^+ is a set of items constituting S_u. The goal of *sequential recommendation* (SR) is to predict subsequent items with a given user item sequence [18]. For every $u \in U$ and each $1 \le t' \le |S_u| - 1$, the objective is to maximize $P(i_{t'+1}|S_{u,(t \le t')})$, when compared to negative items $N_u = \{i_j^-\}_{j=1}^{|N_u|}$ that are randomly sampled from I_u^-. We assume that each item has auxiliary information in addition to its interaction history, which is referred to as *item attributes*. The item attributes exist in various forms, such as images, titles, descriptions, keyword tags, etc. Here, we assume that preprocessing for attributes is done, and we can use it as a d_A-dimensional vector f_{i_t} for the item i_t. Meanwhile, a *context* is the contextual information, generally meaning time, place, situation, etc., that can affect the preference independently of users and items. Similar to attributes, we assume that the context is preprocessed as a d_C-dimensional vector c_t. We extend SR into *attribute and context-aware sequential recommendation* (ACSR) as follows: maximize $P(i_{t'+1}|S^{AC}_{u,(t \le t')}, c_{t'+1})$ where $S_u^{AC} = \{(i_t, f_{i_t}, c_t)|1 \le t \le |S_u^{AC}|\}$.

3.2 Background

The general architecture of an ACSR model can be divided into three parts: the item encoding layer, the sequence encoding blocks, and the item scoring layer. We explain each part based on the design of CARCA [33], where it has (1) a strong item encoding layer that utilizes item attributes and contexts, (2) self-attention-based sequence encoding blocks, and (3) a state-of-the-art item scoring layer using a cross-attention.

3.2.1 Item Encoding Layer. The main function of this layer is to extract the *item vector* $v_{i_t} \in \mathbb{R}^d$ given the input (i_t, f_{i_t}, c_t), where d denotes the latent dimension. First, the individual *item embedding* for the item i_t is assigned as IE_{i_t}, from the full-item look-up table $\text{IE} \in \mathbb{R}^{|I| \times d_{\text{IE}}}$, where d_{IE} denotes the dimension of the item embedding. These three vectors $(\text{IE}_{i_t}, f_{i_t}, c_t)$ are then passed to a shallow neural network to obtain the final item vector:

$$z_{i_t} = \sigma^{\text{AC}}(\text{cat}_{\text{col}}(f_{i_t}, c_t) W^{\text{AC}} + b^{\text{AC}}), \tag{1}$$

$$v_{i_t} = \sigma^{\text{item}}(\text{cat}_{\text{col}}(\text{IE}_{i_t}, z_{i_t}) W^{\text{item}} + b^{\text{item}}), \tag{2}$$

where $W^{\text{AC}} \in \mathbb{R}^{(d_A + d_C) \times d_{\text{AC}}}$, $b^{\text{AC}} \in \mathbb{R}^{d_{\text{AC}}}$, $W^{\text{item}} \in \mathbb{R}^{(d_{\text{IE}} + d_{\text{AC}}) \times d}$, $b^{\text{item}} \in \mathbb{R}^d$ are weights and biases for corresponding network, σ^{AC}, σ^{item} are activation functions, and cat_{col} denotes the column-wise vector concatenation.

3.2.2 Sequence Encoding Blocks. With the advent of self-attention blocks, the sequence encoding blocks have been commonly implemented with Transformer [42] architecture that is superior in terms of both performance and computational efficiency to that of the past, namely RNN or CNN-based methods [18, 32]. An *attention* (Attn) is defined as follows: for three given sequences of vectors $Q \in \mathbb{R}^{L_Q \times d_Q}$, $K \in \mathbb{R}^{L_K \times d_K}$, $V \in \mathbb{R}^{L_V \times d_V}$,

$$\text{Attn}(Q, K, V) = \text{softmax}\left((QK^\top)/\sqrt{d_{QK}}\right) V, \tag{3}$$

where $d_Q = d_K = d_{QK}$ and $L_K = L_V = L_{KV}$. Given the number of heads H that is the divisor of d_Q, d_K, d_V, we can create separate linear projection layers so that each head can handle different representations, often referred *multi-head attention* (MHA):

$$\text{MHA}(Q, K, V) = \text{cat}_{\text{col}}\left(\left[\text{Attn}(QW_h^Q, KW_h^K, VW_h^V)\right]_{h=1}^{H}\right), \tag{4}$$

where $W_h^Q \in \mathbb{R}^{d_Q \times d_Q/H}$, $W_h^K \in \mathbb{R}^{d_K \times d_K/H}$, $W_h^V \in \mathbb{R}^{d_V \times d_V/H}$. The *self-attention* assumes that Q, K, and V are the same, and it is used to encode the complex information of the input sequence. In our problem setting, we apply self-attention to item vectors $V_u = [v_{i_1}^\top, v_{i_2}^\top, ..., v_{i_{t'}}^\top]^\top \in \mathbb{R}^{|S_{u,(t \le t')}^{\text{AC}}| \times d}$:

$$\hat{V}_u = \text{MHA}(V_u, V_u, V_u). \tag{5}$$

Following the architecture of Transformer, the results above are passed to a two-layered point-wise feed-forward network (PWFF), where it is defined as follows: for a matrix $X \in \mathbb{R}^{L_X \times d_X}$ denoting a set of vectors,

