Self-guided Contrastive Learning for Sequential Recommendation

Hui Shi[1], Hanwen Du[1], Yongjing Hao[1], Victor S. Sheng[2], Zhiming Cui[3], and Pengpeng Zhao[1(✉)]

[1] School of Computer Science and Technology, Soochow University, Suzhou, China
{hshi1,hwdu,yjhaozb}@stu.suda.edu.cn, ppzhao@suda.edu.cn

[2] Department of Computer Science, Texas Tech University, Lubbock, USA
Victor.Sheng@ttu.edu

[3] School of Electronic and Information Engineering, Suzhou University of Science and Technology, Suzhou, China
zmcui@mail.usts.edu.cn

Abstract. Sequential recommendation has injected plenty of vitality into online marketing and retail industry. Existing contrastive learning-based models usually resolve data sparsity issue of sequential recommendation with data augmentations. However, the semantic structure of sequences is typically corrupted by data augmentations, resulting in low-quality views. To tackle this issue, we propose **Self-**guided contrastive learning enhanced **BERT** for sequential recommendation (**Self-BERT**). We devise a self-guided mechanism to conduct contrastive learning under the guidance of BERT encoder itself. We utilize two identically initialized BERT encoders as view generators to pass bi-directional messages. One of the BERT encoders is parameter-fixed, and we use the all Transformer layers' output as a series of views. We employ these views to guide the training of the other trainable BERT encoder. Moreover, we modify the contrastive learning objective function to accommodate one-to-many positive views constraints. Experiments on four real-world datasets demonstrate the effectiveness and robustness of **Self-BERT**.

Keywords: Sequential recommendation · Contrastive learning · BERT

1 Introduction

Recommender systems, owing to their ability to give suggestions for users effectively, can alleviate the information overload issue and help users derive valuable insights from big data. Sequential Recommendation aims to model user behaviors dynamically by taking the sequential patterns of user interaction history into consideration [9,11,16–18]. Given recent observations of users, sequential recommendation models are built to capture sequential item relationships.

Due to the high practical value of sequential recommendation, various works are proposed to resolve it. Early works [16] on sequential recommendation are based on the Markov Chain (MC) assumption to model users' pair-wise behavior transition. Due to the limitation of MC-based models, Recurrent Neural Network (RNN) [3] is adopted in sequential recommendation to model sequence-wise relationships. Recently, Transformer [19] adopts the self-attention mechanism to encode sequences, which has proved significantly powerful in various fields [2,20,21]. Thus, many researchers design Transformer-based models in sequential recommendation [11,17]. For example, BERT4Rec [17] employs Bi-directional Encoder Representations from Transformers (BERT) [4], stacked by multiple Transformer encoder layers, to reveal the correlations of sequential items.

Although existing approaches have achieved great recommendation results, the issue of data sparsity is still not well explored [13,22], leading to suboptimal performance. In detail, the data sparsity problem includes inadequate data amount and short data length, causing insufficient model training. More recently, a contrastive learning paradigm is introduced to alleviate the above issue [13,22]. In detail, this paradigm usually constructs both positive and negative views of an original sequence through data augmentations. The goal is to push positive views close to the original sample by optimizing the value of contrastive learning loss, while negative views are the opposite. Such a paradigm can enhance the discrimination ability of the encoders and improve the robustness of the model.

However, we consider that there are two points in existing contrastive learning models that could be improved. First, most contrastive learning sequential recommendation models are modified based on unidirectional Transformers. In fact, bi-directional Transformers perform better than unidirectional Transformers, which means we can consider integrate bi-directional Transformers (such as BERT) with contrastive learning to obtain better recommendation performance. Second, the data augmentations adopted in the existing contrastive learning-based models have two shortcomings, i.e., (1) finding and realizing the optimal data augmentation method for different datasets is very time-consuming, (2) the data augmentation process of generating views has a certain degree of randomness, leading to the destruction of important original semantic information. Therefore, instead of data augmentation-based solutions, a more efficient and stable contrastive learning scheme is highly demanded.

To this end, we propose a new framework, **Self-**guided contrastive learning enhanced **BERT** for sequential recommendation (**Self-BERT**) to cope with the above issues. It consists of three essential parts: (1) a traditional BERT-based sequential recommendation task; (2) a self-guided contrastive learning paradigm to take advantage of all the information captured in BERT encoder; (3) a joint-learning training framework to optimize two loss functions simultaneously in contrastive learning and recommendation tasks. Specifically, Self-BERT uses two BERT encoders. One BERT encoder is fixed after initialization and its outputs of all hidden layers are regarded as positive views. The other is used to obtain general sequence representations. In this way, multiple pair of positive views are automatically generated. It is equivalent to using the information inside the BERT encoder to guide its training, and we call it a self-guided mechanism. In addition, we modify NT-Xent loss [1] by extending its one-to-one view constraints into one-to-many view constraints, which allows our model to train multiple pairs of positive and negative views simultaneously. We conduct experiments on four real-world datasets to verify the effectiveness and robustness of our **Self-BERT**. To distinguish our work from other sequential recommendation solutions, main contributions of this paper are listed as follows:

- To the best of our knowledge, this is the first work to apply self-guided contrastive learning-based BERT to sequential recommendation.
- We propose a novel data augmentation-free contrastive learning paradigm to tackle the unstable and time-consuming challenges in contrastive learning. It exploits self-guided BERT encoders and extends one-to-many view constraints to preserve the view-wise semantic correlations.
- We conduct extensive experiments on four real-world benchmark datasets. Our experimental results improve competitive baselines with a large margin under the challenges of data sparsity.

