Disentangled Self-Supervision in Sequential Recommenders

Jianxin Ma*
majx13fromthu@gmail.com
Tsinghua University, Beijing, China
Alibaba Group, China

Chang Zhou
ericzhou.zc@alibaba-inc.com
Alibaba Group, China

Hongxia Yang
yang.yhx@alibaba-inc.com
Alibaba Group, China

Peng Cui
cuip@tsinghua.edu.cn
Tsinghua University, Beijing, China

Xin Wang†
xin_wang@tsinghua.edu.cn
Tsinghua University, Beijing, China
Key Laboratory of Pervasive Computing, Ministry of Education, China

Wenwu Zhu†
wwzhu@tsinghua.edu.cn
Tsinghua University, Beijing, China
Key Laboratory of Pervasive Computing, Ministry of Education, China

ABSTRACT

To learn a sequential recommender, the existing methods typically adopt the sequence-to-item (seq2item) training strategy, which supervises a sequence model with a user's next behavior as the label and the user's past behaviors as the input. The seq2item strategy, however, is myopic and usually produces non-diverse recommendation lists. In this paper, we study the problem of mining extra signals for supervision by looking at the longer-term future. There exist two challenges: i) reconstructing a future sequence containing many behaviors is exponentially harder than reconstructing a single next behavior, which can lead to difficulty in convergence, and ii) the sequence of all future behaviors can involve many intentions, not all of which may be predictable from the sequence of earlier behaviors. To address these challenges, we propose a sequence-to-sequence (seq2seq) training strategy based on latent self-supervision and disentanglement. Specifically, we perform self-supervision in the latent space, i.e., reconstructing the representation of the future sequence as a whole, instead of reconstructing the items in the future sequence individually. We also disentangle the intentions behind any given sequence of behaviors and construct seq2seq training samples using only pairs of sub-sequences that involve a shared intention. Results on real-world benchmarks and synthetic data demonstrate the improvement brought by seq2seq training.

KEYWORDS

recommender systems; sequence model; disentangled representation learning; self-supervised learning; contrastive learning

ACM Reference Format:
Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled Self-Supervision in Sequential Recommenders. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403091

*Work done when he was a research intern at Alibaba Group.
†Corresponding authors.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403091

1 INTRODUCTION

Sequences of user behaviors in recommender systems represent a significant portion of the traffic in modern web and mobile applications. The central task related to this kind of sequence data is to recommend the next item to a target user based on this user's sequence of past behaviors such as clicking and bookmarking. Motivated by deep learning's expressive power in describing sequential data, recent efforts [18, 25, 49, 54] have gained impressive success on this task with deep sequential models, such as the recurrent neural networks and self-attention networks (aka., Transformers) [10, 51].

The standard approach for training the sequential models is to take a user's sequence of past behaviors as the input and use the user's next behavior as the supervision signals, i.e., to perform sequence-to-item (seq2item) training. However, seq2item training is myopic and can easily lead to non-diverse recommendation lists. For example, the sequence "shirt, shirt, shirt, shirt, shirt, trousers" contains far more consecutive sub-sequences whose corresponding labels are shirt and only a few sub-sequences whose labels are trousers. As a result, the algorithm trained via the seq2item strategy will tend to recommend shirts much more frequently after a user clicks a shirt, while in a real-world top-k recommender system we would like the algorithm to recommend both shirts and trousers in a more balanced manner when generating a page of k items [4]. Second, seq2item training is vulnerable if the next immediate behavior in the training data is irrelevant to the sequence of behaviors that happens before this new behavior. User nowadays have diverse and constantly-changing intentions, and may even click novel items merely out of curiosity regardless of the previous intentions.

In this paper, we study the sequential recommendation problem and mine extra signals for supervision by looking at the longer-term future, with the aim of complementing the standard sequence-to-item training strategy. Nevertheless, supervising a sequential model using a sequence of many future behaviors, instead of just the next single behavior, poses significant challenges:

- First, a future sequence of many behaviors is exponentially harder to reconstruct than a single next behavior. And it is inefficient to reconstruct the behaviors (such as clicking an

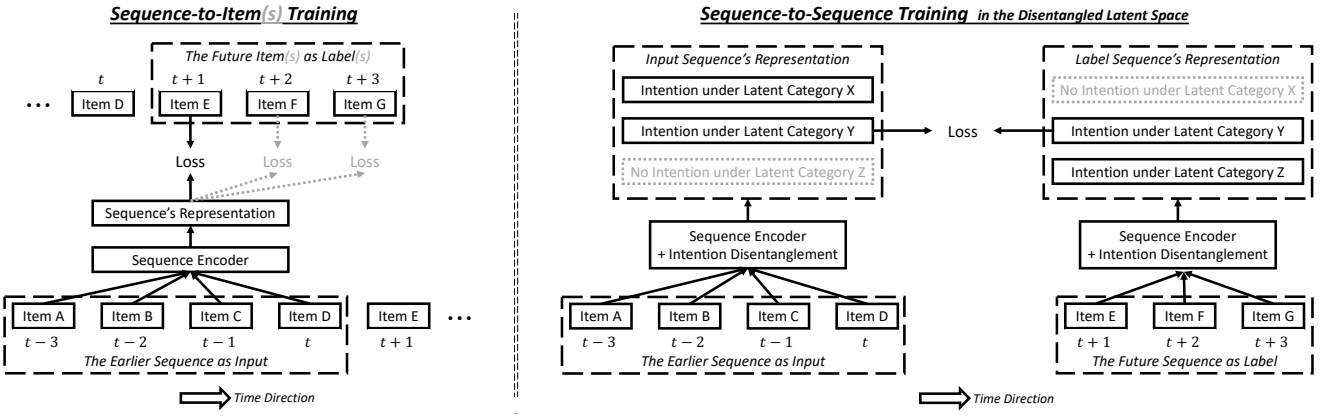


Figure 1: The sequence-to-item(s) training strategy in the literature directly reconstructs the future items, while our sequence-to-sequence training strategy reconstructs the disentangled representations of the future sequences and avoids explicitly reconstructing every item. In other words, we perform sequence-to-sequence self-supervision in a disentangled latent space.

item) one by one since there might be redundant supervision signals in the future sequence, e.g., many clicks reflecting the same intention.

- Second, the future sequence of behaviors, as the label of a training sample, can involve multiple constantly evolving user intentions. Notably, not all the intentions hidden in the future may be relevant to the earlier behavior sequence that serve as the input of the training sample. This being the case, we will face a low signal-to-noise ratio unless we can identify which portion of the future sequence is relevant to and predictable from the earlier behaviors.