$$\text{PWFF}(X) = \text{cat}_{\text{row}}\left(\left[\sigma(X_r W^{(1)} + b^{(1)}) W^{(2)} + b^{(2)}\right]_{r=1}^{L_X}\right), \tag{6}$$

where σ is an activation function, $W^{(1)}, W^{(2)} \in \mathbb{R}^{d_X \times d_X}$, $b^{(1)}, b^{(2)} \in \mathbb{R}^{d_X}$ are weights and biases, and cat_{row} is a row-wise vector concatenation. PWFF is an additional layer that helps the model understand more complex relationships and provides further non-linearity. Assuming an additive residual connection, the self-attention block can be stacked into multiple blocks as follows:

$$\hat{V}_u^{(b)} = \text{MHA}(V_u^{(b)}, V_u^{(b)}, V_u^{(b)}), \tag{7}$$

$$V_u^{(b+1)} = \text{PWFF}(\hat{V}_u^{(b)}) + \hat{V}_u^{(b)}, \tag{8}$$

where $V_u^{(1)} = V_u$, and (b) denoting the b-th block, up to the total number of B attention blocks. The final vectors $V_u^{(B)}$ from the sequence encoding blocks are dubbed *latent sequence vectors*. Unlike many other attention-based models, since the temporal information is often explicitly given in the form of a context [33, 37], we do not employ positional embeddings.

3.2.3 Item Scoring Layer. After extracting the latent sequence vectors, the *item scoring layer* calculates preference scores to rank candidate items, producing recommendation output. There are two major methodologies for the layer.

Inner Product (IP). Similar to MF, this method uses the inner product (IP) value between a user vector and an item vector to obtain a preference score. Since the IP method is commonly implemented via negative item sampling, it is widely used due to its great advantage in computational efficiency, especially in terms of memory and parameters [18, 25]. The sampled candidate items are encoded into *candidate item vectors* $C_u = [v_{i_{t'+1}}^\top, v_{i_1^-}^\top, v_{i_2^-}^\top, ..., v_{i_{|N_u|}^-}^\top]^\top \in \mathbb{R}^{(|N_u|+1) \times d}$ via the same item encoding layer used in the user-item sequence encoding, with shared parameters. The training method of IP can be categorized into two losses [6]:

(1) Binary cross-entropy loss (BCE): This method trains positive and negative items independently by treating the IP value as the logit of binary classification.
(2) Normalized temperature-scaled cross-entropy loss (NT-Xent) [6, 39]: After the L_2-normalization to make IP as cosine similarity, a temperature-scaled softmax is applied to perform $(N+1)$-way classification. This method trains the positive score to be relatively higher than the negative score.

Cross-Attention (CA). Unlike the IP, the sequence latent vector is not treated as the user vector at step t in CA [33]. The method uses candidate item vectors C_u as query and sequence latent vectors $V_u^{(B)}$ as both key and value for another multi-head attention layer. After applying a multiplicative residual connection, the results are passed to a scoring layer ϕ^{score} that produces the preference scores:

$$\hat{C}_u = C_u \odot \text{MHA}(C_u, V_u^{(B)}, V_u^{(B)}), \tag{9}$$

$$Y = \phi^{\text{score}}(\hat{C}_u) = \sigma^{\text{score}}(\hat{C}_u W^{\text{score}} + b^{\text{score}}), \tag{10}$$

where $W^{\text{score}} \in \mathbb{R}^d$ is a scoring weight vector, $b^{\text{score}} \in \mathbb{R}$ is a scoring bias, and σ^{score} is the activation function. The score is treated as the logit of binary classification, similar to BCE from the IP method. When CA decides whether or not to recommend each candidate item, it explicitly considers the entire item sequence via the cross-attention mechanism, leading to superior performance in most cases.

3.3 Proxy-based Item Representation

Our proposed model, the **P**roxy-based **I**tem **R**epresentation (PIR) method, replaces the item embeddings IE_{i_t} in the item encoding layer. First, we introduce the learnable *proxy embeddings*, namely $P = [p_1^\top, p_2^\top, ..., p_{n_{\text{proxy}}}^\top]^\top \in \mathbb{R}^{n_{\text{proxy}} \times d_{\text{proxy}}}$. Here, the number of proxy embeddings n_{proxy} and its dimension d_{proxy} are hyper-parameters. We calculate the appropriate weights for each proxy embedding by item attribute f_{i_t} and context c_t to produce a *proxy-based item representation* $\mathrm{PIR}_{i_t} \in \mathbb{R}^{d_\text{proxy}}$ that is a weighted sum of proxy embeddings. Specifically, we use a 2-layered neural network φ to compute the unnormalized weight for each proxy embedding:

$$w'_{i_t} = \sigma^{\varphi,(1)}(\mathrm{cat}_\text{col}(f_{i_t}, c_t)W^{\varphi,(1)} + b^{\varphi,(1)}), \tag{11}$$

$$\begin{aligned} w_{i_t} &= \sigma^{\varphi,(2)}(w'_{i_t}W^{\varphi,(2)} + b^{\varphi,(2)}) \\ &= \varphi(f_{i_t}, c_t) \in \mathbb{R}^{n_\text{proxy}}, \end{aligned} \tag{12}$$

