Learning Vector-Quantized Item Representation for Transferable Sequential Recommenders

Yupeng Hou
houyupeng@ruc.edu.cn
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China

Zhankui He
zhh004@eng.ucsd.edu
UC San Diego
San Diego, California, USA

Julian McAuley
jmcauley@eng.ucsd.edu
UC San Diego
San Diego, California, USA

Wayne Xin Zhao[†✉]
batmanfly@gmail.com
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China

ABSTRACT

Recently, the generality of natural language text has been leveraged to develop transferable recommender systems. The basic idea is to employ pre-trained language models (PLM) to encode item text into item representations. Despite the promising transferability, the binding between item text and item representations might be *too tight*, leading to potential problems such as over-emphasizing the effect of text features and exaggerating the negative impact of domain gap. To address this issue, this paper proposes **VQ-Rec**, a novel approach to learning Vector-Quantized item representations for transferable sequential Recommenders. The main novelty of our approach lies in the new item representation scheme: it first maps item text into a vector of discrete indices (called *item code*), and then employs these indices to lookup the code embedding table for deriving item representations. Such a scheme can be denoted as "*text* $\Longrightarrow$ *code* $\Longrightarrow$ *representation*". Based on this representation scheme, we further propose an enhanced contrastive pre-training approach, using semi-synthetic and mixed-domain code representations as hard negatives. Furthermore, we design a new cross-domain fine-tuning method based on a differentiable permutation-based network. Extensive experiments conducted on six public benchmarks demonstrate the effectiveness of the proposed approach, in both cross-domain and cross-platform settings. Code and pre-trained model are available at: https://github.com/RUCAIBox/VQ-Rec.

CCS CONCEPTS

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

† Beijing Key Laboratory of Big Data Management and Analysis Methods.
✉ Corresponding author.

WWW '23, May 1–5, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583434

ACM Reference Format:
Yupeng Hou, Zhankui He, Julian McAuley, Wayne Xin Zhao. 2023. Learning Vector-Quantized Item Representation for Transferable Sequential Recommenders. In *Proceedings of the ACM Web Conference 2023 (WWW '23), May 1–5, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543507.3583434

1 INTRODUCTION

Sequential recommender systems have been widely deployed on various application platforms for recommending items of interest to users. Typically, such a recommendation task is formulated as a sequence prediction problem [16, 24, 39, 44], inferring the next item(s) that a user is likely to interact with based on her/his historical interaction sequences. Although a similar task formulation has been taken for different sequential recommenders, it is difficult to reuse an existing well-trained recommender for new recommendation scenarios [18, 29]. For example, when a new domain emerges with specific interaction characteristics, one may need to train a recommender from scratch, which is time-consuming and can suffer from cold-start issues. Thus, it is desirable to develop *transferable sequential recommenders* [10, 18, 50], which can quickly adapt to new domains or scenarios.

For this purpose, in recommender systems literature, early studies mainly conduct cross-domain recommendation methods [29, 73, 74] by transferring the learned knowledge from existing domains to a new one. These studies mainly assume that shared information (*e.g.,* overlapping users/items [19, 43, 73] or common features [46]) are available for learning cross-domain mapping relations. However, in real applications, users or items are only partially shared or completely non-overlapping across different domains (especially in a cross-platform setting), making it difficult to effectively conduct cross-domain transfer. Besides, previous content-based transfer methods [12, 46] usually design specific approaches tailored for the data format of shared features, which is not generally useful in various recommendation scenarios.

As a recent approach, several studies [10, 18, 50] propose to leverage the generality of natural language texts (*i.e.,* title and description text of items, called *item text*) for bridging the domain gap in recommender systems. The basic idea is to employ the learned text encodings via pre-trained language models (PLM) [2, 44] as

universal item representations. Based on such item representations, sequential recommenders pre-trained on the interaction data from a mixture of multiple domains [10, 18, 50] have shown promising transferability. Such a paradigm can be denoted as "*text* $\Longrightarrow$ *representation*". Despite the effectiveness, we argue that the binding between item text and item representations is "*too tight*" in previous approaches [10, 18], thus leading to two potential issues. First, since these methods employ text encodings to derive item representations (without using item IDs), text semantics have a direct influence on the recommendation model. Thus, the recommender might overemphasize the effect of text features (*e.g.,* generating very similar recommendations in texts) instead of sequential characteristics reflected in interaction data. Secondly, text encodings from different domains (with varied distributions and semantics [11, 18]) are not naturally aligned in a unified semantic space, and the domain gap existing in text encodings is likely to cause a performance drop during multi-domain pre-training. The tight binding between text encodings and item representations might exaggerate the negative impact of the domain gap.

Considering these issues, our solution is to incorporate intermediate discrete item indices (called *codes* in this work) in item representation scheme and relax the strong binding between item text and item representations, which can be denoted as "*text* $\Longrightarrow$ *code* $\Longrightarrow$ *representation*". Instead of directly mapping text encodings into item representations, we consider a two-step item representation scheme. Given an item, it first maps the item text to a vector of discrete indices (*i.e.,* item code), and then aggregates the corresponding embeddings according to the item code as the item representation. The merits of such a representation scheme are twofold. Firstly, item text is mainly utilized to generate discrete codes, which can reduce its influence on the recommendation model meanwhile inject useful text semantics. Second, the two mapping steps can be learned or tuned according to downstream domains or tasks, making it more flexible to fit new recommendation scenarios. To develop our approach, we highlight two key challenges to address: (i) how to learn discrete item codes that are sufficiently distinguishable for accurate recommendation; (ii) how to effectively pre-train and adapt the item representations considering the varied distribution and semantics across different domains.

To this end, we propose **VQ-Rec**, a novel approach to learn Vector-Quantized item representations for transferable sequential Recommenders. Different from existing transferable recommenders based on PLM encoding, VQ-Rec maps each item into a discrete D-dimensional code as the indices for embedding lookup. To obtain semantically-rich and distinguishable item codes, we utilize optimized product quantization (OPQ) techniques to discretize text encodings of items. In this way, the discrete codes that preserve the textual semantics are distributed over the item set in a more uniform way, so as to be highly distinguishable. Since our representation scheme does not modify the underlying backbone (*i.e.,* Transformer), it is generally applicable to various sequential architectures. To capture transferable patterns based on item codes, we pre-train the recommender on a mixture of multiple domains in a contrastive learning approach. Both mixed-domain and semi-synthetic code representations are used as hard negatives to enhance the contrastive training. To transfer the pre-trained model to a downstream domain, we propose a differentiable permutation-based network to learn the code-embedding alignment, and further update the code embedding table to fit the new domain. Such fine-tuning is highly parameter-efficient, as only the parameters involved in item representations need to be tuned.

Empirically, we conduct extensive experiments on six benchmarks, including both cross-domain and cross-platform scenarios. Experimental results demonstrate the strong transferability of our approach. Especially, inductive recommenders purely based on item text can recommend new items without re-training, and meanwhile gain better performance on known items.

2 METHODOLOGY

In this section, we present the proposed transferable sequential **Rec**ommendation approach based on **V**ector-**Q**uantized item indices, named **VQ-Rec**.