2 Related Work

2.1 Sequential Recommendation

Sequential recommendation predicts future items that users may be interested in by capturing item correlations in history interaction sequences. Early proposed models, such as FPMC [16], adopt the MC assumption to capture pairwise item correlations. Besides, FPMC incorporates Matrix Factorization (MF) to simulate users' personal preferences. With the advancement of neural networks in many other research domains [27,28], RNN [3] and its variants are widely used in sequential recommendation. For example, Hidasi et al. [9] propose GRU4Rec, which exploits Gated Recurrent Unit (GRU) [5], and it aims to dynamically model long-term users' preferences from their historical interaction sequences. In addition, other deep neural networks, like Convolutional Neural Network (CNN), also made remarkable achievements in sequential recommendation. Tang et al. design a CNN-based model named Caser [18], which views history interaction sequences as "images" and captures local patterns via convolutional filters. Yuan et al. [25] devise a generative model (NextItNet) by stacking holed convolutional layers to increase the receptive fields of convolutional filters. Recently, inspired by the application of self-attention network [19] in Natural Language Processing (NLP), various attention-based models are proposed in sequential recommendation [23,26]. Kang et al. [11] develop SASRec to characterize advanced item transition correlations by adapting Transformer layer [19]. Sun et al. [17] propose BERT4Rec by applying BERT [4] to obtain better sequence representations. BERT4Rec essentially consists of a stack of Transformers that can fuse contextual information from two directions. Nowadays, multiple contrastive learning-based models are proposed, trying to obtain characteristic signals within the sequence. Both S^3-Rec [13] and CL4SRec [22] are very competitive models. However, the encoders in these models only consider unidirectional information transfer without the ability to fuse contextual information in the sequence.

2.2 Self-supervised Learning

The goal of Self-Supervised Learning (SSL) is to utilize the unlabeled data to obtain valuable signals for downstream tasks. Generally speaking, SSL models can mainly be divided into generative and contrastive approaches [10]. Representative models of generative SSL-based models includes Generative Adversarial Networks (GAN) [7] and Variational Auto Encoders (VAE) [12]. These models are designed to make the generated data close to the original data. Different from generative models, contrastive SSL-based models are built to maximize the consistency between pairs of positive views and the opposite for pairs of negative views. Positive views are obtained by performing a series of augmentation operations on original sequences. Recently, Contrastive SSL-based methods have achieved remarkable success in various research domains [6,8,29]. For different data types, views are generated by applying reasonable augmentation methods. Chen et al. [1] propose SimCLR and employ random crop, color distortions, and Gaussian blur to obtain augmented visual representations. GraphCL [24] devises four data augmentations to generate views, including node dropping, edge perturbation, attribute masking, and subgraph selection. Existing contrastive SSL-based sequential recommendation models fully consider the sequence characteristics of user interaction history and design effective view generators. Zhou et al. [13] propose S^3-Rec to learn the item transition correlations among attribute, item, subsequence and sequence by optimizing four auxiliary self-supervised objectives. CL4SRec [22] devises three data augmentation operations (crop, mask, reorder) to randomly generate views of original sequences. However, we hold the opinion that the contrastive learning views generated by data augmentations cannot preserve the complete semantic information of the original sequences.

3 Method: Self-BERT

In this section, we first provide the problem definition of sequential recommendation. Then, we introduce the two tasks of Self-BERT, the main recommendation task and the auxiliary self-guided contrastive learning task. Finally, we train the above two tasks together through a joint learning framework.

3.1 Problem Definition

In this paper, we symbolize user and item sets as $\mathcal{U}$ and $\mathcal{V}$, respectively. $|\mathcal{U}|$ and $|\mathcal{V}|$ represent the number of users and items. Each user has a chronological interaction sequence consisting of a series of items. For user u, we denote the corresponding sequence as $S_u = [v_1, v_2, ..., v_{|S_u|}]$, where v_t represents user u interacted with this item at time t. Given the interaction sequence S_u of user u, the target of sequential recommendation is to predict the most possible item that user u will interact with at time $|S_u|+1$, which can be formulated as follows:

$$\arg\max_{v_i \in \mathcal{V}} P(v_{|S_u|+1} = v_i | S_u). \tag{1}$$

3.2 Recommendation Task

In this subsection, we introduce the recommendation task implemented by a BERT encoder. As shown in Fig. 1, the BERT encoder consists of stacked Transformer encoder layers. Unlike the left-to-right unidirectional Transformer, the BERT encoder can fuse bi-directional contextual information, which gives the model a global receptive field to capture the dependencies in any distance.