where $W^{\varphi,(1)} \in \mathbb{R}^{(d_\text{A}+d_\text{C})\times d_\varphi}$, $b^{\varphi,(1)} \in \mathbb{R}^{d_\varphi}$ are the weight and bias for the first layer, and $W^{\varphi,(2)} \in \mathbb{R}^{d_\varphi \times n_\text{proxy}}$, $b^{\varphi,(2)} \in \mathbb{R}^{n_\text{proxy}}$ are the weight and bias for the second layer. $\sigma^{\varphi,(1)}$ and $\sigma^{\varphi,(2)}$ can be any appropriate activation functions, but we use LeakyReLU for $\sigma^{\varphi,(1)}$ and the identity function for $\sigma^{\varphi,(2)}$, which were empirically chosen through hyper-parameter tuning. After normalizing through a softmax layer, PIR_{i_t} is computed as the weighted sum of proxy embeddings:

$$\hat{w}_{i_t} = \mathrm{softmax}(w_{i_t}), \tag{13}$$

$$\mathrm{PIR}_{i_t} = \hat{w}_{i_t}P. \tag{14}$$

The final item vector v_{i_t} is then calculated by appending the item attribute and context information to the newly created representation. Slightly different from the process described in CARCA, we introduce an additional 1-layer neural network for attribute vectors to provide an additional non-linearity to the attribute information.

$$f'_{i_t} = \phi^\text{A}(f_{i_t}) = \sigma^\text{A}(f_{i_t}W^\text{A} + b^\text{A}), \tag{15}$$

$$z_{i_t} = \phi^\text{AC}(f'_{i_t}, c_t) = \sigma^\text{AC}(\mathrm{cat}_\text{col}(f'_{i_t}, c_t)W^\text{AC} + b^\text{AC}), \tag{16}$$

$$v_{i_t} = \phi^\text{item}(i_t, z_{i_t}) = \sigma^\text{item}(\mathrm{cat}_\text{col}(\mathrm{PIR}_{i_t}, z_{i_t})W^\text{item} + b^\text{item}), \tag{17}$$

where $W^\text{A} \in \mathbb{R}^{d_\text{A}\times d'_\text{A}}$, $b^\text{A} \in \mathbb{R}^{d'_\text{A}}$, $W^\text{AC} \in \mathbb{R}^{(d'_\text{A}+d_\text{C})\times d_\text{AC}}$, $b^\text{AC} \in \mathbb{R}^{d_\text{AC}}$, $W^\text{item} \in \mathbb{R}^{(d_\text{proxy}+d_\text{AC})\times d}$, $b^\text{item} \in \mathbb{R}^d$ are the weights and biases for the corresponding networks and σ^A, σ^AC, σ^item are the activation functions. The overall architecture of our model is illustrated in Figure 2.

The above structure itself represents items solely based on their content information and behaves similarly to content-based filtering, which cannot reflect the collaborative signals between the items on its own. To give direct collaborative signals to frequent items, we introduce a *frequent item bias*, a concept similar to the known user bias from [10]. The frequent item bias is a structure where the selected top K frequent items can memorize the biases for the proxy weights:

$$\hat{w}_{i_t} = \mathrm{softmax}(w_{i_t} + b^\text{freq}_{i_t}), \tag{18}$$

where $b^\text{freq}_{i_t} \in \mathbb{R}^{n_\text{proxy}}$ is a learnable bias vector. Existing models often attempt to reduce parameter usage by allocating different dimensions [27, 53] or model structures [44, 55] for frequent and infrequent items. In our proposed model, the only difference between the frequent and infrequent items is whether it can partly memorize the proxy weights or not in an identical model structure. Note that if $K = 0$, the proxy-based model can be interpreted as a deep learning version of content-based filtering, since it only uses the content information. On the other hand, if $K = |I|$, the model behaves similarly to low-rank factorization of the full-item look-up table IE, with a latent dimension of n_proxy, since the computation is similar to matrix factorization but with softmax non-linearity. Note

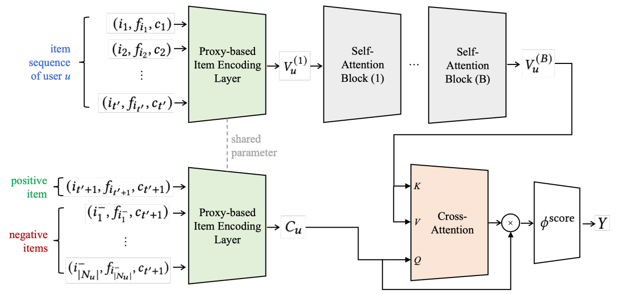

Figure 2: Overall model architecture.

that in this case, the model can still be applied to the case where there is no content information available, which reduces exactly into the low-rank factorization with softmax non-linearity.