2.1 Approach Overview

Task formulation. We consider the sequential recommendation task setting that *multi-domain* interaction data is available as training (or pre-training) data. Formally, the interaction data of a user in some domain can be denoted as an interaction sequence $s = \{i_1, i_2, \ldots, i_n\}$ (in chronological order), where each interacted item i is associated with a unique item ID and text data, *e.g.,* title or description (*item text*). Since a user is likely to interact with items from multiple domains, we can derive multiple interaction sequences for a user. Considering the large semantic gap across different domains [18], we don't combine the multiple interaction sequences of a user into a single sequence, but instead keep these sequences per domain. Note that the item IDs are not explicitly utilized to generate item representations in our approach. The task goal is to pre-train a transferable sequential recommender that can effectively adapt to new domains (unseen in training data).

Solution overview. To develop the sequential recommender, we adopt the popular Transformer architecture [24] as the backbone of our approach. It is built on the self-attention mechanism, taking as input item embeddings and positional embeddings at each time step. Unlike previous related studies [18], we don't include any additional components (*e.g.,* adaptors) into the Transformer architecture, but instead learn transferable item representations for feeding the backbone. The key novelty of our approach lies in the new item representation scheme for sequential recommenders. In this scheme, we first map item text into a vector of discrete indices (called an *item code*), and then employ these indices to lookup the *code embedding table* for deriving item representations. Such a scheme can be denoted as "*text* $\Longrightarrow$ *code* $\Longrightarrow$ *representation*", which removes the tight binding between item text and item representations. In order to learn and transfer such item representations, we further propose specific strategies for *contrastive recommender pre-training* and *cross-domain recommender fine-tuning*.

The overall framework of the proposed approach VQ-Rec is depicted in Figure 1. We consider three key components for developing transferable recommenders: (i) how to represent the items with vector-quantized code representation (Section 2.2); (ii) how to train the recommenders based on the new representation scheme

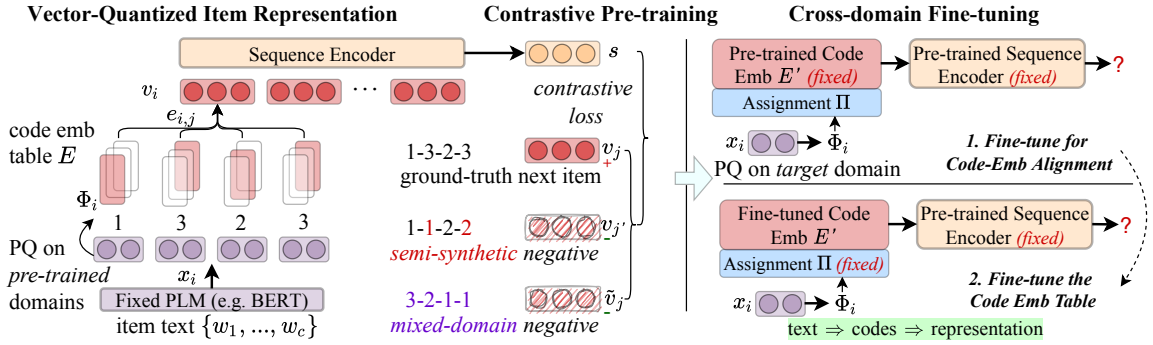

Figure 1: The overall framework of the proposed vector-quantized code-based transferable sequential recommender VQ-Rec.

(Section 2.3); (iii) how to transfer the pre-trained recommender to new domains (Section 2.4).

2.2 Vector-Quantized Item Representation

As introduced before, we propose a two-step item representation scheme: (i) item text is encoded by PLMs into a vector of discrete codes (Section 2.2.1), and (ii) discrete codes are employed to lookup the code embedding table to generate the item representation (Section 2.2.2). Next, we present the representation scheme in detail.

2.2.1 Vector-Quantized Code Learning. In this part, we first study how to map item text into a discrete code. In order to leverage the generality of natural language text, we first encode the descriptive text of items into text encodings via PLMs. Then, we build a mapping between text encodings and discrete codes based on optimized product quantization. Such a process is described as follows:

(1) item text $\overset{PLM}{\Longrightarrow}$ text encodings. We utilize the widely used BERT model [9] to encode the text information of items. Given an item i, we insert a special token `[CLS]` at the beginning of its item text t_i (denoted by $\{w_1, \ldots, w_c\}$), and then feed this extended text to BERT for obtaining the text encoding of item i as:

$$\boldsymbol{x}_i = \text{BERT}([\texttt{[CLS]}; w_1; \ldots; w_c]), \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^{d_W}$ is the final hidden vector corresponding to the first input token (`[CLS]`), and "[;]" is the concatenation operation. Note that the BERT encoder is fixed in our approach.

(2) text encodings $\overset{PQ}{\Longrightarrow}$ discrete codes. Next, we consider mapping the text encoding $\boldsymbol{x}_i$ to a vector of discrete codes for item i, based on product quantization (PQ) [20]. PQ defines D sets of vectors, each of which corresponds to M centroid embeddings with dimension d_W/D. Formally, we denote $\boldsymbol{a}_{k,j} \in \mathbb{R}^{d_W/D}$ as the j-th centroid embedding for the k-th vector set. Given the text encoding $\boldsymbol{x}_i$, PQ first splits it into D sub-vectors $\boldsymbol{x}_i = \left[\boldsymbol{x}_{i,1}; \ldots; \boldsymbol{x}_{i,D}\right]$. Then for each sub-vector, PQ selects the *index* of the nearest centroid embedding from the corresponding set to compose its discrete code (*i.e.,* a vector of indices). For the k-th sub-vector of item i, formally we have the selected index as:

$$c_{i,k} = \arg\min_j \|\boldsymbol{x}_{i,k} - \boldsymbol{a}_{k,j}\|^2 \in \{1, 2, \ldots, M\}, \tag{2}$$

where $c_{i,k}$ is the k-th dimension of the discrete code representation for item i. To learn these PQ centroid embeddings (*i.e.,* $\{\boldsymbol{a}_{k,j}\}$), we adopt a popular method *optimized product quantization (OPQ)* [13], and then optimize centroid embeddings based on the text encodings of items from all the training domains. After centroid embedding learning, we can perform the index assignment (Eq. (2)) independently for each dimension, and obtain the discrete code representation $\boldsymbol{c}_i = (c_{i,1}, \ldots, c_{i,D})$ for item i.

2.2.2 Code Embedding Lookup as Item Representations. Given the learned discrete item codes, we can directly perform the embedding lookup to generate item representations.

Since our approach uses D-dimensional discrete indices as item codes, denoted by $(c_{i,1}, \ldots, c_{i,D})$, we set up D *code embedding matrices* (called *code embedding table*) for lookup, each of which corresponds a dimension of the code representation. For each dimension k, the discrete codes of all the items share a common code embedding matrix $\boldsymbol{E}^{(k)} \in \mathbb{R}^{M \times d_V}$, where d_V is the dimension of the final item representations. Next we can obtain the code embeddings for item i by embedding lookup as $\{\boldsymbol{e}_{1,c_{i,1}}, \ldots, \boldsymbol{e}_{D,c_{i,D}}\}$, where $\boldsymbol{e}_{k,c_{i,k}} \in \mathbb{R}^{d_V}$ is the $c_{i,k}$-th row of parameter matrix $\boldsymbol{E}^{(k)}$. Note that the code embedding $\boldsymbol{e}_{k,j} \in \mathbb{R}^{d_V}$ is different from the corresponding PQ centroid embedding $\boldsymbol{a}_{k,j} \in \mathbb{R}^{d_W/D}$. We randomly initialize $\boldsymbol{E}_k$ for pre-training. After getting the code embeddings of item i, we further aggregate them to generate the final item representation:

$$\boldsymbol{v}_i = \text{Pool}\left(\left[\boldsymbol{e}_{1,c_{i,1}}; \ldots; \boldsymbol{e}_{D,c_{i,D}}\right]\right), \tag{3}$$

where $\boldsymbol{v}_i \in \mathbb{R}^{d_V}$ is the derived item representations, and $\text{Pool}(\cdot) : \mathbb{R}^{D \times d_V} \to \mathbb{R}^{d_V}$ is the mean pooling function.