To tackle these challenges, we propose a novel sequence-to-sequence (seq2seq) training strategy in this paper. Our seq2seq training strategy is executed in parallel to the standard sequence-to-item (seq2item) training strategy, and complements the latter by further mining supervision signals from the whole future sequences. The proposed seq2seq strategy employs the ideas of latent self-supervision and intention disentanglement to address the aforementioned two challenges, respectively. Our first core idea is to perform self-supervision [1] in the latent space, rather than in the data space. In other words, our seq2seq training strategy asks the model to predict the representation of the future sub-sequence given the earlier sequence's representations. This design avoids individually reconstructing all the behaviors in the future sequence and eases convergence of the seq2seq training process. The representation to be predicted effectively serves as a distilled pseudo behavior (e.g., clicking a pseudo item) in the vector space, which summarizes the main intention present in the future sequence. Our second core idea is to design a sequence encoder that can infer and disentangle the latent intentions reflected by a given sequence of behaviors. The disentangled encoder outputs multiple representations of a given sequence of behaviors, where each representation focuses on a distinct sub-sequence of the given sequence. Each of the multiple representations characterizes the user's intention related to a different latent category. We can then construct seq2seq training samples using only pairs of sub-sequences whose intentions are relevant in that they involve the same latent category.

We conduct extensive experiments on both real-world benchmarks and synthetic data. The empirical results demonstrate that our seq2seq training strategy brings improvements over the baselines, by discovering additional supervision signals not covered by seq2item training.

We summarize our main contributions as follows:

- We propose a novel seq2seq training strategy, which extracts additional supervision signals by investigating the longer-term future instead of just the next immediate behavior.
- We suggest performing self-supervision in the latent space to boost convergence, and propose intention disentanglement to determine if two sub-sequences are relevant when selecting seq2seq training samples.
- We empirically demonstrate the efficacy of our seq2seq training strategy, which complements seq2item training.

2 RELATED WORK

2.1 Sequential Recommendation

Early recommender systems typically employ collaborative filtering [9, 22, 43, 45, 47], especially those based on matrix factorization [29, 46], for mining users' preference from their behavior. The advent of deep learning later further advances the field [8, 17, 32, 33, 52, 58]. Despite their success, these pioneering work typically neglect the sequential nature of a user's interactions with the recommender systems. As a result, some later recommenders propose to model the users' sequential behaviors as first-order or higher-order Markov chains [14–16, 44, 50, 53]. Motivated by the expressive power of deep sequential models, later sequential recommenders [7, 18, 19, 23, 25, 30, 35, 36, 49, 50, 56, 58] have achieved further success with more advanced deep models such as the recurrent networks, the convolutional neural networks, and the more recent self-attention based models such as Transformer [51] and BERT [10]. However, so far the deep sequential

[1] We use *self-supervision* to refer to the idea of training a model by asking it to predict one part of an instance given another part of the same instance, i.e., to predict a sub-sequence of behaviors given another sub-sequence by the same user in our case.

recommenders in the literature are mainly inspired by neural language models, and follow the setting of language generation where every item in the future sequence is explicitly reconstructed. Our work differs from these existing sequential recommender systems in that we perform sequence-to-sequence training in a space more abstract than the data space, i.e., the latent space.

2.2 Disentangling a User's Mixed Intentions

A disentangled representation characterizes the various explanatory factors behind an observed instance in different parts of the vector representation [1]. Many approach have been proposed to force the emergence of disentanglement in the learned representations. For example, some re-interpret the variational auto-encoders [28] from an information-theoretic perspective and derive various regularization terms that minimize the mutual information between a representation's different parts [3, 5, 20, 26]. Disentangling the factors behind an instance is sometimes also studied from the perspective of mixture data [2, 6, 11, 12, 24]. Lately, several representation learning-based algorithms are proposed for disentangling and preserving the multiple intentions behind the edges in relation data such as social networks and user-item interaction graphs [34, 37, 38]. We disentangle the intentions for a different purpose than these recent algorithms. Our purpose is to determine if there are shared intentions between the input sequence and the label sequence, and thus whether to use this pair for training.

2.3 Self-Supervision and Contrastive Learning

Self-supervised learning has gained increasing popularity for learning representations from unlabeled data via the pretext tasks. The pretext tasks dismantle complex objects into sub-parts and/or transform them into another near-equivalent form, based on which a prediction task or a discrimination task is conducted. The prediction tasks predict one part given another like a general Cloze test [10, 41, 57]. There are also works that predict the ordering relationship, trying to recover the original order or position of each part of a data example [39, 40]. The discrimination tasks usually adopt a contrastive learning paradigm which tries to discriminate the relationship of a paired example, e.g., whether a data example is transformed from the other [13, 21, 55], or whether two subparts are from the same object [31, 48]. Recently, some contrastive losses are proved capable of reducing the selection bias in training data, e.g., the exposure bias in recommender systems [59]. Among these works, contrastive predictive coding (CPC) [41] pioneers the practice of predicting the future part in the latent space for unsupervised pre-training, i.e., performing self-supervision in a latent space. However, CPC explores general settings where entanglement is much less severe and does not incorporate disentangled representation learning.

3 METHOD

3.1 Notations and Problem Formulation

Sequence data. Let $\{\mathbf{x}^{(u)}\}_{u=1}^{N}$ be the training data, i.e. the set of user click sequences. Here N is the number of users, while $\mathbf{x}^{(u)} = [x_1^{(u)}, x_2^{(u)}, \ldots, x_{T_u}^{(u)}]$ is an ordered sequence of items clicked by user u, where T_u is the number of clicks made by user u and $x_{T_u}^{(u)}$ is the user's latest click. Each element $x_t^{(u)} \in \{1, 2, \ldots, M\}$ $(1 \le t \le T_u)$ in the sequence is the index of the item being clicked. We focus on the candidate generation phase of a modern recommender system [8], where the task is to predict the next item(s) that user u is likely to click, among all the M possible options, based on the observed sequence $\mathbf{x}^{(u)}$.

Table 1: Notations.