Our choice of using the softmax function generates PIR_{i_t} to be a vector that resides inside a simplex, where vertices are proxy embeddings: $\hat{w}_{i_t}P = \sum_{r=1}^{n_\text{proxy}} \hat{w}_{i_t,r}P_r$ where $\sum_{r=1}^{n_\text{proxy}} \hat{w}_{i_t,r} = 1$, meeting the condition for simplex. Here, we can expect that the abundant interaction of frequent items will train the proxy embeddings to their promising state. Therefore, any PIR_{i_t} can be expected to be a comprehensible vector to the network, where otherwise the infrequent item representations would have been learned poorly due to inadequate training. In addition, another property of the weighted sum mechanism is that the gradient of parameter update is formed as a unit of proxy embeddings: assuming that the proxy embeddings are fixed, $\frac{\partial \mathrm{PIR}_{i_t}}{\partial\theta} = \sum_{r=1}^{n_\text{proxy}} \frac{\partial \hat{w}_{i_t,r}}{\partial\theta}P_r$, for a parameter θ to be updated. This property allows infrequent items to borrow training signals from frequent items via proxy embeddings. We also prove the following propositions in the appendix: *content locality*, where infrequent items with similar attributes are close to each other in the proxy-based representation space, and *bias priority*, where the frequent item bias can be prioritized over the content information.

3.4 Complexity of Parameters

Since IE uses $|I|$ embeddings while P only uses n_proxy embeddings, the proxy-based item representation can reduce a significant number of parameters. For computational convenience, assume that all latent dimensions are equal to, or proportional to d: $d_\text{IE} = d$, $d_\text{AC} \propto d$, $d'_\text{A} = d$, $d_\varphi \propto d$, and $d_\text{proxy} = d$. In a baseline model that uses IE, the number of parameters is $O(|I|d + M(d))$, where $M(d)$ is the number of parameters of the network other than the item embedding. For our model, the number of parameters is $O((K+d)n_\text{proxy} + M'(d))$, where $M'(d)$ denotes the number of parameters of the network other than the PIR layer. Note that in this case, the number of parameters does not increase proportionally to the number of items $|I|$. Even in the extreme case where $K = |I|$, parameters can be reduced whenever n_proxy is just slightly smaller than d: let $n_\text{proxy} = \alpha d$, then we get $|I|d + M(d) > (|I|+d)n_\text{proxy} + M'(d) \Leftrightarrow (|I|+d)^{-1}(|I| + (M(d) - M'(d))/d) > \alpha$. Assuming that $M(d) \approx M'(d)$, the model can shrink whenever α meets the condition (note that $d \ll |I|$ holds in general).

Table 2: Dataset statistics after preprocessing.

Dataset	#user	#item	#interaction	Density	Unique Attributes	Duplicates	Memorization
Fashion	45,184	166,270	358,003	0.0048%	163,985	1.0	98.09%
Men	34,244	110,636	254,870	0.0067%	109,282	1.0	98.36%
Beauty	52,204	57,289	394,908	0.0132%	11,094	5.2	41.92%
Game	31,013	23,715	287,107	0.0390%	1,701	13.9	7.26%
ML-20M	138,287	20,720	9,995,410	0.3488%	1,342	20.3	4.92%

3.5 Training Objective

In sequential recommendation, it is common to create $|S_u^{\mathrm{AC}}| - 1$ tasks of Next Item Prediction (NIP) by using the aforementioned latent sequence vectors as user vectors, namely $V_{u,t}^{(B)}$ for each time step $1 \leq t \leq |S_u^{\mathrm{AC}}| - 1$. However, in the case of cross-attention, the latent sequence vectors possess bidirectional information of the item sequence. Therefore, training cross-attention with NIP task incurs a discrepancy between the training phase and the inference phase since NIP assumes a causality constraint in the training phase. To this end, we adopt Last Item Prediction (LIP) task, which uses only the very last item of each user as a positive item. In every training epoch, each item sequence S_u^{AC} is randomly cut [46] into a subsequence so that diverse input data can be trained for the same user. We also apply NT-Xent [35, 39] on the cross-attention, so that the model can employ multiple negatives. We call this method Contrastive CA, where it can only be implemented in LIP task since it is not feasible to sample multiple negative items having the same context in NIP task due to the enormous memory cost. The final training objective is as follows: Assuming that for every epoch, each item sequence S_u^{AC} of user u is randomly cut, and N_u is randomly sampled except for testing,

$$\mathcal{L} = -|U|^{-1} \textstyle\sum_{u \in U} \log \left(e^{Y_1} / \sum_{j=1}^{|I_u^-|+1} e^{Y_j} \right). \tag{19}$$

4 EXPERIMENT

To demonstrate the effectiveness of our proposed model, we have constructed experiments to answer the following research questions:

- **RQ1** Does PIR layer show superior performance in a plug-and-play manner?
- **RQ2** Does PIR layer enhance representation quality of infrequent items?
- **RQ3** Does ProxyRCA have a parameter-efficient structure?
- **RQ4** Do the proxy weights and proxy embeddings together compositionally compute item representations?

4.1 Experimental Settings

4.1.1 Datasets. To evaluate and analyze the proposed model, we use five widely used recommendation datasets from different domains, namely Fashion, Men, Game, and Beauty from Amazon Review datasets [24], and MovieLens-20M (ML-20M). Fashion and Men use the image vectors extracted from the pre-trained ResNet50 model as their item attributes. Except for the price feature of Game, the attributes for Beauty and Game are discrete and categorical information, namely tags. In ML-20M, only the genre of each movie is used as item attributes, which represents the case where the content information are extremely limited. For contexts, timestamp data is decomposed into multi-dimensional date information, following the preprocessing steps in [33]. The overall statistics of the datasets are given in Table 2. In the preprocessing step, users with three or fewer interactions are excluded from the dataset to make a proper train-valid-test split.