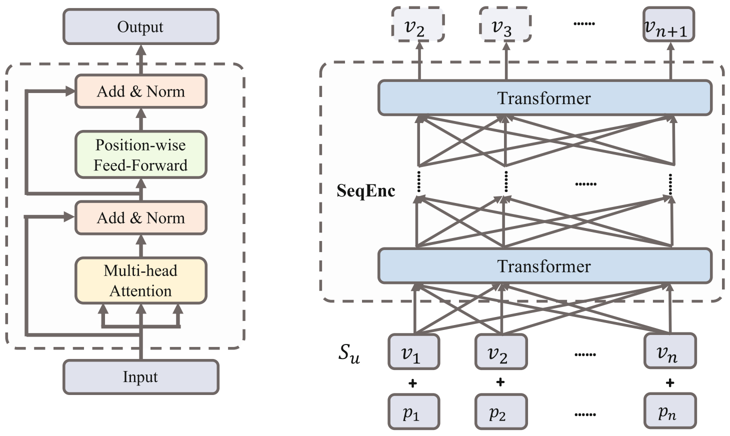


Fig. 1. The architecture of Transformer encoder (left) and BERT (right).

Embedding Layer. To take full advantage of the sequential information of the input, for a given item v_i of sequence S_i, its embedding representation $\boldsymbol{h}_i^0$ considers both original item embedding $\boldsymbol{v}_i$ and corresponding positional embedding $\boldsymbol{p}_i$. The formula is as follows:

$$\boldsymbol{h}_i^0 = \boldsymbol{v}_i + \boldsymbol{p}_i, \tag{2}$$

where $\boldsymbol{h}_i^0$, $\boldsymbol{v}_i$, $\boldsymbol{p}_i$ are all d-dimensional embedding vectors (d represents the dimension of embedding). By this way, a positional embedding representation $\boldsymbol{H}^0 \in \mathbb{R}^{N \times d}$ of sequence S_i is obtained, where N denotes the maximum length of all sequences[1].

Transformer Layer. Given a sequence of length N, after passing through the embedding layer, we iteratively pass it through L layers of Transformers to get the hidden layer representation $\boldsymbol{h}_i^l$ of each layer l at position i. We stack the hidden layer representations of all positions together to get $\boldsymbol{H}^l \in \mathbb{R}^{N \times d}$. Next, we pass $\boldsymbol{H}^l$ through a multi-head self-attention network to capture the dependencies

[1] We pad the sequence length with zeros on the left when the length is less than N.

between representation pairs at arbitrary distances in the sequence. We linearly projects $\boldsymbol{H}^l$ into h subspaces, and the formula is as follows:

$$\begin{aligned} \mathrm{MH}(\boldsymbol{H}^l) &= \mathrm{concat}(\mathrm{head}_1; \mathrm{head}_2; \cdots; \mathrm{head}_h)\boldsymbol{W}^O, \\ \mathrm{head}_i &= \mathrm{Attention}(\boldsymbol{H}^l \boldsymbol{W}_i^Q, \boldsymbol{H}^l \boldsymbol{W}_i^K, \boldsymbol{H}^l \boldsymbol{W}_i^V), \end{aligned} \tag{3}$$

where $\boldsymbol{W}^O$, $\boldsymbol{W}_i^Q$, $\boldsymbol{W}_i^K$, and $\boldsymbol{W}_i^V$ are trainable parameters. $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ represent queries, keys, and values, respectively. The Attention function is implemented by the Scaled Dot-Product Attention:

$$\mathrm{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top / \sqrt{d/h})\boldsymbol{V}, \tag{4}$$

After linear projection through multi-head self-attention, we utilize a position-wise feed-forward network (PFFN) to endow the interactions between different dimensions at each position $\boldsymbol{h}_i$:

$$\mathrm{PFFN}(\boldsymbol{H}^l) = [\mathrm{FFN}(\boldsymbol{h}_1^l)^\mathsf{T}; ...; \mathrm{FFN}(\boldsymbol{h}_N^l)^\mathsf{T}]^\mathsf{T}, \tag{5}$$

where FFN(·) represents a two-layer feedforward network with GELU as an activation function. We also use residual connections, dropout, and normalization to connect the two sub-layers of multi-head self-attention and position-wise feed-forward network. The process of stacking Transformer layers is as follows:

$$\begin{aligned} \boldsymbol{H}^l = \mathrm{Trm}(\boldsymbol{H}^{l-1}) &= \mathrm{LayerNorm}(\boldsymbol{F}^{l-1} + \mathrm{Dropout}(\mathrm{PFFN}(\boldsymbol{F}^{l-1}))), \\ \boldsymbol{F}^{l-1} &= \mathrm{LayerNorm}(\boldsymbol{H}^{l-1} + \mathrm{Dropout}(\mathrm{MH}(\boldsymbol{H}^{l-1})). \end{aligned} \tag{6}$$

Finally, we pass $\boldsymbol{H}^L$ through a linear prediction layer and a softmax operation to get an output distribution $P(v)$ over all candidate items.

Training with the Cloze Task. In order to realize the bi-directional delivery of information in the Transformer encoder, the Cloze task is selected to train the model. The input sequence is randomly masked according to the ratio ρ during the training process, and the masked items are predicted based on the two-way context information. The objective function of the recommendation task is:

$$\mathcal{L}_{Rec} = \frac{1}{|S_{u,m}|} \sum_{v_m \in S_{u,m}} -\log P(v_m = v_m^* | S_u^{'}), \tag{7}$$

where $S_u^{'}$ represents the masked user behavior sequence S_u. $S_{u,m}$ is a collection of random masked items. v_m and v_m^* represent masked item and true item, respectively. In the testing process, we append a "[mask]" token at the end of the input sequence to predict the next item.