2.2.3 Representation Distinguishability vs. Code Uniformity. To make accurate recommendations, the item representations should be distinguishable over a large candidate space, especially for those with similar text encodings. In our case, we should try to avoid the collision (*i.e.,* assigning the same code for two items) in the discrete codes between any two items. It has been shown by Zhan et al. [62] that the minimum collision probability is achieved when vectors are equally quantized to all possible discrete codes. As a result, ideally, the discrete codes should be *uniformly* distributed for maximum distinguishability, *i.e.,* code $c_{i,k}$ has equal probability of be each of $\{1, 2, \ldots, M\}$. According to previous empirical results [59, 61, 62], the used technique for training OPQ, *i.e.,* K-means, tends to generate clusters with a relatively uniform distribution on the

cluster sizes. Such results indicate that OPQ can generate discrete code representations with strong distinguishability for items.

2.3 Contrastive Recommender Pre-training

Compared with direct text mapping methods [10, 18], it is more difficult to optimize our item representation approach, since it involves discrete codes and employs a two-step representation mapping. In the following, we first introduce the sequential encoder architecture and then present our proposed contrastive pre-training tasks. To improve the sequential recommenders training, we propose to use both *mixed-domain* and *semi-synthetic* negative instances.

2.3.1 Self-attentive Sequence Encoding. Given a sequence of item representations derived from the vector-quantized item codes, we use a sequential encoder to obtain the sequence representation. Following SASRec [24], we adopt a widely-used self-attentive Transformer architecture [49]. In detail, a typical Transformer encoder consists of stacks of multi-head self-attention layers (denoted by $\text{Attn}(\cdot)$) and multilayer perceptron networks (denoted by $\text{FFN}(\cdot)$). The index representations $\boldsymbol{v}_i$ (Eqn. (3)) and absolute position embeddings $\boldsymbol{p}_j$ are summed up as input of Transformer encoder at position j. Then the update process can be formally written as:

$$\boldsymbol{f}_j^0 = \boldsymbol{v}_i + \boldsymbol{p}_j, \tag{4}$$

$$\boldsymbol{F}^{l+1} = \text{FFN}\left(\text{Attn}\left(\boldsymbol{F}^l\right)\right), l \in \{1, 2, \ldots, L\}, \tag{5}$$

where $\boldsymbol{F}^l = [\boldsymbol{f}_0^l; \ldots; \boldsymbol{f}_n^l] \in \mathbb{R}^{n \times d_V}$ denotes the hidden states at each position in the l-th layer. We take the final hidden state $\boldsymbol{f}_n^L$ corresponding to the n-th (last) position as the *sequence representation*.

2.3.2 Enhanced Contrastive Pre-training. To optimize the sequential recommender, a commonly used approach is to conduct a batch-level optimization objective with contrastive learning [4, 33, 58]. Specifically, for a batch of B training instances, where each instance is a pair of the sequential context (*i.e.,* historical interaction sequence) and the ground-truth next item (*i.e.,* positives). We encode them into representations $\{\langle \boldsymbol{s}_1, \boldsymbol{v}_1 \rangle, \ldots, \langle \boldsymbol{s}_B, \boldsymbol{v}_B \rangle\}$, where $\boldsymbol{s}_j$ denotes the j-th normalized sequence representations (Eqn. (5)) and $\boldsymbol{v}_j$ denotes the j-th normalized representation of positive item paired with $\boldsymbol{s}_j$ (Eqn. (3)). For conducting contrastive training, a key step is to sample a number of negatives to contrast the positive, which are usually obtained by random sampling.

However, random negative sampling cannot work well in our approach, due to two major reasons. Firstly, since our representation scheme involves the discrete code, it generates an exponential-sized combination of discrete indices (size M^D for D sets and M vectors in each set). While, the set of observed discrete indices in training data will be much smaller, which causes the *representation sparsity* issue. Besides, we also need to effectively alleviate the domain gap when learning with multi-domain training data. Considering the two issues, we design two types of enhanced negatives accordingly.

Semi-synthetic negatives. Besides the item indices that exist in the training set, we also consider synthesizing augmented item indices as negatives to alleviate the representation sparsity issue. However, fully-synthesized indices may be far away from ground-truth items in the sparse code representation space, which will be uninformative to guide contrastive learning [22]. As a result, we consider generating semi-synthetic codes based on true item codes as *hard negatives*. Given a true item code $\boldsymbol{c}_i$, the idea is to randomly replace each index according to a Bernoulli probability parameterized by $\rho \in (0, 1)$, while keeping the remaining indices unchanged. In this way, the item representations derived from semi-synthetic code can be given as follows:

$$\widetilde{\boldsymbol{v}}_i = \text{Emb-Pool}\left(\text{G}(\boldsymbol{c}_i)\right), \tag{6}$$

where $\widetilde{\boldsymbol{v}}_i$ is the representation of a semi-synthetic hard negative instance, $\text{Emb-Pool}(\cdot)$ is the embedding lookup and aggregation operations described in Section 2.2.2, and $\text{G}(\cdot)$ is a point-wise generation function:

$$\text{G}(c_{i,j}) = \begin{cases} \text{Uni}(\{1, \ldots, M\}), & X = 1 \\ c_{i,j}, & X = 0 \end{cases}, \tag{7}$$

where $X \sim \text{Bernoulli}(\rho)$, and $\text{Uni}(\cdot)$ samples the item code from the input set uniformly. Note that uniform sampling ensures the codes from the semi-synthetic indices follow a similar distribution as true items, which are shown to be distinguishable in Section 2.2.2.

Mixed-domain negatives. Different from previous next-item prediction models that use in-domain negatives [16, 24], we adopt mixed-domain items as negatives for enhancing the multi-domain fusion during pre-training. Due to efficiency concerns, we directly use the *in-batch* items as negatives (*i.e.,* the $B-1$ ground-truth items paired with other sequential contexts). Since we construct the batch by sampling from multiple domains, in-batch sampling can naturally generate mixed-domain negatives.

By integrating the two kinds of negatives, the pre-training objective can be formulated as:

$$\ell = -\frac{1}{B}\sum_{j=1}^{B} \log \frac{\exp\left(\boldsymbol{s}_j \cdot \boldsymbol{v}_j/\tau\right)}{\underbrace{\exp\left(\boldsymbol{s}_j \cdot \widetilde{\boldsymbol{v}}_j/\tau\right)}_{\text{semi-synthetic}} + \underbrace{\sum_{j'=1}^{B} \exp\left(\boldsymbol{s}_j \cdot \boldsymbol{v}_{j'}/\tau\right)}_{\text{mixed-domain}}}, \tag{8}$$

where τ is a temperature hyper-parameter. Note that here we include the positive in the mixed-domain samples for simplifying the notation, *i.e.,* one positive and $B-1$ negatives.