4.1.2 Evaluation Protocol. Overall, we follow the conventional evaluation protocol from the sequential recommendation studies [15, 18, 33]. We use the leave-one-out evaluation where the last item of each user is the test item, and the previous one to the last item is the valid item. A total of 100 negative items are randomly sampled among items that have not been interacted with the user. The performance is measured with Hit-Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG), averaged over all users. We run 5 times each with different random seeds and report the average performance.

4.1.3 Comparison Models. For the fair comparison, we implemented the baseline models to have a similar structure and the same training objective (LIP task) to our setting. Full explanation and experimental results on NIP task baselines, namely BERT4Rec [40], SASRec [18], SSE-PT [46], and S^{3}Rec [54] are in the appendix.

(1) Popular: A non-personalized recommendation where the preference score is based on the item's global popularity. It serves as a sanity check for the performance lower bound.
(2) SASRec++ [33]: An extension of SASRec [18] that utilizes item attributes and context.
(3) CARCA [33]: A state-of-the-art ACSR model that uses cross-attention as an item scoring layer.
(4) BPR++ (ours): An extension of the non-sequential model BPR [35] that utilizes item attributes and context.
(5) MixDim++ (ours): An extension of CARCA with parameter-efficient mixed-dimension embedding [27] for encoding infrequent items. Note that this is an important baseline, which represents models that handle infrequent items (either long-tail or cold-start).
(6) ProxyRCA (ours): This is our main proposed model that employs proxy-based item representation as the item encoding layer on top of CARCA, named after **Proxy**-based item representation **R**ecommendation model with **C**ross-**A**ttention.

4.1.4 Implementation Details. To match the similar scale of parameters, we fixed the latent dimension d to 256, and n_{proxy} was set to 128. We initialized $b_{i_t}^{\mathrm{freq}}$ to be a zero vector in order to prevent the unintended influence on the computation of proxy weights

Table 3: Performance comparison on all datasets. The best and the second best results are marked as bold and italic numbers respectively. The asterisk(*) denotes statistically significant ($p < 0.05$) gain against the non-PIR counterpart, using the t-test.

Task	Model	Fashion		Men		Beauty		Game		ML-20M	
		R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10
	Popular	0.407	0.262	0.415	0.269	0.451	0.261	0.519	0.314	0.815	0.530
BPR	BPR++	0.523	0.332	0.429	0.266	0.505	0.352	0.768	0.564	0.956	0.702
	↪ with **PIR**	0.620*	0.406*	0.554*	0.358*	0.511	0.353	0.770	*0.582**	0.957	0.761*
LIP	MixDim++	0.623	0.407	0.570	0.365	0.587	0.398	0.766	0.556	*0.961*	0.781
	SASRec++	0.630	0.416	0.587	0.379	0.601	0.415	0.753	0.539	0.949	0.761
	↪ with **PIR**	0.635	0.426*	0.580	0.381	0.599	0.422	*0.779**	0.572*	0.944	0.773*
	CARCA	*0.648*	*0.427*	*0.614*	*0.398*	*0.608*	*0.423*	0.762	0.560	0.961	*0.788*
	↪ with **PIR (ProxyRCA)**	**0.661***	**0.446***	**0.617**	**0.408**	**0.626***	**0.449***	**0.809***	**0.611***	**0.962**	**0.792***

in the early stages of the training. Detailed hyper-parameter tuning procedures are described in the appendix. The PyTorch-based implementation is shared as an open-source repository [1].

4.2 Overall Performance Comparison (RQ1)

Table 3 shows the overall performance comparison. We replaced the item encoding layer of three representative baselines, namely BPR++, SASRec++ and CARCA, and achieved performance improvement on almost all cases, as well as, all datasets in terms of NDCG. The final model, ProxyRCA reaches the state-of-the-art performance, improving up to 17% when compared to the previously reported results. The improvement on BPR model demonstrates that our PIR layer is not bound to sequential models. Interesting point is that Fashion and Men were previously known to be sensitive to sequential information, but with PIR, BPR++ reaches comparable performance to sequential models. Note that in any case, since we are using $n_{\text{proxy}} = 0.5d$, only a minimum of 10% and a maximum of 50% of training parameters are used when compared to non-PIR counterparts. Moreover, our choice of the training objective, the LIP task, is generally superior to NIP task if tuned properly, which is shown in the appendix.

To further analyze the above results, we counted the unique number of attributes and the number of duplicate items for each attribute, and implemented a 2-layered neural network that takes item attributes as input to perform a memorization task of classifying items based on their attributes. The result is summarized in Table 2. In Fashion and Men, the item attributes distinguish the item accurately. In Game and ML-20M however, more than ten items share exactly the same attributes on average, and thus it is impractical to distinguish items by their attributes alone. Even with the datasets where attribute information is extremely weak, our ProxyRCA model can still outperform the baselines. We claim that the performance gain is due to the content-oriented shared embedding effect, similar to SSE [47], explaining why the model outperforms even with limited attribute information.