Notation	Description
N	the number of sequences, aka. the number of users
M	the number of items
D	the dimensionality of the latent representations
K	the number of disentangled user intentions
$\mathbf{x}^{(u)}$	the sequence of items clicked by the u^{th} user
$x_t^{(u)}$	the t^{th} click in the u^{th} user's sequence $\mathbf{x}^{(u)} = [x_1^{(u)}, x_2^{(u)}, \ldots, x_{T_u}^{(u)}]$, where $x_t^{(u)} \in \{1, 2, \ldots, M\}$
T_u	the length of the u^{th} user's click sequence $\mathbf{x}^{(u)}$
$\boldsymbol{\theta}$	parameters of the sequence encoder
$\mathbf{H} \in \mathbb{R}^{M \times D}$	the item embedding table, included in $\boldsymbol{\theta}$
$\mathbf{H}_{i,:} \in \mathbb{R}^{D}$	the i^{th} item's representation, i.e. the i^{th} row of $\mathbf{H}$
$\mathbf{h}_t^{(u)} \in \mathbb{R}^{D}$	the representation of item $x_t^{(u)}$, i.e. row $x_t^{(u)}$ of $\mathbf{H}$
$\phi_{\boldsymbol{\theta}}(\cdot)$	the sequence encoder, which outputs K vectors
$\phi_{\boldsymbol{\theta}}^{(k)}(\mathbf{x}^{(u)})$	the representation of user u's intention under the k^{th} latent category, where $1 \le k \le K$
$\lambda \in [0, 1]$	the threshold for selecting sequence-to-sequence training samples of high confidence
$\mathcal{B}$	a mini-batch of sequences for training

Deep sequential recommenders. A deep sequential model for candidate generation typically has a sequence encoder $\phi_{\boldsymbol{\theta}}(\cdot)$ and an item embedding table $\mathbf{H} \in \mathbb{R}^{M \times D}$, where $\boldsymbol{\theta}$ is the set that contains all the trainable parameters including $\mathbf{H}$. The encoder takes a sequence $\mathbf{x}_u$ as input and outputs the representation of the sequence $\phi_{\boldsymbol{\theta}}(\mathbf{x}_u)$, which can be viewed as the representation of the user's intention(s). Most encoders $\phi_{\boldsymbol{\theta}}(\cdot)$ in the literature output a single D-dimensional vector, while there are also models that outputs K D-dimensional vectors to preserve the user's intentions under K latent categories. The model then estimates the probability that user u will click the i^{th} item by measuring the similarity between the user representation $\phi_{\boldsymbol{\theta}}(\mathbf{x}_u)$ and the i^{th} item's representation $\mathbf{H}_{i,:}$ in the vector space.

Sequence-to-item (seq2item) training. So far the most common practice for training a deep sequential recommender is to train the model to recover the next click $x_{t+1}^{(u)}$ based on the truncated sequence prior to the click, i.e. $[x_1^{(u)}, x_2^{(u)}, \ldots, x_t^{(u)}]$. For example, one commonly used training loss of this kind is

$$\mathcal{L}_{s2i}(\boldsymbol{\theta}) = \sum_u \sum_t \mathcal{L}_{s2i}(\boldsymbol{\theta}, u, t), \tag{1}$$

$$\mathcal{L}_{s2i}(\boldsymbol{\theta}, u, t) = -\ln p_{\boldsymbol{\theta}}(x_{t+1}^{(u)} \mid x_1^{(u)}, x_2^{(u)}, \ldots, x_t^{(u)}), \tag{2}$$

where the probability $p_{\theta}(x_{t+1}^{(u)} \mid \{x_i^{(u)}\}_{i=1}^{t})$ is designed to be proportional to the similarity between the item to be recovered and the given sequence in the vector space.

3.2 Sequence-to-Sequence Self-Supervision

In this subsection, we describe our seq2seq training strategy. The purpose of our seq2seq loss is to complement, not replace, the traditional seq2item loss. In other words, we minimize both the seq2item loss and the seq2seq loss when processing each mini-batch $\mathcal{B}$ using mini-batch gradient descent.

Each mini-batch $\mathcal{B}$ is a set of sampled sequence. We construct the mini-batch $\mathcal{B}$ by sampling each of its element from the training set $\{(u,t) : 1 \le u \le N, 1 \le t \le T_u - 1\}$ uniformly. Each training example (u,t) in the mini-batch $\mathcal{B}$ refers to an earlier sequence $\mathbf{x}_{1:t}^{(u)} = [x_1^{(u)}, x_2^{(u)}, \ldots, x_t^{(u)}]$ and its corresponding future sequence $\mathbf{x}_{t+1:T_u}^{(u)} = [x_{t+1}^{(u)}, x_{t+2}^{(u)}, \ldots, x_{T_u}^{(u)}]$. We further use $\mathbf{x}_{T_u:t+1}^{(u)}$ to represent the reversed sequence of $\mathbf{x}_{t+1:T_u}^{(u)}$, i.e. $\mathbf{x}_{T_u:t+1}^{(u)} = [x_{T_u}^{(u)}, x_{T_u-1}^{(u)}, \ldots, x_{t+1}^{(u)}]$.

We assume that we have a sequence encoder $\phi_{\theta}(\cdot)$, whose implementation will be presented in subsection 3.3. The output of the encoder are K vectors in D-dimensional space, i.e. $\phi_{\theta}(\cdot) = \{\phi_{\theta}^{(k)}(\cdot)\}_{k=1}^{K}$, which represents a user's preference under K different latent categories of items. We also assume that the value of $\phi_{\theta}^{(k)}(\mathbf{x}_{1:t}^{(u)})$ will be merely a white noise vector if the sequence $\mathbf{x}_{1:t}^{(u)}$ does not contain any items under the k^{th} latent category. In our implementation of the encoder, each output $\phi_{\theta}^{(k)}(\mathbf{x}_{1:t}^{(u)})$ of the K outputs pay attention to a different sub-sequence of the input sequence. We can view $\phi_{\theta}^{(k)}(\mathbf{x}_{1:t}^{(u)})$ as a pseudo item that summarizes the clicked items that are under the k^{th} latent category.