3.3 Self-guided Contrastive Learning Task

Unlike existing models that construct views through data augmentation, we propose Self-BERT and use the hidden layer representations obtained by the

BERT encoder itself as positive views. The main architecture of Self-BERT is shown in Fig. 2. Essentially, these representations are all yielded from the same input sequence and thus can be treated as positive views. We call it self-guided contrastive learning since the encoder uses its internal signals to guide the contrastive learning.

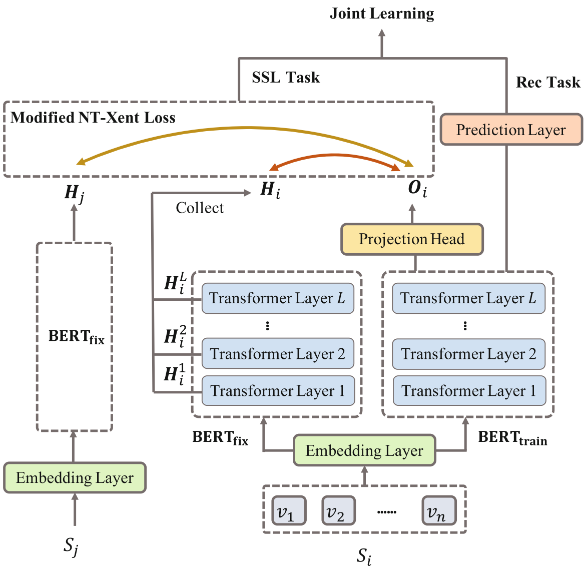


Fig. 2. The overall framework of Self-BERT. The positive views H_i (a series of hidden layer outputs) and O_i are obtained through the BERT$_{\text{fix}}$ and BERT$_{\text{train}}$ encoders, respectively. The negative view H_j is obtained through the BERT$_{\text{fix}}$. And we calculate the contrastive learning loss through this ternary relationship.

Specifically, we first pre-train the BERT encoder for z epochs[2] on the Cloze task. Then, we use two BERT encoders to obtain positive views in the contrastive learning task. One of the encoders, called BERT$_{\text{train}}$, is initialized according to the parameters in the trained BERT, and the parameters are updated synchronously in the subsequent training process. The other encoder, called BERT$_{\text{fix}}$, is also a copy of the trained BERT, but its parameters are fixed after initialization. Self-BERT uses all hidden layer outputs obtained by BERT$_{\text{fix}}$ as positive views for contrastive learning. Therefore, the information captured by each Transformer layer in the BERT encoder can be fully utilized.

Given an interaction sequence S_i of user i, we first pass it through the same embedding layer as in the recommendation task to get the embedded input $\boldsymbol{S}_i^E$ of the encoder. A contrastive learning view $\boldsymbol{O}_i$ is obtained directly by passing

[2] We found in experiments that the performance of Self-BERT does not change much with different z, and we choose $z = 50$.

$\boldsymbol{S}_i^E$ through $\mathrm{BERT}_{\mathrm{train}}$ and a projection head. At the same time, we feed $\boldsymbol{S}_i^E$ into $\mathrm{BERT}_{\mathrm{fix}}$ and obtain the output of each layer of Transformers:

$$\boldsymbol{H}_i = \mathrm{BERT}_{\mathrm{fix}}(\boldsymbol{S}_i^E) = [\boldsymbol{H}_i^1, ..., \boldsymbol{H}_i^k, ..., \boldsymbol{H}_i^L] \quad (1 \leq k \leq L). \tag{8}$$

Since each Transformer layer in the BERT encoder can capture different level information, we regard $\boldsymbol{O}_i$ and each element in $\boldsymbol{H}_i$ as a pair of positive views. Thus, there are L pairs of positive views in total.

As for negative views, we directly choose other sequences in the batch where S_i is located. For example, Fig. 2 shows that we feed the negative sample S_j into $\mathrm{BERT}_{\mathrm{fix}}$ and acquire a series of negative views $\boldsymbol{H}_j$.

After computing these vectors, we compute the NT-Xent loss [1] for each positive sample, which is commonly used in contrastive learning.

$$\phi(u, v) = \exp(\mathrm{sim}(f(u), f(v))/\tau), \tag{9}$$

$$\mathcal{L}_{Cl} = -\log \frac{\phi(\boldsymbol{O}_i, \boldsymbol{H}_i^L)}{\phi(\boldsymbol{O}_i, \boldsymbol{H}_i^L) + \Sigma_{j=1, j\neq i}^{b} \phi(\boldsymbol{O}_i, \boldsymbol{H}_j^L)}, \tag{10}$$

where b denotes the current batch size, τ denotes the temperature parameter, $\mathrm{sim}(\cdot)$ and $f(\cdot)$ symbolize the cosine similarity function and a projection head, $\boldsymbol{H}_i^L$ and $\boldsymbol{H}_j^L$ represent the last layer output of $\mathrm{BERT}_{\mathrm{fix}}$.