4.3 Performance on Infrequent Items (RQ2)

As mentioned in Table 1, the baseline model showed a performance increase when infrequent item embeddings are replaced into the shared unknown token in sacrifice of the recommendation diversity. This phenomenon is not observed in models with PIR, which indirectly proves that the representation quality of the infrequent items is improved.

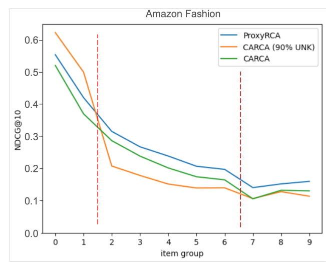

Figure 3: Performance comparison over frequency group.

For further investigation, we divide items into 10 groups, sorted by frequency, so that the sum of item occurrence in each group is equal (*i.e.*, the frequent group will have much fewer items compared to the infrequent group). We measure the mean performance of each group, average over each item on the test set, since an item can be a test item multiple times. The result is shown in Figure 3. For item groups 0 to 1, the baseline model with 90% of removal surmounts others but gives up the recommendation for majority of relatively infrequent item groups, which is an intuitive result. On the other hand, looking at item groups 7 to 9, CARCA does not show a big difference compared to the case of removing infrequent items, but ProxyRCA shows performance improvement in these groups as well, which also shows that the representation quality is improved.

4.4 Parameter Efficiency (RQ3)

To demonstrate the parameter efficiency of ProxyRCA, we created four new datasets, namely Fashion T_r ($r = 1, 2, 3, 4$) by truncating Fashion so that the number of items for each partition is equal to $r/4 \times |I|$. These datasets simulate the real-world recommendation scenario where new items are added to the system. As shown in Figure 4, the baseline model continuously expands the look-up table to match the increasing number of items as r increases. However,

[1] https://github.com/theeluwin/ProxyRCA

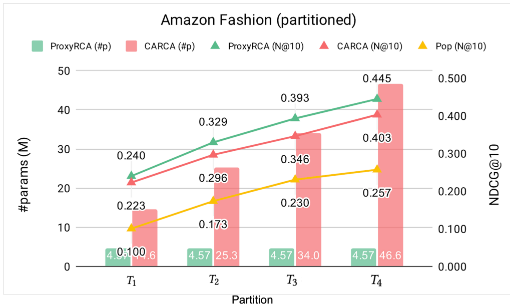

Figure 4: Parameter and performance comparison on the partitioned Fashion, simulating the growth of data.

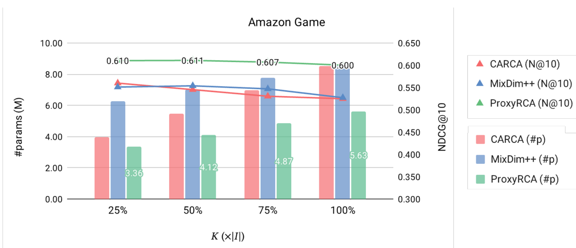

Figure 5: Parameter and performance comparison on Game over different K value.

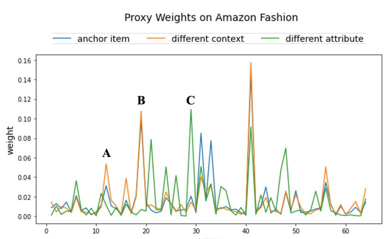

Figure 6: Visualization of proxy weights that reflect attribute and context modification.

ProxyRCA succeeds in maintaining a superior performance with a fixed number of parameters.

To analyze the effect of the number of frequent items, we compare CARCA, MixDim++, and ProxyRCA in Game. For a fair comparison, CARCA treats items beyond top K frequent items as unknown items, assigning the shared unknown embedding. For MixDim++, K corresponds to the ratio of items using full latent dimension, while smaller dimension is set to n_{proxy}. The experiment result is in Figure 5. We can see that even with similar settings, ProxyRCA uses less parameters and slow parameter growth rate, with superior recommendation performance.

4.5 Proxy Analysis (RQ4)

Since the proxy weights are mainly computed using item attribute and context, the weight values should be different when the attribute and context are changed without the frequent item bias. In Fashion, we first randomly choose an anchor item and visualize the computed proxy weights, as in Figure 6. To see the impact of context in proxy weights, we change context to an another value

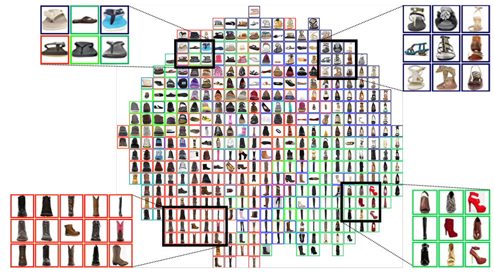

Figure 7: PCA visualization of item vectors on Fashion, using shoes category. The border colors denote the heaviest-weighted proxy.

and visualize the weights. Similarly, to see the impact of attributes in proxy weights, we sample another item with different attributes but same context, and visualize the weights. As shown in the figure, due to the nature of Fashion that uses item images as attributes, the attributes are strongly reflected in the proxy weights. Therefore, we can see that most weights are similar when only contexts differ (points **B** and **C** in the figure). Even so, different context still assign different proxy weights (point **A** as demonstrated in the figure). This visualization indicates that each proxy has its role to represent items compositionally.