Sequence-to-sequence (seq2seq) loss. We define the sequence-to-sequence loss for this sample as

$$\mathcal{L}_{s2s}(\theta, u, t, k) = -\ln p_{\theta}(\phi_{\theta}^{(k)}(\mathbf{x}_{T_u:t+1}^{(u)}) \mid \phi_{\theta}^{(k)}(\mathbf{x}_{1:t}^{(u)})) = -\ln \frac{\exp\left(\frac{1}{\sqrt{D}}\, \phi_{\theta}^{(k)}(\mathbf{x}_{T_u:t+1}^{(u)}) \cdot \phi_{\theta}^{(k)}(\mathbf{x}_{1:t}^{(u)})\right)}{\sum_{(u',t')\in\mathcal{B}} \sum_{k'=1}^{K} \exp\left(\frac{1}{\sqrt{D}}\, \phi_{\theta}^{(k')}(\mathbf{x}_{T_{u'}:t'+1}^{(u')}) \cdot \phi_{\theta}^{(k)}(\mathbf{x}_{1:t}^{(u)})\right)}, \tag{3}$$

We have scaled the dot product scores by a factor of $\frac{1}{\sqrt{D}}$ because the last layer of our encoder is a layer-normalization layer and the scaling factor helps convergence. Here the softmax is normalized over the samples that appear in the current mini-batch [48] $\mathcal{B}$, instead of being normalized over all possible options that appear in the training set, so as to save computation. We use the reversed sequence $\mathbf{x}_{T_u:t+1}^{(u)}$ instead of the original $\mathbf{x}_{t+1:T_u}^{(u)}$, because our encoder weights the items in a sequence according to the time order and we would like the items closer to position t to gain more weights.

Select samples of high confidence for seq2seq training. However, we should only use a selected subset of $\{\mathcal{L}_{s2s}(\theta, u, t, k) : (u,t) \in \mathcal{B}, 1 \le k \le K\}$, rather than using all of them, for training. For example, if an earlier sequence $\mathbf{x}_{1:t}^{(u)}$ involves intention under latent category $k = 1$ and $k = 3$ while the future sequence $\mathbf{x}_{T_u:t+1}^{(u)}$ involves intention under category $k = 1$ and $k = 2$, then we should use only $\mathcal{L}_{s2s}(\theta, u, t, k = 1)$, but not use $k = 2$ and $k = 3$, when training the model. We therefore choose to compute the loss for the current mini-batch $\mathcal{B}$ based on some selected samples that are considered to be of high confidence according to the model. To be specific, the sequence-to-sequence loss is computed as

$$\mathcal{L}_{s2s}(\theta, \mathcal{B}) = \sum_{(u,t)\in\mathcal{B}} \sum_{k=1}^{K} \mathcal{L}_{s2s}(\theta, u, t, k) \cdot \mathbf{1}[\mathcal{L}_{s2s}(\theta, u, t, k) \le \tau] \tag{4}$$

where τ is the $\lceil \lambda \cdot |\mathcal{B}| \cdot K \rceil^{\text{th}}$ smallest value in $\{\mathcal{L}_{s2s}(\theta, u, t, k) : (u,t) \in \mathcal{B}, 1 \le k \le K\}$. Here $\lambda \in [0,1]$ is a hyper-parameter. In other words, we keep only the top λ-percent of the sequence-to-sequence training samples that are considered to be of high confidence by the model.

Sequence-to-item (seq2item) loss. The traditional sequence-to-item training strategy is necessary for learning a proper encoder in a relatively short time, as well as aligning the sequences' vector space and the items' vector space. Our sequence-to-item loss is defined as follows:

$$\mathcal{L}_{s2i}(\theta, \mathcal{B}) = \sum_{(u,t)\in\mathcal{B}} \mathcal{L}_{s2i}(\theta, u, t), \tag{5}$$

$$\mathcal{L}_{s2i}(\theta, u, t) = -\ln p_{\theta}(\mathbf{h}_{t+1}^{(u)} \mid \phi_{\theta}(\mathbf{x}_{1:t}^{(u)})) = -\ln \frac{\max_{k\in\{1,2,\ldots,K\}} \exp\left(\frac{1}{\sqrt{D}}\, \mathbf{h}_{t+1}^{(u)} \cdot \phi_{\theta}^{(k)}(\mathbf{x}_{1:t}^{(u)})\right)}{\sum_{(u',t')\in\mathcal{B}} \sum_{k'=1}^{K} \exp\left(\frac{1}{\sqrt{D}}\, \mathbf{h}_{t'+1}^{(u')} \cdot \phi_{\theta}^{(k')}(\mathbf{x}_{1:t}^{(u)})\right)}, \tag{6}$$

where $\mathbf{h}_{t+1}^{(u)} \in \mathbb{R}^D$ is the representation of item $x_{t+1}^{(u)}$, i.e. row $x_{t+1}^{(u)}$ of the item embedding table $\mathbf{H} \in \mathbb{R}^{M\times D}$.

We optimize the following loss when training our model using mini-batch gradient descent:

$$\mathcal{L}(\theta, \mathcal{B}) = \mathcal{L}_{s2i}(\theta, \mathcal{B}) + \mathcal{L}_{s2s}(\theta, \mathcal{B}). \tag{7}$$

3.3 Disentangled Sequence Encoding

The state-of-art sequence encoders for recommendation are those that based on the multi-head self-attention encoder, aka. the Transformer encoder [51]. For example, the SASRec encoder [25] is a recent variant of Transformer that uses a set of trainable position embeddings, instead of the original handcrafted position embeddings, to encode the order of the items in a sequence. Moreover, SASRec reuses the item embedding table $\mathbf{H}$, instead of building another item embedding table for the encoder, when encoding a sequence. In other words, its first layer takes $[\mathbf{h}_1^{(u)}, \mathbf{h}_2^{(u)}, \ldots, \mathbf{h}_t^{(u)}]$ as input when encoding the sequence $\mathbf{x}_{1:t}^{(u)} = [x_1^{(u)}, x_2^{(u)}, \ldots, x_t^{(u)}]$.

However, the SASRec encoder alone does not completely fulfill our requirements of the sequence encoder $\phi_{\theta}(\cdot)$. In particular, its ability at capturing multiple intentions is limited. The authors of SASRec report that the multi-head version of SASRec, which outputs multiple vector representations for the same input sequence, does not seem to have a clear advantage over the single-head implementation. Empirically both single-head SASRec and multi-head SASRec tend to recommend items of the same category as the latest one click in the input sequence, even if the user has clicked items of other categories earlier.

We therefore propose an intention-disentanglement layer here, which is appended after a single-head SASRec encoder so as to reuse

SASRec's expressive power. Let $[\mathbf{z}_1^{(u)}, \mathbf{z}_2^{(u)}, \ldots, \mathbf{z}_t^{(u)}]$, where $\mathbf{z}_i^{(u)} \in \mathbb{R}^D$, be the outputs of the single-head SASRec encoder at the t positions when given the input sequence $\mathbf{x}_{1:t}^{(u)} = [x_1^{(u)}, x_2^{(u)}, \ldots, x_t^{(u)}]$. We can view $\mathbf{z}_i^{(u)}$ as the latent intention of the user when the user is clicking item $x_i^{(u)}$, $i = 1, 2, \ldots, t$.