2.4 Cross-domain Recommender Fine-tuning

In the pre-training stage, we optimize the parameters in the code embedding matrices and Transformer, while the BERT encoder is fixed and PQ is performed independent of pre-training. Next, we continue to discuss how to conduct the fine-tuning in a cross-domain or cross-platform setting. During fine-tuning, we fix the Transformer sequence encoder (transferred across different domains), and only optimize the parameters involved in item representations, enabling parameter-efficient fine-tuning.

To further leverage the learned knowledge from pre-trained recommenders, we consider learning to transfer the code representation scheme ($M \times D$ PQ indices) and the code embedding table (M embedding matrices). Considering the two aspects, we decompose the fine-tuning optimization into two stages, namely fine-tuning code-embedding alignment and the code embedding table.

2.4.1 Fine-tuning for Code-Embedding Alignment. In order to transfer the item representation scheme, a straightforward approach is to directly reuse the discrete index set and corresponding embedding

table. However, such a simple way neglects the large semantic gap across different domains, thus leading to a weak transfer capacity to downstream domains. Our solution is to only reuse the discrete index set, and rebuild the mapping from indices to embeddings.

Permutation-based code-embedding alignment. To generate the item codes in a downstream domain, we re-train new PQ centroids for capturing domain-specific semantic characteristics. By sharing the code set, we map a new item i into $\hat{\boldsymbol{c}}_i \in \mathbb{N}_+^D$ through Eqn. (1)-(2), with the same number of sets D and sub-vectors M. Since we keep all the discrete indices and code embeddings, we employ a permutation-based approach to re-learning the mapping relations (*i.e.,* new lookup scheme) to associate indices with code embeddings. For each dimension k of the discrete indices, we formulate the embedding alignment as a matrix $\boldsymbol{\Pi}_k \in \{0,1\}^{M\times M}$. To enforce a bijection alignment, $\boldsymbol{\Pi}_k$ should be a permutation matrix that has exactly one entry of 1 in each row and each column and 0s elsewhere. Formally, the aligned code embedding table under a new domain $\hat{\boldsymbol{E}}^{(k)}$ can be given as:

$$\hat{\boldsymbol{E}}^{(k)} = \boldsymbol{\Pi}_k \cdot \boldsymbol{E}^{(k)}. \tag{9}$$

Alignment optimization. To learn the code-embedding alignment matrices $\boldsymbol{\Pi}_k$, we optimize the corresponding parameters by a conventional next-item prediction objective. Given the sequential context, we can predict the next item according to the following probability:

$$P(i_{t+1}|i_1,\ldots,i_t) = \text{Softmax}(\hat{\boldsymbol{s}} \cdot \hat{\boldsymbol{v}}_{i_{t+1}}), \tag{10}$$

where $\hat{\boldsymbol{s}}$ is the output of the pre-trained sequential encoder that takes as input $\hat{\boldsymbol{v}}_{i_1},\ldots,\hat{\boldsymbol{v}}_{i_t}$, and the item representations $\hat{\boldsymbol{v}}_i$ can be derived based on the code $\hat{\boldsymbol{c}}_i$ in a similar way in Eqn. (6). To make the permutation matrices $\boldsymbol{\Pi}_k$ differentiable, we are inspired by Birkhoff's theorem [47]: any doubly stochastic matrix can be considered as a convex combination of permutation matrices. Following this idea, we use doubly stochastic matrices to simulate the permutation. Concretely, we start by randomly initializing a parameter matrix for each dimension $\boldsymbol{\Theta}_k \in \mathbb{R}^{M\times M}$, then convert it into the doubly stochastic matrix via Gumbel-Sinkhorn algorithm [36]. Then, we optimize $\{\boldsymbol{\Theta}_1,\ldots,\boldsymbol{\Theta}_D\}$ according to the next item probability (Eqn. (10)) using cross-entropy loss, while fixing the remaining parameters in the pre-trained recommender.

2.4.2 Fine-tuning the Code Embedding Table. After code-embedding alignment, we continue to fine-tune the permuted code embedding table for increasing the representation capacity of fitting the downstream domains. To be specific, we optimize the parameters $\hat{\boldsymbol{E}}^{(1)},\ldots,\hat{\boldsymbol{E}}^{(D)}$ in Eqn. (9) according to the next-item prediction loss in Eqn. (10). In this stage, the parameters $\{\boldsymbol{\Theta}_1,\ldots,\boldsymbol{\Theta}_D\}$ in Section 2.4.1 are fixed. The fine-tuned VQ-Rec does not rely on item IDs and can be applied in an *inductive setting*, *i.e.,* recommending new items without re-training the model. When a new item emerges, one can encode its item text into discrete indices, and then obtain the corresponding item representations by embedding lookup.

2.5 Discussion

In this part, we highlight the merits of the proposed VQ-Rec approach in the following three aspects.

Table 1: Statistics of the preprocessed datasets. "Avg. n" denotes the average length of item sequences. "Avg. c" denotes the average number of words in item text.

Datasets	#Users	#Items	#Inters.	Avg. n	Avg. c
Pre-trained	1,361,408	446,975	14,029,229	13.51	139.34
Scientific	8,442	4,385	59,427	7.04	182.87
Pantry	13,101	4,898	126,962	9.69	83.17
Instruments	24,962	9,964	208,926	8.37	165.18
Arts	45,486	21,019	395,150	8.69	155.57
Office	87,436	25,986	684,837	7.84	193.22
Online Retail	16,520	3,469	519,906	26.90	27.80

• *Capacity.* By leveraging the generality of text semantics, the learned discrete codes and code embeddings can effectively capture transferable patterns across different domains. Unlike existing related studies [10, 18], our approach doesn't directly map the text encodings into item representations, thus it can avoid overemphasizing the text similarity and is also more robust to the noise from text data. Besides, as discussed in Section 2.2.3, our approach can generate highly distinguishable code representations.

• *Flexibility.* As another difference with previous studies, we don't modify the underlying sequence encoder (*i.e.,* Transformer), without any minor change such as the incorporation of adaptors. Besides, text encoder and PQ are independent (in terms of optimization) from the underlying sequence encoder. These decoupled designs make it flexible to extend our approach with various choices of PLM, sequential encoder and discrete coding method.

• *Efficiency.* Our approach introduces three specific designs for efficient model training and utilization: (1) fixed text encoder; (2) independently learned discrete codes; (3) fixing the sequence encoder during fine-tuning. Besides, compared to existing methods like UniSRec [18], VQ-Rec has better time complexity when deriving transferable item representations ($O(d_V D)$ *vs.* $O(d_W d_V D)$). As shown in Section 2.2.2, VQ-Rec does not require any matrix multiplication as previous methods based on adaptors.

3 EXPERIMENTS

In this section, we empirically demonstrate the effectiveness and transferability of the proposed approach VQ-Rec.

3.1 Experimental Setup

3.1.1 Datasets. We conduct experiments on public benchmarks for transferable recommender evaluation. Five domains from Amazon review datasets [38] are used for pre-training (*Food*, *Home*, *CDs*, *Kindle*, and *Movies*). Then the pre-trained model will be transferred to five downstream cross-domain datasets (*Scientific*, *Pantry*, *Instruments*, *Arts*, and *Office*, all from Amazon) and one cross-platform dataset (*Online Retail*[1], a UK-based online retail platform).