To show that the proxy-based item representation can also provide a clustering effect on infrequent items, we visualize the item vectors with $K = 0$ using PCA, as in Figure 7. We use items of the shoes category among others (*e.g.*, dress, outer) in Fashion to see if the model successfully captures the subcategory information without any labels, which is a rather challenging task when compared to supercategory clustering. The figure shows that the items of similar subcategories (e.g., flip-flops on top-left, sandals on top-right, boots on bottom-left, and pumps on bottom-right in the figure) are naturally clustered, even without any explicit labels.

5 CONCLUSION

In this paper, we highlighted the issues with the full-item embedding table from the perspective of infrequent item training through experimental demonstration. To overcome the issues, we proposed a proxy-based item representation model that can replace the existing item encoding layer in a plug-and-play fashion, by computing the item representation as a weighted sum of learnable proxy embeddings. To prove the effectiveness of our model, experiments were conducted using recommendation benchmark datasets and achieved state-of-the-art performance.

ACKNOWLEDGMENTS

This work was made in collaboration with Seoul National University and IntelliSys Co., Ltd. Also, this work was partly supported by the National Research Foundation of Korea (NRF) [No. RS-2023-00243243] and Institute of Information & communications Technology Planning & Evaluation (IITP) [NO.2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)] [NO.2021-0-02068, Artificial Intelligence Innovation Hub (Artificial Intelligence Institute, Seoul National University)], both grant funded by the Korea government (MSIT).

REFERENCES

[1] Newsha Ardalani, Carole-Jean Wu, Zeliang Chen, Bhargav Bhushanam, and Adnan Aziz. 2022. Understanding Scaling Laws for Recommendation Models. *arXiv preprint arXiv:2208.08489* (2022).

[2] Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. 2019. A Review on Deep Learning for Recommender Systems: Challenges and Remedies. *Artificial Intelligence Review* 52 (2019), 1–37.

[3] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. 2020. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. *Advances in Neural Information Processing Systems* 33 (2020), 9912–9924.

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging Properties in Self-supervised Vision Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9650–9660.

[5] Hao Chen, Zefan Wang, Feiran Huang, Xiao Huang, Yue Xu, Yishi Lin, Peng He, and Zhoujun Li. 2022. Generative Adversarial Framework for Cold-start Item Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2565–2571.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *International Conference on Machine Learning*. 1597–1607.

[7] Yizhou Chen, Guangda Huzhang, Anxiang Zeng, Qingtao Yu, Hui Sun, Hengyi Li, Jingyi Li, Yabo Ni, Han Yu, and Zhiming Zhou. 2023. Clustered Embedding Learning for Recommender Systems. In *The World Wide Web Conference*.

[8] Yongjun Chen, Zhiwei Liu, Jia Li, Julian McAuley, and Caiming Xiong. 2022. Intent Contrastive Learning for Sequential Recommendation. *arXiv preprint arXiv:2202.02519* (2022).

[9] Hyunsoo Cho, Jinseok Seol, and Sang-goo Lee. 2021. Masked Contrastive Learning for Anomaly Detection. *The 30th International Joint Conference on Artificial Intelligence* (2021).

[10] Junsu Cho, SeongKu Kang, Dongmin Hyun, and Hwanjo Yu. 2021. Unsupervised Proxy Selection for Session-based Recommender Systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 327–336.

[11] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for Youtube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 191–198.

[12] Manqing Dong, Feng Yuan, Lina Yao, Xiwei Xu, and Liming Zhu. 2020. Mamo: Memory-augmented Meta-optimization for Cold-start Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 688–697.

[13] Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S Yu. 2021. Continuous-time Sequential Recommendation with Temporal Graph Collaborative Transformer. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 433–442.

[14] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-machine based Neural Network for CTR Prediction. *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (2017).

[15] Ruining He and Julian McAuley. 2016. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the AAAI conference on Artificial Intelligence*, Vol. 30.

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.

[17] Wang-Cheng Kang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Ting Chen, Lichan Hong, and Ed H Chi. 2021. Learning to Embed Categorical Features without Embedding Tables for Recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 840–850.

[18] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining*. 197–206.

[19] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. 2020. Proxy Anchor Koss for Deep Metric Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3238–3247.

[20] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[21] Junnan Li, Pan Zhou, Caiming Xiong, and Steven CH Hoi. 2021. Prototypical Contrastive Learning of Unsupervised Representations. *International Conference on Learning Representations* (2021).

[22] Yang Li, Tong Chen, Peng-Fei Zhang, and Hongzhi Yin. 2021. Lightweight Self-attentive Sequential Recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 967–977.

[23] Siyi Liu, Chen Gao, Yihong Chen, Depeng Jin, and Yong Li. 2021. Learnable embedding sizes for recommender systems. *2021 International Conference on Learning Representations* (2021).

[24] 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*. 43–52.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems* 26 (2013).

[26] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. 2017. No Fuss Distance Metric Learning using Proxies. In *Proceedings of the IEEE International Conference on Computer Vision*. 360–368.