Intention clustering. Our intention-disentanglement layer starts by clustering the intentions according to their distance to a set of intention prototypes:

$$p_{k|i} = \frac{\exp\left(\frac{1}{\sqrt{D}}\text{LayerNorm}_1(\mathbf{z}_i^{(u)}) \cdot \text{LayerNorm}_2(\mathbf{c}_k)\right)}{\sum_{k'=1}^{K} \exp\left(\frac{1}{\sqrt{D}}\text{LayerNorm}_1(\mathbf{z}_i^{(u)}) \cdot \text{LayerNorm}_2(\mathbf{c}_{k'})\right)}, \tag{8}$$

where $i = 1, 2, \ldots, t$ and $k = 1, 2, \ldots, K$. Here $\{\mathbf{c}_k \in \mathbb{R}^D : 1 \le k \le K\}$ are the prototypical intention representations under the K latent categories, which are part of the model parameters $\boldsymbol{\theta}$. $\text{LayerNorm}_l(\cdot)$ is a layer-normalization layer, where we use the subscript l to avoid confusion of the different layer-normalization layers, since each has its own parameters for scaling its output. We are in effect using cosine, instead of dot product, due to the normalization, when measuring the similarity between a given intention $\mathbf{z}_i^{(u)}$ and a typical intention prototype $\mathbf{c}_{k'}$. Previous work [38] has found that cosine is much less vulnerable than dot product when it comes to mode collapse, i.e., the degenerate situation where most prototypes are being ignored by the model.

Intention weighting. The attention weight $p_{k|i}$ described above measures how likely the primary intention at position i is related with the k^{th} latent category. We now introduce another attention weight p_i to measure how likely the primary intention at position i is important for predicting the user's future intentions:

$$p_i = \frac{\exp\left(\frac{1}{\sqrt{D}}\text{key}_i \cdot \text{query}\right)}{\sum_{i'=1}^{t} \exp\left(\frac{1}{\sqrt{D}}\text{key}_{i'} \cdot \text{query}\right)}, \tag{9}$$

$$\text{key}_i = \widetilde{\text{key}}_i + \text{ReLU}(\mathbf{W}^\top \widetilde{\text{key}}_i + \mathbf{b}), \tag{10}$$

$$\widetilde{\text{key}}_i = \text{LayerNorm}_3(\boldsymbol{\alpha}_i + \mathbf{z}_i^{(u)}), \tag{11}$$

$$\text{query} = \text{LayerNorm}_4(\boldsymbol{\alpha}_t + \mathbf{z}_t^{(u)} + \mathbf{b}'), \tag{12}$$

where $i = 1, 2, \ldots, t$. Here $\mathbf{W} \in \mathbb{R}^{D \times D}$, $\mathbf{b} \in \mathbb{R}^D$, $\mathbf{b}' \in \mathbb{R}^D$, and $\boldsymbol{\alpha}_i \in \mathbb{R}^D$ are parameters and are included in $\boldsymbol{\theta}$. We can view $\{\boldsymbol{\alpha}_i : \boldsymbol{\alpha}_i \in \mathbb{R}^D, 1 \le i \le \max T_u\}$ as a set of position embeddings used by our intention-disentanglement layer. We compute the query based on $\boldsymbol{\alpha}_t$ and $\mathbf{z}_t^{(u)}$ as well as a trainable parameter $\mathbf{b}'$, based on the assumptions that: the more recent clicks are more valuable, and the earlier intentions that are close to the latest intention in the vector space are more likely to be important. Yet these two assumptions may not always be correct, which necessitates the introduction of the trainable parameters $\mathbf{W}$, $\mathbf{b}$, and $\mathbf{b}'$.

Intention aggregation. We can now aggregate the intentions collected at all the positions according to $p_{k|i}$ and p_i. The K outputs of the encoder are computed as follows:

$$\phi_{\boldsymbol{\theta}}^{(k)}(\mathbf{x}_{1:t}^{(u)}) = \text{LayerNorm}_5\left(\boldsymbol{\beta}_k + \sum_{i=1}^{t} p_{k|i} \cdot p_i \cdot \mathbf{z}_i^{(u)}\right), \tag{13}$$

where $k = 1, 2, \ldots, K$. Here $\boldsymbol{\beta}_k \in \mathbb{R}^D$ is the bias for output k, initialized as a sample from a normal distribution of mean 0 and standard deviation $\frac{1}{\sqrt{D}}$. We use two different sets of $\{\boldsymbol{\beta}_k\}_{k=1}^K$. One of the two sets is for encoding a sequence that serve as the *input* of a seq2seq sample, i.e., $\mathbf{x}_{1:t}^{(u)}$, while the other one is for encoding a sequence that serve as the *label*, i.e., $\mathbf{x}_{T_u:t+1}^{(u)}$.

To encourage disentanglement in the loss functions. In the literature, many regularization methods are proposed to encourage the K parts $\{\phi_{\boldsymbol{\theta}}^{(k)}(\cdot)\}_{k=1}^K$ to preserve sufficiently different information, e.g., regularization terms that minimize their mutual information. However, we note that in our loss functions, each positive case's score computed based on the k^{th} part $\phi_{\boldsymbol{\theta}}^{(k)}(\mathbf{x}_{1:t}^{(u)})$ (see the numerators in Eq. 3 and Eq. 6) are compared to those based on all the other $K-1$ parts $\{\phi_{\boldsymbol{\theta}}^{(k')}(\mathbf{x}_{1:t}^{(u)}) : 1 \le k' \le K, k' \ne k)\}$ (see the denominators in Eq. 3 and Eq. 6). As a result, the model is forced to preserve different information in $\phi_{\boldsymbol{\theta}}^{(k)}(\mathbf{x}_{1:t}^{(u)})$ than the other $K-1$ parts, if it wants to maximize the likelihood of the positive case related with part k. We therefore do not introduce any extra regularization term for disentanglement.

4 EXPERIMENTS

In this section, we evaluate [2] the performance of our approach in comparison with the state-of-art sequential recommenders and demonstrate the benefits of seq2seq training. We then conduct ablation study and analyze the impact of the hyper-parameters.