Following Hou et al. [18], we filter users and items with fewer than five interactions. We then group the interactions in each subdataset by users and sort them in chronological order. For the descriptive item text, we concatenate fields including *title*, *categories*, and *brand* for sub-datasets from Amazon, and use the *Description*

[1] https://www.kaggle.com/carrie1/ecommerce-data

Table 2: Performance comparison of different models. The best and the second-best performance is denoted in bold and underlined fonts, respectively. "R@K" is short for "Recall@K" and "N@K" is short for "NDCG@K", respectively. The features used for item representations of each compared model have been listed, whether ID, text (T), or both (ID+T).

Scenario	Dataset	Metric	SASRec $_{\text{ID}}$	BERT4Rec $_{\text{ID}}$	FDSA $_{\text{ID+T}}$	S^3-Rec $_{\text{ID+T}}$	RecGURU $_{\text{ID}}$	SASRec $_{\text{T}}$	ZESRec $_{\text{T}}$	UniSRec $_{\text{T}}$	VQ-Rec $_{\text{T}}$
Cross-Domain	Scientific	R@10	0.1080	0.0488	0.0899	0.0525	0.1023	0.0994	0.0851	0.1188	**0.1211**
		N@10	0.0553	0.0243	0.0580	0.0275	0.0572	0.0561	0.0475	0.0641	**0.0643**
		R@50	0.2042	0.1185	0.1732	0.1418	0.2022	0.2162	0.1746	**0.2394**	0.2369
		N@50	0.0760	0.0393	0.0759	0.0468	0.0786	0.0815	0.0670	**0.0903**	0.0897
	Pantry	R@10	0.0501	0.0308	0.0395	0.0444	0.0469	0.0585	0.0454	0.0636	**0.0660**
		N@10	0.0218	0.0152	0.0209	0.0214	0.0209	0.0285	0.0230	**0.0306**	0.0293
		R@50	0.1322	0.1030	0.1151	0.1315	0.1269	0.1647	0.1141	0.1658	**0.1753**
		N@50	0.0394	0.0305	0.0370	0.0400	0.0379	0.0523	0.0378	**0.0527**	**0.0527**
	Instruments	R@10	0.1118	0.0813	0.1070	0.1056	0.1113	0.1127	0.0783	0.1189	**0.1222**
		N@10	0.0612	0.0620	**0.0796**	0.0713	0.0681	0.0661	0.0497	0.0680	0.0758
		R@50	0.2106	0.1454	0.1890	0.1927	0.2068	0.2104	0.1387	0.2255	**0.2343**
		N@50	0.0826	0.0756	0.0972	0.0901	0.0887	0.0873	0.0627	0.0912	**0.1002**
	Arts	R@10	0.1108	0.0722	0.1002	0.1003	0.1084	0.0977	0.0664	0.1066	**0.1189**
		N@10	0.0587	0.0479	**0.0714**	0.0601	0.0651	0.0562	0.0375	0.0586	0.0703
		R@50	0.2030	0.1367	0.1779	0.1888	0.1979	0.1916	0.1323	0.2049	**0.2249**
		N@50	0.0788	0.0619	0.0883	0.0793	0.0845	0.0766	0.0518	0.0799	**0.0935**
	Office	R@10	0.1056	0.0825	0.1118	0.1030	0.1145	0.0929	0.0641	0.1013	**0.1236**
		N@10	0.0710	0.0634	**0.0868**	0.0653	0.0768	0.0582	0.0391	0.0619	0.0814
		R@50	0.1627	0.1227	0.1665	0.1613	0.1757	0.1580	0.1113	0.1702	**0.1957**
		N@50	0.0835	0.0721	**0.0987**	0.0780	0.0901	0.0723	0.0493	0.0769	0.0972
Cross-Platform	Online Retail	R@10	0.1460	0.1349	0.1490	0.1418	0.1467	0.1380	0.1103	0.1449	**0.1557**
		N@10	0.0675	0.0653	0.0719	0.0654	0.0658	0.0672	0.0535	0.0677	**0.0730**
		R@50	0.3872	0.3540	0.3802	0.3702	0.3885	0.3516	0.2750	0.3604	**0.3994**
		N@50	0.1201	0.1131	0.1223	0.1154	0.1188	0.1137	0.0896	0.1149	**0.1263**

field in the Online Retail dataset. The item text is truncated to a length of 512. The statistics of the preprocessed datasets are summarized in Table 1.

3.1.2 Compared Methods. We compare the proposed approach with the following baseline methods:

• **SASRec** [24] utilizes a self-attentive model to capture item correlations. We implement two versions, either with (1) conventional ID embeddings, or (2) fine-tuned BERT representations of item text as basic item representations.

• **BERT4Rec** [44] adopts a bi-directional self-attentive model with a cloze objective for sequence modeling.

• **FDSA** [63] models item and feature sequences with separate self-attentive sequential models.

• **S^3-Rec** [71] captures feature-item correlations at the pre-training stage with mutual information maximization objectives.

• **RecGURU** [29] proposes an adversarial learning paradigm to pre-train user representations via an auto-encoder.

• **ZESRec** [10] encodes item text with PLM as basic item representations. For fair comparison, ZESRec is pre-trained on the same datasets as VQ-Rec.

• **UniSRec** [18] equips textual item representations with an MoE-enhanced adaptor for domain fusion and adaptation. Both item-sequence and sequence-sequence contrastive learning tasks are designed for pre-training transferable sequence representations.

For our approach, we first pre-train one **VQ-Rec** model on the mixture of item sequences from the pre-training datasets. The pre-trained model will be fine-tuned to each downstream dataset.

3.1.3 Evaluation Settings. Following previous works [18, 68, 71], we adopt two widely used ranking metrics, Recall@K and NDCG@K, where $K \in \{10, 50\}$. For dataset splitting, we apply the leave-one-out strategy, *i.e.,* the latest interacted item as test data, the item before the last one as validation data. The ground-truth item of each sequence is ranked among all the other items while evaluating. We finally report the average scores of all test users.

3.1.4 Implementation Details. We implement our models based on Faiss ANNS library [21] and RecBole [67, 69]. We use ($M = 32$) × ($D = 256$) PQ indices as the code representation scheme. We pre-train VQ-Rec for 300 epochs with temperature $\tau = 0.07$ and semi-synthetic ratio $\rho = 0.75$. The iteration number for Gumbel-Sinkhorn algorithm is set to 3. The main baseline results are taken from Hou et al. [18] directly. For other models, we search the hyper-parameters to find optimal results. The batch size is set to 2,048. The learning rate is tuned in $\{0.0003, 0.001, 0.003\}$. The number of permutation learning epochs is tuned in $\{3, 5, 10\}$. The models with the highest NDCG@10 results on the validation set are selected for evaluation on the test set. We adopt early stopping with a patience of 10 epochs.

3.2 Overall Performance

We compare VQ-Rec with the baseline methods on six benchmarks and report the results in Table 2.

For the baseline methods, we can see that text-based models (*i.e.,* SASRec (T), ZESRec, and UniSRec) perform better than other approaches on small-scale datasets (*e.g.,* Scientific and Pantry). For

Table 3: Ablation analysis on three downstream datasets. The best and the second-best performance is denoted in bold and underlined fonts, respectively.