However, as shown in Eq. (10), the commonly used NT-Xent loss only computes a pair of positive and negative views. Therefore, we modify Eq. (10) so that it could take into account the output of intermediate layers and calculate the loss of L pairs of positive and negative views simultaneously:

$$\mathcal{L}_{Cl} = \frac{1}{L} \sum_{k=1}^{L} -\log \frac{\phi(\boldsymbol{O}_i, \boldsymbol{H}_i^k)}{\phi(\boldsymbol{O}_i, \boldsymbol{H}_i^k) + \Sigma_{j=1, j\neq i}^{b} \Sigma_{l=1}^{L} \phi(\boldsymbol{O}_i, \boldsymbol{H}_j^l)}. \tag{11}$$

3.4 Joint Learning

Finally, we train the recommendation task and the self-guided contrastive learning task simultaneously through a joint learning framework to improve the performance of sequential recommendation. The overall loss function is as follows:

$$\mathcal{L} = \mathcal{L}_{Rec} + \lambda \mathcal{L}_{Cl}, \tag{12}$$

where λ is a hyperparameter that controls the proportion of the auxiliary contrastive learning task in the two tasks.

4 Experiments

In this section, we conduct extensive experiments to answer the following Research Questions (**RQs**):

- **RQ1:** How does Self-BERT perform compared to the existing methods?
- **RQ2:** How is the sensitivity of the hyper-parameters in Self-BERT?
- **RQ3:** How does the self-guided mechanism contribute to the whole model?
- **RQ4:** How does Self-BERT perform under the issue of data sparsity?

4.1 Experimental Settings

Datasets. We conduct experiments on four real-world public datasets to answer the four **RQs**. The Movielens dataset includes movies, users, and corresponding rating data, commonly used in evaluating recommendation systems. We employ Movielens 1 m (**ML-1M**)[3] and Movielens 20 m (**ML-20M**)[4] to carry out our experiments. McAuley et al. [14] developed a series of datasets based on the Amazon web store, which is split according to the products categories. We select two of the datasets[5], i.e., Beauty and Toys, for experiments.

As for data preprocessing, we follow the standard practice in [13,17,22]. First, we regard the cases with ratings or reviews as positive samples and the rest as negative samples. Then for each user, the corresponding items are sorted chronologically. We follow the "5-core" principle, which means we discard users with less than 5 interactions and items related with less than 5 users. The statistics of the four datasets after data preprocessing are shown in Table 1.

Table 1. Statistics of four datasets after preprocessing.

Dataset	Users	Items	Actions	Avg. length	Sparsity
ML-1M	6,040	3,416	999,611	165.5	95.21%
ML-20M	138,493	18,345	19,984,024	144.3	99.21%
Beauty	22,363	12,101	198,502	8.9	99.93%
Toys	19,412	11,924	167,597	8.6	99.93%

Baselines. We use the following baseline methods for comparison to demonstrate the effectiveness of Self-BERT.

- **Pop:** It is a non-personalized recommendation method based on item popularity. The items that occur most frequently in the interaction sequence are recommended for all users.
- **BPR-MF** [15]**:** It is a matrix factorization model that captures pairwise item correlations and optimizes a Bayesian Personalized Ranking (BPR) loss.
- **GRU4Rec** [9]**:** This method employs GRU, which is a variant of RNN, to obtain better representations of user interaction sequences.
- **Caser** [18]**:** It is a CNN-based model that utilizes convolution kernels in horizontal and vertical orientations to capture local patterns.
- **SASRec** [11]**:** It adopts a left-to-right Transformer encoder to model users' interests dynamically.
- **BERT4Rec** [17]**:** This model regards the recommendation task as the Cloze task to train model so that the Transformer encoder could fuse contextual information for better sequence representations.

[3] https://grouplens.org/datasets/movielens/1m/.
[4] https://grouplens.org/datasets/movielens/20m/.
[5] http://jmcauley.ucsd.edu/data/amazon/.

- **S^3-Rec** [13]**:** It devises four auxiliary SSL objectives to gain the relations among attribute, item, subsequence and sequence. However, for the fairness of the comparison, we remove the modules related to attribute.
- **CL4SRec** [22]**:** It applies contrastive learning to construct self-supervision signals from the original sequences by three data augmentations.

Metrics. To compare the next-item prediction effectiveness among Self-BERT and baseline methods, we follow [11,17] to choose Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) as evaluation metrics. It is worth noting that we rank the predictions on each whole dataset without negative sampling for a fair comparison. We report HR and NDCG with k = 5, 10. For all evaluation metrics, the higher values indicate better model performance.

Implementation Details. We use code provided by the authors for GRU4Rec[6], Caser[7], SASRec[8], and S^3-Rec[9]. We implement BPR, BERT4Rec, and CL4SRec on public resources. All hyperparameters are set following the original papers and tuned on the performance of the validation set. We report the results of each baseline at its optimal hyperparameter setting. For ML-20M, the batch size is 64 due to insufficient GPU memory, while for other datasets, the batch size is 256. We tune the max sequence length on different datasets and choose the corresponding optimal parameters. The contrastive loss proportion is tuned (see Sect. 4.3 for more details) and is finally decided as 0.1. As for other hyperparameters, we follow the guidance from BERT4Rec. We train our model using Adam optimizer with learning rate of 0.0001, $\beta_1 = 0.9$, $\beta_2 = 0.999$.