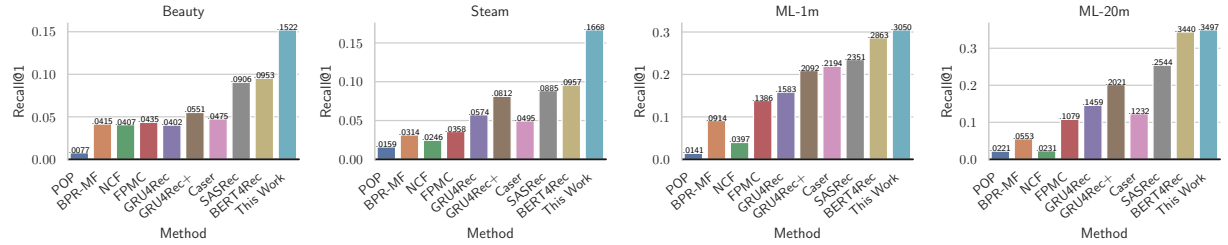
4.1 Experimental Setup

We follow the experimental setup described by BERT4Rec, a state-of-the-art sequential recommender based on BERT and Transformer.

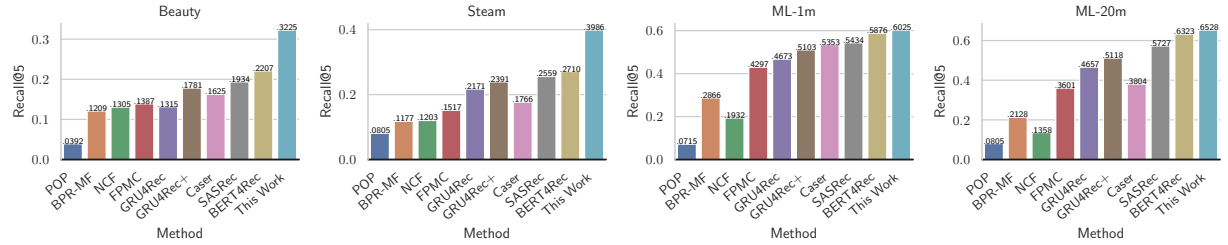
Datasets. We conduct our experiments on the datasets processed by SASRec [25] and BERT4Rec [49]. The four datasets are Amazon Beauty (40,226 users and 54,542 items), Steam (281,428 users and 13,044 items), MovieLens-1M (6,040 users and 3,416 items), and MovieLens-20M (138,493 users and 26,744 items), where the average sequence lengths are 8.8, 12.4, 163.5, and 144.4, respectively. The items in a sequence are ordered by time, where the last position corresponds to the latest click. We split the datasets in the same way as the previous work [25, 49], i.e., the last item of each user's sequence for testing, the second-to-last for validation, and the remaining items for training.

Evaluation metrics. We evaluate all the methods in terms of recall, normalized discounted cumulative gain (NDCG), and mean reciprocal rank (MRR). Higher value in all these metrics reflect better recommendation performance. We follow BERT4Rec's advice, and pair each ground-truth item in the test set with 100 negative items randomly sampled according to their popularity, which is a common practice in the literature [49]. The recommendation task then becomes to identify which item among these 101 items is the ground-truth next item for each user.

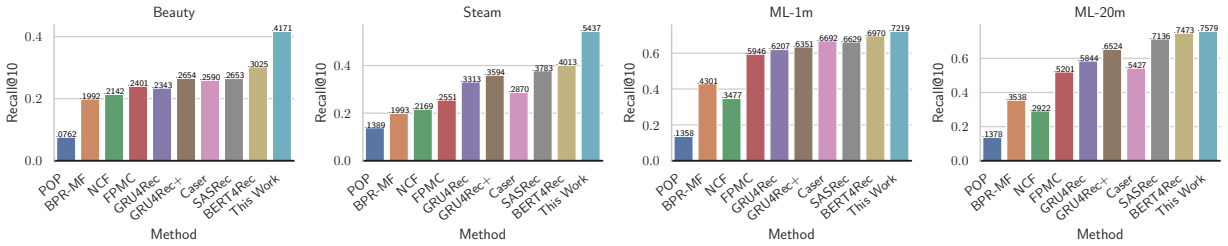
[2] As pointed out by two reviewers, our empirical conclusions are not as reliable as they seem, due to two flaws in our experimental setup. First, most of the evaluation metrics are not reliable if sampling is used, except for the AUC, especially when the sample size is small [42]. Second, instead of using a time based split, we use the latest events of the users as test examples, which is vulnerable to information leak.



(a) Recall items in the top 1 position (Recall@1).



(b) Recall items in the top 5 positions (Recall@5).



(c) Recall items in the top 10 positions (Recall@10).

Figure 2: Recommendation performance in terms of RecallG@1, Recall@10, and Recall@10. These metrics measure how well a method can retrieve the relevant items with a limited budget.

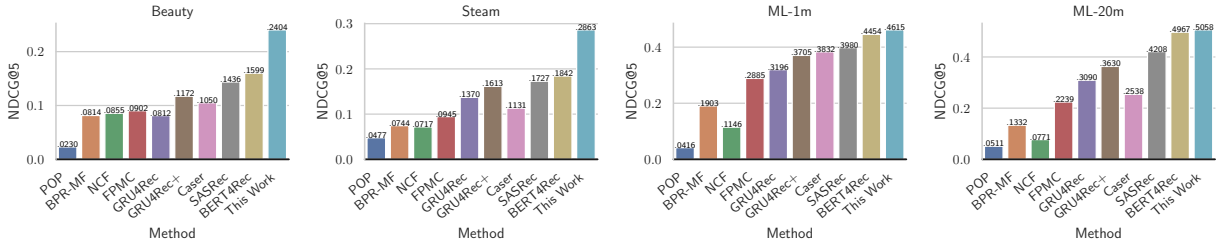
Implementation and hyper-parameters. We implement our model in TensorFlow and initialize the parameters using the default initialization recommended by TensorFlow. We use the Adam [27] optimizer for mini-batch gradient descent and set the learning rate to 0.001, while the size of each mini-batch is 128. We use the single-head implementation of SASRec as part of our encoder. We cap the maximum sequence length to 200 for MovieLens-1M and MovieLens-20M, while capping it to 50 for the other two datasets, which is the same configuration used by SASRec and BERT4Rec. The other hyper-parameters are then tuned using random search. Specifically, we follow BERT4Rec and choose the dimensionality of the item embeddings, D, from $\{16, 32, 64, 128, 256\}$. The number of self-attention blocks, which are used by the part of our encoder borrowed from SASRec, is chosen from $\{1, 2, 3\}$. The hyper-parameter λ is from $\{0.05, 0.10, \ldots, 1.0\}$. The number of latent categories, K, is chosen from $\{1, 2, \ldots, 8\}$. The dropout rate is chosen from $\{0, 0.1, 0.2, \ldots, 0.9\}$, while the l_2 regularization term is selected from $\{0, 0.0001, 0.001, \ldots, 1\}$.