[27] Maxim Naumov, Dheevatsa Mudigere, Jiyan Yang, and James Zou. 2021. Mixed Dimension Embeddings with Application to Memory-efficient Recommendation Systems. In *IEEE International Symposium on Information Theory*. 2786–2791.

[28] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-n Recommender Systems. In *IEEE 11th International Conference on Data Mining*. 497–506.

[29] Aleksandr Petrov and Craig Macdonald. 2022. Effective and Efficient Training for Sequential Recommendation using Recency Sampling. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 81–91.

[30] Aleksandr Petrov and Craig Macdonald. 2022. A Systematic Review and Replicability Study of BERT4Rec for Sequential Recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 436–447.

[31] Andreas Pfadler, Huan Zhao, Jizhe Wang, Lifeng Wang, Pipei Huang, and Dik Lun Lee. 2020. Billion-scale recommendation with Heterogeneous Side Information at Taobao. In *IEEE 36th International Conference on Data Engineering*. 1667–1676.

[32] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *Proceedings of the 11th ACM Conference on Recommender Systems*. 130–137.

[33] Ahmed Rashed, Shereen Elsayed, and Lars Schmidt-Thieme. 2022. Context and Attribute-Aware Sequential Recommendation via Cross-Attention. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 71–80.

[34] Steffen Rendle. 2010. Factorization Machines. In *IEEE International Conference on Data Mining*. 995–1000.

[35] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. 452–461.

[36] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web*. 285–295.

[37] Jinseok Seol, Youngrok Ko, and Sang-goo Lee. 2022. Exploiting Session Information in BERT-based Session-aware Sequential Recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2639–2644.

[38] Jiayi Shen, Haotao Wang, Shupeng Gui, Jianchao Tan, Zhangyang Wang, and Ji Liu. 2021. UMEC: Unified Model and Embedding Compression for Efficient Recommendation Systems. *International Conference on Learning Representations*.

[39] Kihyuk Sohn. 2016. Improved Deep Metric Learning with Multi-class n-pair Loss Objective. *Advances in Neural Information Processing Systems* 29 (2016).

[40] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1441–1450.

[41] Viet-Anh Tran, Romain Hennequin, Jimena Royo-Letelier, and Manuel Moussallam. 2019. Improving Collaborative Metric Learning with Efficient Negative Sampling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1201–1204.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. *Advances in Neural Information Processing Systems* 30 (2017).

[43] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. 2019. Sequential Recommender Systems: Challenges, Progress and Prospects. *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (2019).

[44] Yinwei Wei, Xiang Wang, Qi Li, Liqiang Nie, Yan Li, Xuanping Li, and Tat-Seng Chua. 2021. Contrastive Learning for Cold-start Recommendation. In *Proceedings of the 29th ACM International Conference on Multimedia*. 5382–5390.

[45] Mikołaj Wieczorek, Barbara Rychalska, and Jacek Dąbrowski. 2021. On the Unreasonable Effectiveness of Centroids in Image Retrieval. In *Neural Information Processing: 28th International Conference, ICONIP 2021, Proceedings, Part IV 28*. Springer, 212–223.

[46] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential Recommendation via Personalized Transformer. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 328–337.

[47] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James L Sharpnack. 2019. Stochastic Shared Embeddings: Data-driven Regularization of Embedding Layers. *Advances in Neural Information Processing Systems* 32 (2019).

[48] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems.. In *IJCAI*, Vol. 17. Melbourne, Australia, 3203–3209.

[49] Bencheng Yan, Pengjie Wang, Kai Zhang, Wei Lin, Kuang-Chih Lee, Jian Xu, and Bo Zheng. 2021. Learning Effective and Efficient Embedding via an Adaptively-masked Twins-based Layer. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3568–3572.

[50] Xufeng Yao, Yang Bai, Xinyun Zhang, Yuechen Zhang, Qi Sun, Ran Chen, Ruiyu Li, and Bei Yu. 2022. PCL: Proxy-based Contrastive Learning for Domain Generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7097–7107.

[51] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. 2012. Challenging the Long Tail Recommendation. *Proceedings of the VLDB Endowment 5.9* (2012).

[52] Dejiao Zhang, Feng Nan, Xiaokai Wei, Shangwen Li, Henghui Zhu, Kathleen McKeown, Ramesh Nallapati, Andrew Arnold, and Bing Xiang. 2021. Supporting Clustering with Contrastive Learning. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2021).

[53] Xiangyu Zhao, Haochen Liu, Hui Liu, Jiliang Tang, Weiwei Guo, Jun Shi, Sida Wang, Huiji Gao, and Bo Long. 2020. Memory-efficient Embedding for Recommendations. *arXiv preprint arXiv:2006.14827* (2020).

[54] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-Rec: Self-supervised Learning for Sequential Recommendation with Mutual Information Maximization. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1893–1902.

[55] Yongchun Zhu, Ruobing Xie, Fuzhen Zhuang, Kaikai Ge, Ying Sun, Xu Zhang, Leyu Lin, and Juan Cao. 2021. Learning to Warm Up Cold Item Embeddings for Cold-start Recommendation with Meta Scaling and Shifting Networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1167–1176.