4.2 Overall Performance Comparison (RQ1)

To answer **RQ1**, we compare the performance of Self-BERT with the baseline methods. These baselines can be divided into two categories, non-SSL-based models and SSL-based models. Table 2 and Table 3 present the best results of all models on four real-world datasets individually. The best score and the second best score in each column are bolded and underlined, respectively. Improvements over the best baseline method are indicated in the last row. From the experimental results, we obtain the following observations:

Comparison with Non-SSL-Based Baselines. Non-personalized methods, such as PopRec and BPR-MF, exhibit worse recommendation performance on all datasets, which indicates the necessity of capturing sequential features in next item recommendation. In sequential recommendation, the Transformer-based models (e.g., SASRec and BERT4Rec) perform better than other models. This phenomenon shows that the self-attention mechanism can more effectively mine the information in the sequence than RNN and CNN. BERT4Rec performs relatively well on datasets with longer average sequence lengths, such as Movielens. We speculate that BERT4Rec can capture more contextual information

[6] https://github.com/hidasib/GRU4Rec.

[7] https://github.com/graytowne/caser_pytorch.

[8] https://github.com/kang205/SASRec.

[9] https://github.com/RUCAIBox/CIKM2020-S3Rec.

as sequence gets longer, hence obtain better performance. However, Self-BERT achieves better performance than BERT4Rec, which verifies the effectiveness of the self-supervised contrastive learning task under a joint learning framework.

Comparison with SSL-Based Baselines. Both S^3-Rec and CL4SRec are SSL-based models, but S^3-Rec performs worse than CL4SRec. One possible reason is that the two-stage training mode may lead to catastrophic forgetting. We also observe that Self-BERT performs better than CL4SRec. It is likely that data augmentations might corrupt the semantic information in sequence. In contrast, Self-BERT uses the complete sequence representation obtained in the encoder without data augmentations, improving the validity of contrastive learning.

Table 2. Overall performance of different methods on Movielens datasets. Bold scores are the best in method group, while underlined scores are the second best. The last row is the relative improvements compared with the best baseline results. (H is short for HR, N is short for NDCG)

Dataset	ML-1M				ML-20M			
Metric	H@5	H@10	N@5	N@10	H@5	H@10	N@5	N@10
PopRec	0.0078	0.0162	0.0052	0.0079	0.0165	0.0311	0.0094	0.0140
BPR-MF	0.0366	0.0603	0.0226	0.0302	0.0280	0.0482	0.0168	0.0233
GRU4Rec	0.0806	0.1344	0.0475	0.0649	0.0691	0.1187	0.0436	0.0595
Caser	0.0912	0.1442	0.0565	0.0734	0.0637	0.1051	0.0398	0.0531
SASRec	0.1071	0.1727	0.0634	0.0845	0.1276	0.1895	0.0842	0.1041
S^3-Rec	0.1020	0.1724	0.0612	0.0839	0.1187	0.1807	0.0775	0.0974
CL4SRec	0.1142	0.1810	0.0705	0.0920	0.1108	0.1782	0.0707	0.0924
BERT4Rec	0.1308	0.2219	0.0804	0.1097	0.1380	0.2092	0.0928	0.1157
Self-BERT	**0.1834**	**0.2620**	**0.1227**	**0.1480**	**0.1732**	**0.2468**	**0.1199**	**0.1436**
Improv	40.21%	18.07%	52.61%	34.91%	25.51%	17.97%	29.20%	24.11%

Table 3. Overall performance of different methods on Amazon datasets. Bold scores are the best in method group, while underlined scores are the second best. The last row is the relative improvements compared with the best baseline results. (H is short for HR, N is short for NDCG)

Dataset	Beauty				Toys			
Metric	H@5	H@10	N@5	N@10	H@5	H@10	N@5	N@10
PopRec	0.0075	0.0143	0.0041	0.0063	0.0066	0.0094	0.0046	0.0055
BPR-MF	0.0143	0.0253	0.0091	0.0127	0.0124	0.0178	0.0087	0.0105
GRU4Rec	0.0206	0.0332	0.0132	0.0172	0.0121	0.0184	0.0077	0.0097
Caser	0.0254	0.0436	0.0154	0.0212	0.0205	0.0333	0.0125	0.0166
SASRec	0.0371	0.0592	0.0233	0.0305	0.0429	0.0652	0.0248	0.0320
S^3-Rec	0.0365	0.0610	0.0228	0.0306	0.0405	0.0644	0.0258	0.0335
CL4SRec	0.0396	0.0630	0.0232	0.0307	0.0434	0.0635	0.0249	0.0314
BERT4Rec	0.0370	0.0598	0.0233	0.0306	0.0371	0.0524	0.0259	0.0309
Self-BERT	**0.0516**	**0.0740**	**0.0350**	**0.0421**	**0.0540**	**0.0759**	**0.0381**	**0.0452**
Improv	30.30%	17.46%	50.21%	37.13%	24.42%	16.41%	47.10%	34.93%

4.3 Parameter Sensitivity (RQ2)

To answer **RQ2**, we investigate the influence of essential hyperparameters in Self-BERT, including the weight of $\mathcal{L}_{Cl}$ and the batch size b.