4.2 Recommendation Performance

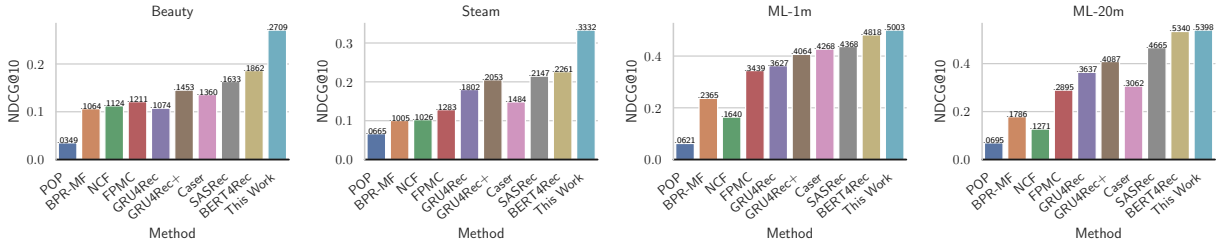
In this subsection, we report the overall performance of our approach compared to the state-of-art sequential recommenders.

Baselines. We compare our approach with a series of representative baselines. We include a naïve baseline that recommends the most popular items (POP), the matrix factorization variant of the classic Bayesian personalized ranking algorithm (BPR-MF) [43], as well as the well-known neural collaborative filtering (NCF) [17]. We then consider strong baselines that leverages the sequential nature of the user behavior data, including the factorized personalized Markov chains (FPMC) [44] that models a sequence as a Markov chain, the recurrent neural network-based GRU4Rec [19] as well as its improved version GRU4Rec+ [18], the convolutional neural network-based Caser [50], the Transformer-based SASRec [25], and the state-of-the-art deep sequential recommender BERT4Rec [49] that trains a bi-directional Transformer encoder using BERT's Cloze objective.

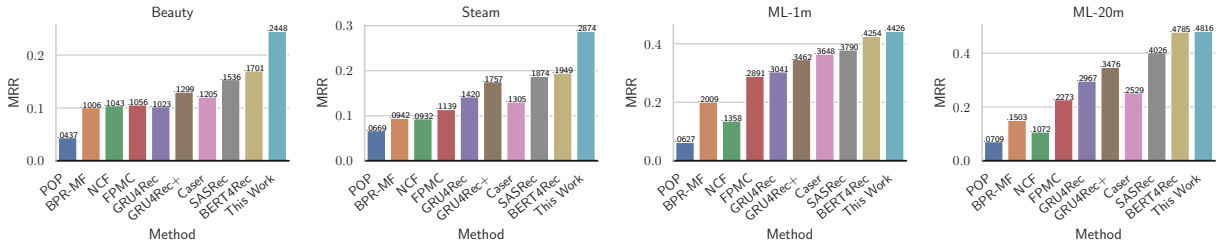
Analysis. Figure 2 and Figure 3 presents the overall recommendation performance of all methods on the four datasets. We can



(a) Normalized discounted cumulative gain, truncated at rank 5 (NDCG@5).



(b) Normalized discounted cumulative gain, truncated at rank 10 (NDCG@10).



(c) Mean reciprocal rank (MRR).

Figure 3: Recommendation performance in terms of NDCG@5, NDCG@10, and MRR. These metrics measure how well a method can rank the relevant items before the irrelevant ones.

see that our approach, which combines the traditional seq2item training strategy with disentangled latent seq2seq training, consistently outperforms all the baselines. The improvement is especially impressive on Beauty and Steam, where the relative improvement over the strongest baselines is in general over 35%. However, we also notice that the performance gains on the other two datasets, MovieLens-1M and MovieLens-20M, are less impressive, where the relative improvement over the strongest baselines is around 5%. This may due to the fact that these two latter datasets contain much longer sequences, where the average lengths are 163.5 and 144.4, respectively, whereas Beauty and Steam contain much shorter sequences of average length 8.8 and 12.4, respectively. Such long sequences, of length over 140, can be particularly challenging to disentangle.

4.3 Robustness to Synthetic Noises

We now analyze how robust our seq2seq training strategy are compared to the traditional strategy that only uses seq2item training. Specifically, we corrupt the training data by randomly replacing a portion of the observed clicks in the training set with uniformly sampled items. We conduct this experiment on Beauty, and range the percentage of the corrupted training data from 10% to 50%.

We show in Figure 4 two variants of our method. One optimizes both the seq2item loss and the seq2seq loss as we have described in Section 3, while the other one optimizes only the seq2item loss. We can see that the recommendation performance drops slower if the seq2seq training strategy is in use, as long as the corruption level remains relatively modest (e.g., $< 20\%$ noises). This indicates that our seq2seq training strategy, by mining additional supervision signals from the longer-term future and selectively learning from seq2seq samples of high confidence, does have the potential to bring improved robustness.

4.4 Ablation Study

Table 2 lists the results of the ablation study. Variant 1 of our method removes the seq2seq loss and use only the seq2item loss. We observe a drop in performance, demonstrating the efficacy of our seq2seq loss. Variant 2 and 3 directly reconstruct every items, i.e., optimize the seq2item loss for every items in the future sequence, instead of using our seq2seq loss. Variant 2 and 3 perform even worse than

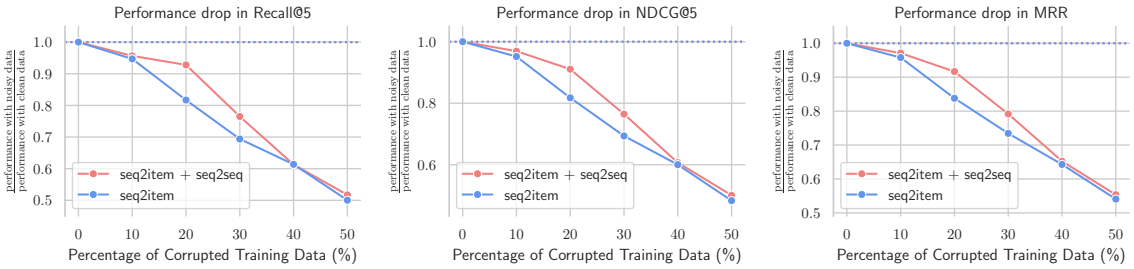


Figure 4: Relative performance drop on dataset Beauty when the training data are corrupted by synthetic noises. The y-axis is the ratio of the performance with noisy training data to the performance with clean training data.