Variants	Scientific				Office				Online Retail			
	R@10	N@10	R@50	N@50	R@10	N@10	R@50	N@50	R@10	N@10	R@50	N@50
(0) VQ-Rec	**0.1211**	**0.0643**	**0.2369**	**0.0897**	0.1236	0.0814	**0.1957**	0.0972	0.1557	0.0730	**0.3994**	0.1263
(1) *w/o* Pre-training	0.1104	0.0593	0.2238	0.0839	0.1108	0.0666	0.1804	0.0818	0.1535	0.0717	0.3953	0.1244
(2) *w/o* Semi-synthetic NS	0.1194	0.0631	0.2358	0.0886	0.1227	0.0809	0.1951	0.0968	0.1529	0.0702	0.3938	0.1230
(3) *w/o* Fine-tuning	0.0640	0.0361	0.0909	0.0421	0.0261	0.0150	0.0373	0.0174	0.0197	0.0095	0.0419	0.0142
(4) Reuse PQ Index Set	0.1168	0.0618	0.2267	0.0859	0.1182	0.0773	0.1869	0.0922	0.1521	0.0707	0.3917	0.1232
(5) *w/o* Code-Emb Alignment	0.1183	0.0612	0.2355	0.0867	**0.1242**	0.0824	0.1956	**0.0980**	0.1515	0.0695	0.3924	0.1224
(6) Random Code	0.0905	0.0494	0.1769	0.0683	0.1134	**0.0837**	0.1698	0.0960	**0.1589**	**0.0769**	0.3933	**0.1282**

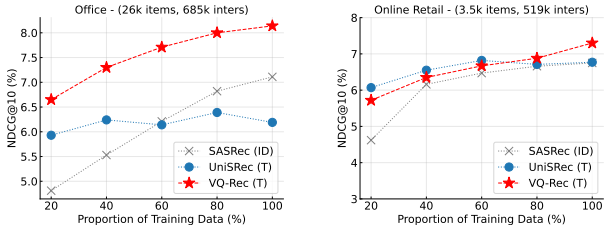

Figure 2: Performance comparison w.r.t. increasing training data on "Office" and "Online Retail" datasets. The x-axis coordinates denote the proportion of original training data that the recommenders are trained on.

these datasets where interactions are not sufficient enough to train a powerful ID-based recommender, the text-based methods may benefit from the text characteristics. While for the models that incorporate item IDs (*i.e.,* SASRec and BERT4Rec), they perform better on those datasets with more interactions (*e.g.,* Arts, Office and Online Retail), showing that over-emphasizing the text similarity may lead to sub-optimal results.

For the proposed approach VQ-Rec, it achieves the best or the second-best performance on all datasets. On small-scale datasets, the results show that the proposed discrete indices can preserve text semantics to give proper recommendations. On large-scale datasets, VQ-Rec can be well-trained to capture sequential characteristics. Note that VQ-Rec can be applied in an inductive setting, which can recommend new items without re-training the model. The experimental results also show that with carefully designed encoding schemes as well as large-scale pre-training, inductive recommenders can also outperform conventional transductive models.

3.3 Ablation Study

We analyze how each of the proposed components affects final performance. Table 3 shows the performance of our default method and its six variants on three representative datasets, including one small-scale dataset (Scientific), one large-scale dataset (Office), and one cross-platform dataset (Online Retail).

(1) *w/o Pre-training*: Without pre-training on multiple domains, the variant performs worse than VQ-Rec on all datasets. The results indicate that VQ-Rec can learn and transfer general sequential patterns of discrete codes to downstream domains or platforms.

(2) *w/o Semi-synthetic NS*: Removing semi-synthetic negative samples from the pre-training loss (Eqn. (8)), VQ-Rec may suffer from sparsity issues and get sub-optimal results.

(3) *w/o Fine-tuning*: The performance drops sharply if the pre-trained model is not fine-tuned, additionally showing that transferring to domains with varied semantics can be difficult.

(4) *Reuse PQ Index Set*: We directly encode the downstream item indices using the PQ centroids that are used for pre-training. The large semantic gap makes the indices follow a long-tail distribution. Due to the reduced distinguishability, this variant performs worse.

(5) *w/o Code-Emb Alignment*: In this variant, we remove matrices $\mathbf{\Pi}_k$ in Eqn. (9) that are used to align the pre-trained embeddings to downstream codes. The results show that permutation-based alignment network can generally improve performance.

(6) *Random Code*: In this variant, we randomly assign the pre-trained embeddings to downstream items. This variant generally has worse performance than the default method, showing that the learned vector-quantized codes can preserve text characteristics. We also note that on Online Retail, this variant has slightly better performance, mainly because the item text is relatively short (*i.e.,* only 27.8 words on average). The results suggest that informative item text is essential for deploying such text-based recommenders.

3.4 Further Analysis

3.4.1 Capacity Analysis w.r.t. Increasing Training Data. To show whether the proposed discrete code embeddings have a better capacity, we simulate a scenario where we have increasing training data. In detail, we train the models with different proportions of training interactions (*i.e.,* 20% – 100%) and present the performance on the test set in Figure 2. As we can see, the performance of VQ-Rec can always improve with more training data, outperforming the compared methods. The results indicate that VQ-Rec has a better capacity to fit training sequences.

3.4.2 Transferability Analysis w.r.t. Downstream Datasets. In this part, we show the relative transferring improvements (*i.e., w/* pre-training compared to *w/o* pre-training) in Figure 3 on each downstream dataset. We can see that due to the capacity issue of PLM-based item representations, UniSRec may suffer from negative transfer issues on several datasets (*i.e.,* Arts, Office, and Online Retail).

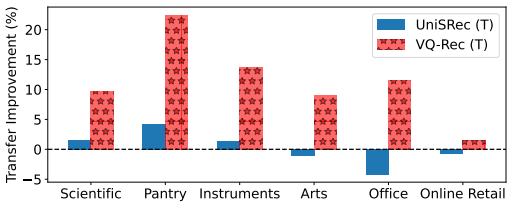

Figure 3: Relative improvement comparison ratios through transferring for Recall@10 w.r.t. different downstream datasets. Below the line denotes negative transfer.

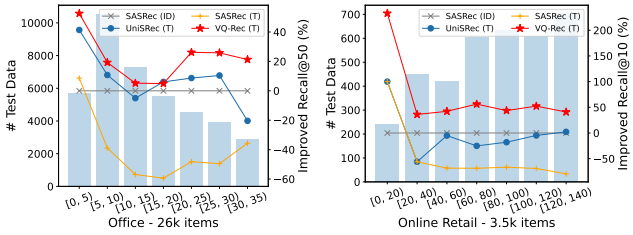

Figure 4: Performance comparison w.r.t. cold-start items on "Office" and "Online Retail" datasets. The bar graph denotes the number of interactions in test data for each group. The line chart denotes the relative improvement ratios compared with the baseline method SASRec (ID).

In contrast, VQ-Rec can benefit from pre-training on all six experimental datasets and has a maximum of 20+% improvement. The results show that the proposed techniques can help recommenders transfer to those downstream scenarios with varied semantics.