Impact of Contrastive Learning Loss. In this section, we study the effect of contrastive learning proportion λ in the model. We select two datasets (ML-1M and Beauty) and conduct experiments with different λ values (0, 0.05, 0.1, 0.2, 0.5, 1, 2, 4) while keeping other parameters optimal. The obtained experimental results (NDCG@5) are shown in Fig. 3 (a). We observe that appropriately increasing the value of λ can improve the performance. However, the performance become worse when λ exceeds a certain threshold. The above observation shows that if λ is too large, the contrastive learning dominates the training process, which may influence the performance of sequential recommendation. We finally choose $\lambda = 0.1$ to achieve a balance between the contrastive learning task and the recommendation task for better recommendation performance.

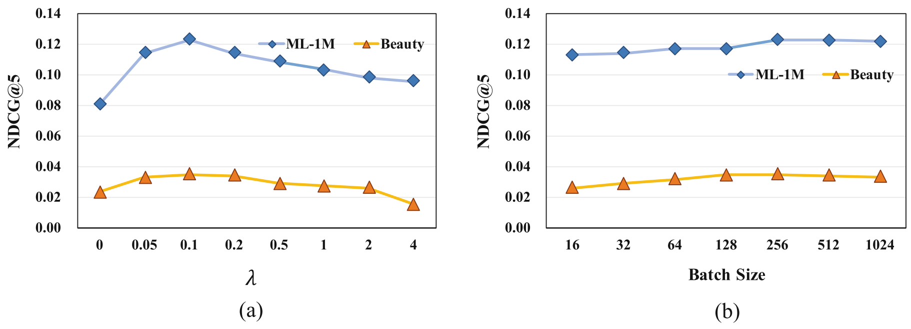


Fig. 3. Performance comparison (in NDCG@5) on Self-BERT w.r.t. different λ (a) and batch size (b) on ML-1M and Beauty datasets.

Impact of Batch Size. In this section, we study the effect of different batch size b on the model. We set the batch size as 16, 32, 64, 128, 256, 512, and 1024 for experiments. The results (NDCG@5) on the ML-1M and Beauty datasets are shown in Fig. 3(b). We observe that a large batch size has an advantage over the smaller ones, but the influence tends to be tiny as the batch size increases. This finding is similar to the experimental results in SimCLR [1]. A larger batch size can provide more negative samples for contrastive learning and promote the model to convergence. According to the experimental results, when the batch size is larger than 256, the effect of the model does not improve much, thus we select batch size $b = 256$.

4.4 Ablation Study (RQ3)

To answer **RQ3**, we conduct ablation study on Self-BERT to analyze the impact of the components in Self-BERT. To verify the effectiveness of these components, we propose the following three variants of Self-BERT: (B) delete the auxiliary contrastive learning task of Self-BERT, (C) only use the last layer output of BERT_{fix} as positive view, (D) cancel the parameter-fix mechanism in BERT_{fix} and use its hidden layer representations as positive views. The experimental results on four datasets are shown in Table 4. From the results, we observe that:

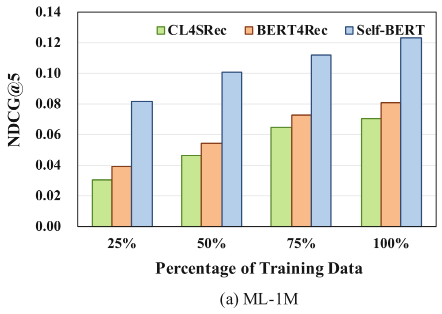
- Self-BERT under the joint learning framework (A) performs better than a single recommendation task model (B). It can be concluded that the auxiliary contrastive learning task can help obtain high quality representations for sequential recommendation.
- In our model, the auxiliary contrastive learning task with one-to-many view constraints (A) performs better than one-to-one view constraints (C). Self-BERT makes full use of the output of all Transformer layer as positive views, which improves the performance of the contrastive learning task by increasing the number of positive views.
- Self-BERT (A) performs better than the model which cancel the parameter-fix mechanism in BERT_{fix} (D). We think that the self-guided mechanism in Self-BERT prevents training signal of BERT_{fix} from being degenerated.

Table 4. Ablation study of Self-BERT.

Model	ML-1M		ML-20M		Beauty		Toys	
	HR@5	NDCG@5	HR@5	NDCG@5	HR@5	NDCG@5	HR@5	NDCG@5
(A) Self-BERT	**0.1834**	**0.1227**	**0.1732**	**0.1199**	**0.0516**	**0.0350**	**0.0540**	**0.0381**
(B) w/o CL	0.1308	0.0804	0.1380	0.0928	0.0370	0.0233	0.0371	0.0259
(C) only H^L	0.1715	0.1106	0.1592	0.1091	0.0479	0.0325	0.0488	0.0353
(D) w/o fix	0.1498	0.0956	0.1464	0.1014	0.0392	0.0279	0.0427	0.0284

4.5 Robustness Analysis (RQ4)

To answer **RQ4**, we conduct experiments to illustrate the robustness of Self-BERT when facing the data sparsity issue. To simulate the data sparsity problem, we only use part of the data (25%, 50%, 75%, 100%) for training and keep the test data unchanged. We compare the proposed model with CL4SRec and BERT4Rec, and the results are shown in Fig. 4. We observe that performance drops when using less training data, but Self-BERT consistently outperforms the other two baseline methods. Although the data sparsity issue effects vary on different datasets, our model can alleviate the influence of this issue for sequential recommendation to some extent.