Table 2: Ablation study on dataset Beauty.

	Evaluation Metrics					
Variants of Our Method	Recall@1	Recall@5	Recall@10	NDCG@5	NDCG@10	MRR
(1) Remove seq2seq training	0.1358	0.3002	0.3891	0.2369	0.2675	0.2420
(2) Individually reconstruct all items in a future sequence	0.1071	0.2709	0.3744	0.1916	0.2251	0.1992
(3) Individually reconstruct the next three items	0.1202	0.2914	0.3898	0.2084	0.2403	0.2139
(0) Default	**0.1522**	**0.3225**	**0.4171**	**0.2404**	**0.2709**	**0.2448**

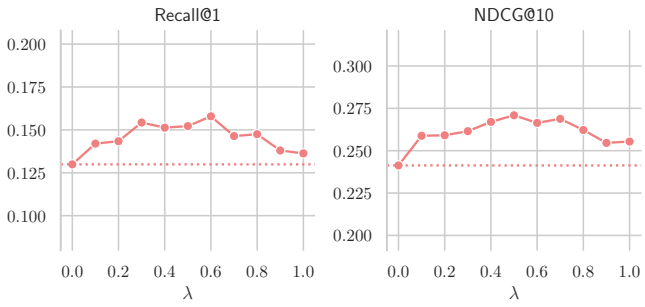


Figure 5: Impact of the threshold hyper-parameter $\lambda \in [0, 1]$, which is for determining whether a seq2seq sample is of high confidence and thus whether to use the sample for self-supervised training. $\lambda = 0$ is equivalent to not using seq2seq training, while $\lambda = 1$ selects all seq2seq samples for training.

variant 1 which considers only one future item. The degradation is likely due to the many irrelevant items in the long-term future.

4.5 Hyper-parameter Sensitivity

Our seq2seq loss involves a critical hyper-parameter $\lambda \in [0, 1]$, which is the threshold for determining if a seq2seq sample in the training set is of high confidence and therefore whether to use this seq2seq sample for training. We conduct experiments on Beauty and illustrate in Figure 5 the impact of this hyper-parameter.

Figure 5 shows that the choice of λ does matter. A threshold too strict will limit the number of seq2seq samples being used, while a threshold too loose will introduce too many irrelevant seq2seq samples. Results on the other datasets follow a similar trend, even though the optimal value may vary between datasets.

5 CONCLUSION AND FUTURE WORK

We have proposed a novel sequence-to-sequence training strategy, which leverages additional supervision signals from the longer-termed future by performing and self-supervised learning in the latent space and disentangling users' intentions. Empirically we demonstrate the additional gains brought by the extra signals.

Future directions include reducing seq2seq training's computational cost via an engineering-efficient framework [59], as well as improving its performance on long sequences.

ACKNOWLEDGMENTS

This work is supported by the Natioanl Key R&D Program of China under Grand No. 2018AAA0102001 and the National Natural Science Foundation of China Major Project (No. U1611461).

REFERENCES

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.
[2] Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. 2018. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[3] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. 2018. Understanding disentangling in $beta$-VAE. *arXiv preprint arXiv:1804.03599* (2018).
[4] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 456–464.
[5] Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. 2018. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*. 2610–2620.

[6] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS 2016*.

[7] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *Proceedings of WSDM 2018*.

[8] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

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

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[11] Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. 2016. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648* (2016).

[12] Emilien Dupont. 2018. Learning disentangled joint continuous and discrete representations. In *Advances in Neural Information Processing Systems*. 710–720.

[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2019. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722* (2019).

[14] Ruining He, Chen Fang, Zhaowen Wang, and Julian McAuley. 2016. Vista: a visually, socially, and temporally-aware model for artistic recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 309–316.

[15] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Proceedings of ACM RecSys 2017*.

[16] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 191–200.

[17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of WWW 2017*.

[18] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 843–852.

[19] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[20] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, Vol. 3.

[21] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670* (2018).

[22] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 263–272.

[23] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR 2018*.

[24] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. 2017. Variational deep embedding: an unsupervised and generative approach to clustering. In *Proceedings of IJCAI 2017*.

[25] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM 2018*.

[26] Hyunjik Kim and Andriy Mnih. 2018. Disentangling by Factorising. In *International Conference on Machine Learning*. 2654–2663.

[27] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*.

[28] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[29] Yehuda Koren, Robert Bell, Chris Volinsky, et al. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[30] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[31] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557* (2019).

[32] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proceedings of SIGKDD 2017*.

[33] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of WWW 2018*.

[34] Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu. 2019. Is a Single Vector Enough? Exploring Node Polysemy for Network Embedding. In *Proceedings of SIGKDD 2019*.

[35] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-aware sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1053–1058.

[36] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1831–1839.

[37] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*.

[38] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning disentangled representations for recommendation. In *Advances in Neural Information Processing Systems*. 5712–5723.

[39] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. 2016. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*. Springer, 527–544.

[40] Mehdi Noroozi and Paolo Favaro. 2016. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*. Springer, 69–84.

[41] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[42] Steffen Rendle. 2019. Evaluation Metrics for Item Recommendation under Sampling. arXiv:arXiv:1912.02263

[43] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*.

[44] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[45] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM, 175–186.

[46] Ruslan Salakhutdinov and Andriy Mnih. 2011. Probabilistic matrix factorization. In *NIPS*, Vol. 20. 1–8.

[47] 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*. ACM, 285–295.

[48] Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*. 1857–1865.

[49] 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 CIKM 2019*.

[50] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[52] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1235–1244.

[53] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of SIGIR 2015*.

[54] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*. 495–503.

[55] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3733–3742.

[56] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *Proceedings of SIGIR 2016*.

[57] Richard Zhang, Phillip Isola, and Alexei A Efros. 2016. Colorful image colorization. In *European conference on computer vision*. Springer, 649–666.

[58] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. ATRank: An attention-based user behavior modeling framework for recommendation. In *AAAI 2018*.

[59] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. 2020. Contrastive Learning for Debiased Candidate Generation in Large-Scale Recommender Systems. arXiv:arXiv:2005.12964