3.4.3 Cold-Start Item Analysis. One of the motivations to develop transferable recommenders is to alleviate the cold-start recommendation issue. We split the test data into different groups according to the popularity of ground-truth items and present the results in Figure 4. Although recommenders that directly map text to textual item representations have good performance on cold-start groups, *e.g.,* $[0, 5)$ for Office and $[0, 20)$ for Online Retail, their performance drops on those popular groups. In contrast, VQ-Rec has better performance than SASRec in all groups, especially in cold-start groups. The results indicate that recommendations on these long-tail items may benefit from the proposed pre-training techniques.

4 RELATED WORK

Sequential recommendation. Sequential recommendation [16, 24, 39] aims to predict the next interacted items based on historical interaction sequences. Early works follow the Markov Chain assumption [39], while recent studies mainly focus on designing different neural network models, including Recurrent Neural Network (RNN) [16, 30], Convolutional Neural Network (CNN) [45], Transformer [15, 17, 24, 44], Graph Neural Network (GNN) [3, 54] and Multilayer Perceptron (MLP) [72]. However, most of these approaches are developed based on item IDs [24] or attributes [63] defined on one specific domain, making it difficult to leverage behavior sequences from other domains or platforms. More recently, there have been attempts that employ textual or visual features as transferable item representations [10, 18, 37, 50]. Furthermore, several studies propose constructing a unified model to solve multiple recommendation-oriented tasks based on PLMs [8, 14]. Our work is built on these studies, but has a different focus: we incorporate discrete codes to decouple the binding between text encodings and item representations, enhancing the representation capacity with specially designed pre-training and fine-tuning strategies.

Transfer learning for recommendation. To alleviate the data sparsity and cold-start issues that broadly exist in recommender systems, researchers have explored the idea of knowledge transfer from other domains [57, 73, 74], markets [1] or platforms [32]. Existing approaches mainly rely on shared information between the source and target domains to conduct the transfer, *e.g.,* common users [19, 53, 56, 60], items [43, 73] or attributes [46]. Recently, pre-trained language models [9, 35] (PLM) have shown to be general semantic bridges to connect different tasks or domains [2], and several studies propose to encode the associated text of items by PLMs as universal item representations [10, 14, 18]. Based on pre-training universal item representations, one can transfer the fused knowledge to downstream domains without overlapping users or items. However, these studies usually enforce a tight binding between the text encodings from PLMs and the final item representations, which might over-emphasize the effect of text features. In contrast, we propose a novel two-step representation scheme based on discrete codes, which is endowed with a more strong representation capacity for enhancing the cross-domain recommendation.

Sparse representation for recommendation. Learning sparse codes [28, 51, 52, 66] is a widely adopted way to represent data objects in machine learning, inspiring a variety of studies related to product quantization [5, 13, 20, 27], multi-way compact embedding [6, 7], semantic hashing [26, 40], etc. In contrast to continuous representation, it seeks a sparse scheme for capturing the most salient representation dimensions. Specifically, discrete representations are also applied in recommender systems [34, 41, 48, 55, 64, 65, 70], and existing studies mainly aim to develop efficient recommendation algorithms in both *memory* and *time* based on the sparse representations, so as to develop large-scale recommender systems [23, 25, 31, 42]. Different from these studies, our work aims to leverage the generality of text semantics for learning transferable item representations based on a pre-trained approach.

5 CONCLUSIONS

In this work, we proposed VQ-Rec to learn vector-quantized item representations for transferable sequential Recommenders. Different from existing approaches that directly map text encodings from PLMs into item representations, we established a two-step item representation scheme, in which it firstly maps text encodings into discrete codes and then employs embedding lookup to derive item representations. To pre-train our approach on multi-domain interaction data, we employed both mixed-domain and semi-synthetic code representations as hard negatives. We further proposed a permutation-based network to learn domain-specific code-embedding alignment, which can effectively adapt to downstream domains. Extensive experiments conducted on six transferring benchmarks demonstrated the effectiveness of VQ-Rec.

ACKNOWLEDGMENTS

This work was partially supported by National Natural Science Foundation of China under Grant No. 62222215, Beijing Natural Science Foundation under Grant No. 4222027, and Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098. Xin Zhao is the corresponding author.