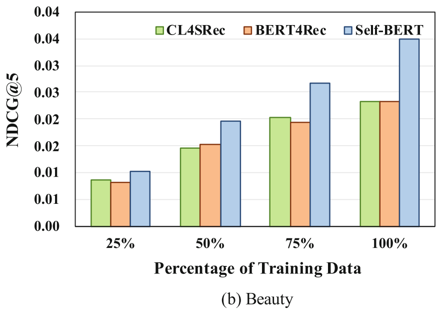


Fig. 4. Model performance (in NDCG@5) comparison w.r.t. sparsity ratio on ML-1M (a) and Beauty (b) datasets.

5 Conclusion

In this paper, we propose self-guided contrastive learning enhanced BERT for sequential recommendation (**Self-BERT**). High-quality contrastive views can be stably generated by introducing the self-guided mechanism, which means the hidden layer representations produced by a fixed BERT encoder are used to guide the training of another trainable BERT encoder. Moreover, we also improve the commonly used contrastive learning loss function (NT-Xent) to make it more suitable for **Self-BERT**. Experimental results on four real-world datasets demonstrate the effectiveness of our **Self-BERT**. We also experimentally verify the data sparsity robustness of our **Self-BERT**.

Acknowledgements. This research was supported by the NSFC(61876117, 61876217, 62176175), the major project of natural science research in Universities of Jiangsu Province (21KJA520004), Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions and Exploratory Self-selected Project of the State Key Laboratory of Software Development Environment.

References

1. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.E.: A simple framework for contrastive learning of visual representations. In: ICML. Proceedings of Machine Learning Research, vol. 119, pp. 1597–1607. PMLR (2020)
2. Chen, W., et al.: Probing simile knowledge from pre-trained language models. In: ACL (2022)
3. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: EMNLP, pp. 1724–1734. ACL (2014)
4. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (1), pp. 4171–4186. Association for Computational Linguistics (2019)
5. Donkers, T., Loepp, B., Ziegler, J.: Sequential user-based recurrent neural network recommendations. In: RecSys, pp. 152–160. ACM (2017)

6. Gao, T., Yao, X., Chen, D.: Simcse: simple contrastive learning of sentence embeddings. In: EMNLP. Association for Computational Linguistics (2021)
7. Goodfellow, I.J., et al.: Generative adversarial networks. CoRR abs/1406.2661 (2014)
8. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.B.: Momentum contrast for unsupervised visual representation learning. In: CVPR, pp. 9726–9735. Computer Vision Foundation/IEEE (2020)
9. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. In: ICLR (Poster) (2016)
10. Jaiswal, A., Babu, A.R., Zadeh, M.Z., Banerjee, D., Makedon, F.: A survey on contrastive self-supervised learning. CoRR abs/2011.00362 (2020)
11. Kang, W., McAuley, J.J.: Self-attentive sequential recommendation. In: ICDM, pp. 197–206. IEEE Computer Society (2018)
12. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
13. Liu, Y., Li, B., Zang, Y., Li, A., Yin, H.: A knowledge-aware recommender with attention-enhanced dynamic convolutional network. In: CIKM, pp. 1079–1088 (2021)
14. McAuley, J.J., Targett, C., Shi, Q., van den Hengel, A.: Image-based recommendations on styles and substitutes. In: SIGIR, pp. 43–52. ACM (2015)
15. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. CoRR abs/1205.2618 (2012)
16. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized Markov chains for next-basket recommendation. In: WWW. ACM (2010)
17. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: BERT4Rec: sequential recommendation with bidirectional encoder representations from transformer. In: CIKM, pp. 1441–1450. ACM (2019)
18. Tang, J., Wang, K.: Personalized top-n sequential recommendation via convolutional sequence embedding. In: WSDM, pp. 565–573. ACM (2018)
19. Vaswani, A., et al.: Attention is all you need. In: NIPS, pp. 5998–6008 (2017)
20. Wang, J., et al.: Knowledge enhanced sports game summarization. In: WSDM (2022)
21. Wang, J., et al.: A survey on cross-lingual summarization. arXiv abs/2203.12515 (2022)
22. Xie, X., et al.: Contrastive learning for sequential recommendation. arXiv preprint arXiv:2010.14395 (2020)
23. Xu, C., et al.: Long- and short-term self-attention network for sequential recommendation. Neurocomputing **423**, 580–589 (2021)
24. You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., Shen, Y.: Graph contrastive learning with augmentations. In: NeurIPS (2020)
25. Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J.M., He, X.: A simple convolutional generative network for next item recommendation. In: WSDM. ACM (2019)
26. Zhao, J., Zhao, P., Zhao, L., Liu, Y., Sheng, V.S., Zhou, X.: Variational self-attention network for sequential recommendation. In: ICDE. IEEE (2021)
27. Zhou, Z., Wang, Y., Xie, X., Chen, L., Liu, H.: Riskoracle: a minute-level citywide traffic accident forecasting framework. In: AAAI, vol. 34, pp. 1258–1265 (2020)
28. Zhou, Z., Wang, Y., Xie, X., Chen, L., Zhu, C.: Foresee urban sparse traffic accidents: a spatiotemporal multi-granularity perspective. IEEE Trans. Knowl. Data Eng. (2020)
29. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Graph contrastive learning with adaptive augmentation. In: WWW, pp. 2069–2080. ACM/IW3C2 (2021)