REFERENCES

[1] Hamed Bonab, Mohammad Aliannejadi, Ali Vardasbi, Evangelos Kanoulas, and James Allan. 2021. Cross-Market Product Recommendation. In *CIKM*.
[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*.
[3] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential Recommendation with Graph Neural Networks. In *SIGIR*.
[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*.
[5] Ting Chen, Lala Li, and Yizhou Sun. 2020. Differentiable product quantization for end-to-end embedding compression. In *ICML*.
[6] Ting Chen, Lala Li, and Yizhou Sun. 2020. Differentiable Product Quantization for End-to-End Embedding Compression. In *ICML*.
[7] Ting Chen, Martin Renqiang Min, and Yizhou Sun. 2018. Learning K-way D-dimensional Discrete Codes for Compact Embedding Representations. In *ICML*.
[8] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. M6-Rec: Generative Pretrained Language Models are Open-Ended Recommender Systems. *arXiv:2205.08084* (2022).
[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.
[10] Hao Ding, Yifei Ma, Anoop Deoras, Yuyang Wang, and Hao Wang. 2021. Zero-Shot Recommender Systems. *arXiv:2105.08318* (2021).
[11] Xinyan Fan, Jianxun Lian, Wayne Xin Zhao, Zheng Liu, Chaozhuo Li, and Xing Xie. 2022. Ada-Ranker: A Data Distribution Adaptive Ranking Paradigm for Sequential Recommendation. In *SIGIR*.
[12] Ignacio Fernández-Tobías and Iván Cantador. 2014. Exploiting Social Tags in Matrix Factorization Models for Cross-domain Collaborative Filtering.. In *CBRecSys@ RecSys*.
[13] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized product quantization. *TPAMI* (2013).
[14] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5). In *RecSys*.
[15] Zhankui He, Handong Zhao, Zhe Lin, Zhaowen Wang, Ajinkya Kale, and Julian J. McAuley. 2021. Locker: Locally Constrained Self-Attentive Sequential Recommendation. In *CIKM*.
[16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.
[17] Yupeng Hou, Binbin Hu, Zhiqiang Zhang, and Wayne Xin Zhao. 2022. CORE: Simple and Effective Session-based Recommendation within Consistent Representation Space. In *SIGIR*.
[18] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards Universal Sequence Representation Learning for Recommender Systems. In *KDD*.
[19] Guangneng Hu, Yu Zhang, and Qiang Yang. 2018. CoNet: Collaborative Cross Networks for Cross-Domain Recommendation. In *CIKM*.
[20] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *TPAMI* (2011).
[21] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Trans. Big Data* (2021).
[22] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. 2020. Hard negative mixing for contrastive learning. In *NeurIPS*.
[23] Wang-Cheng Kang, Derek Zhiyuan Cheng, Ting Chen, Xinyang Yi, Dong Lin, Lichan Hong, and Ed H. Chi. 2020. Learning Multi-granular Quantized Embeddings for Large-Vocab Categorical Features in Recommender Systems. In *TheWebConf*.
[24] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*.
[25] Wang-Cheng Kang and Julian McAuley. 2019. Candidate generation with binary codes for large-scale top-n recommendation. In *CIKM*.
[26] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *ICLR*.
[27] István Kollár. 1984. Statistical theory of quantization: Results and limits. *Periodica Polytechnica Electrical Engineering (Archives)* (1984).
[28] Honglak Lee, Alexis J. Battle, Rajat Raina, and Andrew Y. Ng. 2006. Efficient sparse coding algorithms. In *NIPS*.
[29] Chenglin Li, Mingjun Zhao, Huanming Zhang, Chenyun Yu, Lei Cheng, Guoqiang Shu, Beibei Kong, and Di Niu. 2022. RecGURU: Adversarial Learning of Generalized User Representations for Cross-Domain Recommendation. In *WSDM*.
[30] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *CIKM*.
[31] Defu Lian, Haoyu Wang, Zheng Liu, Jianxun Lian, Enhong Chen, and Xing Xie. 2020. Lightrec: A memory and search-efficient recommender system. In *Proceedings of The Web Conference 2020*.
[32] Tzu-Heng Lin, Chen Gao, and Yong Li. 2019. CROSS: Cross-platform Recommendation for Social E-Commerce. In *SIGIR*.
[33] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving Graph Collaborative Filtering with Neighborhood-enriched Contrastive Learning. In *TheWebConf*.
[34] Chenghao Liu, Tao Lu, Xin Wang, Zhiyong Cheng, Jianling Sun, and Steven C. H. Hoi. 2019. Compositional Coding for Collaborative Filtering. In *SIGIR*.
[35] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv:1907.11692* (2019).
[36] Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. 2018. Learning Latent Permutations with Gumbel-Sinkhorn Networks. In *ICLR*.
[37] Shanlei Mu, Yupeng Hou, Wayne Xin Zhao, Yaliang Li, and Bolin Ding. 2022. ID-Agnostic User Behavior Pre-training for Sequential Recommendation. *arXiv:2206.02323* (2022).
[38] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *EMNLP*.
[39] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*.
[40] Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Lawrence Carin, and Ricardo Henao. 2018. Nash: Toward end-to-end neural architecture for generative semantic hashing. In *ACL*.
[41] Hao-Jun Michael Shi, Dheevatsa Mudigere, Maxim Naumov, and Jiyan Yang. 2020. Compositional embeddings using complementary partitions for memory-efficient recommendation systems. In *KDD*.
[42] Shaoyun Shi, Weizhi Ma, Min Zhang, Yongfeng Zhang, Xinxing Yu, Houzhi Shan, Yiqun Liu, and Shaoping Ma. 2020. Beyond user embedding matrix: Learning to hash for modeling large-scale users in recommendation. In *SIGIR*.
[43] Ajit Paul Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In *SIGKDD*.
[44] 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 *CIKM*.
[45] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*.
[46] Jie Tang, Sen Wu, Jimeng Sun, and Hang Su. 2012. Cross-domain collaboration recommendation. In *SIGKDD*.
[47] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020. Sparse sinkhorn attention. In *ICML*.
[48] Jan Van Balen and Mark Levy. 2019. PQ-VAE: Efficient Recommendation Using Quantized Embeddings.. In *RecSys (Late-Breaking Results)*.
[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*.
[50] Jie Wang, Fajie Yuan, Mingyue Cheng, Joemon M Jose, Chenyun Yu, Beibei Kong, Zhijin Wang, Bo Hu, and Zang Li. 2022. TransRec: Learning Transferable Recommendation from Mixture-of-Modality Feedback. *arXiv:2206.06190* (2022).
[51] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. 2018. A Survey on Learning to Hash. *TPAMI* (2018).
[52] John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S. Huang, and Shuicheng Yan. 2010. Sparse Representation for Computer Vision and Pattern Recognition. *Proc. IEEE* (2010).
[53] Chuhan Wu, Fangzhao Wu, Tao Qi, Jianxun Lian, Yongfeng Huang, and Xing Xie. 2020. PTUM: Pre-training User Model from Unlabeled User Behaviors via Self-supervision. *arXiv:2010.01494* (2020).
[54] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. In *AAAI*.
[55] Yongji Wu, Defu Lian, Neil Zhenqiang Gong, Lu Yin, Mingyang Yin, Jingren Zhou, and Hongxia Yang. 2021. Linear-Time Self Attention with Codeword Histogram for Efficient Recommendation. In *WWW*.
[56] Chaojun Xiao, Ruobing Xie, Yuan Yao, Zhiyuan Liu, Maosong Sun, Xu Zhang, and Leyu Lin. 2021. UPRec: User-Aware Pre-training for Recommender Systems. *arXiv:2102.10989* (2021).
[57] Ruobing Xie, Qi Liu, Liangdong Wang, Shukai Liu, Bo Zhang, and Leyu Lin. 2022. Contrastive Cross-domain Recommendation in Matching. In *SIGKDD*.
[58] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *ICDE*.

[59] Hui Xiong, Junjie Wu, and Jian Chen. 2006. K-means clustering versus validation measures: a data distribution perspective. In *KDD*.
[60] Fajie Yuan, Guoxiao Zhang, Alexandros Karatzoglou, Joemon Jose, Beibei Kong, and Yudong Li. 2021. One person, one model, one world: Learning continual user representation without forgetting. In *SIGIR*. 696–705.
[61] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Jointly Optimizing Query Encoder and Product Quantization to Improve Retrieval Performance. In *CIKM*.
[62] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2022. Learning Discrete Representations via Constrained Clustering for Effective and Efficient Dense Retrieval. In *WSDM*.
[63] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. 2019. Feature-level Deeper Self-Attention Network for Sequential Recommendation. In *IJCAI*.
[64] Yan Zhang, Defu Lian, and Guowu Yang. 2017. Discrete personalized ranking for fast collaborative filtering from implicit feedback. In *AAAI*.
[65] Yan Zhang, Hongzhi Yin, Zi Huang, Xingzhong Du, Guowu Yang, and Defu Lian. 2018. Discrete Deep Learning for Fast Content-Aware Recommendation. In *WSDM*.
[66] Zheng Zhang, Yong Xu, Jian Yang, Xuelong Li, and David Zhang. 2015. A Survey of Sparse Representation: Algorithms and Applications. *IEEE Access* (2015).
[67] Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, Yushuo Chen, Lanling Xu, Gaowei Zhang, Zhen Tian, Changxin Tian, Shanlei Mu, Xinyan Fan, Xu Chen, and Ji-Rong Wen. 2022. RecBole 2.0: Towards a More Up-to-Date Recommendation Library. In *CIKM*.
[68] Wayne Xin Zhao, Zihan Lin, Zhichao Feng, Pengfei Wang, and Ji-Rong Wen. 2022. A Revisiting Study of Appropriate Offline Evaluation for Top-N Recommendation Algorithms. *TOIS* (2022).
[69] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2021. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. In *CIKM*.
[70] 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:2006.14827* (2020).
[71] 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 *CIKM*.
[72] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Filter-enhanced MLP is All You Need for Sequential Recommendation. In *TheWebConf*.
[73] Feng Zhu, Chaochao Chen, Yan Wang, Guanfeng Liu, and Xiaolin Zheng. 2019. DTCDR: A Framework for Dual-Target Cross-Domain Recommendation. In *CIKM*.
[74] Feng Zhu, Yan Wang, Chaochao Chen, Jun Zhou, Longfei Li, and Guanfeng Liu. 2021. Cross-Domain Recommendation: Challenges, Progress, and Prospects. In *IJCAI*.