Future Data Helps Training: Modeling Future Contexts for Session-based Recommendation

Fajie Yuan
Tencent
Shenzhen, China
fajieyuan@tencent.com

Xiangnan He
University of Science and Technology of China
Hefei, China
xiangnanhe@gmail.com

Haochuan Jiang*
University of Edinburgh
Edinburgh, UK
haochuan.jiang@ed.ac.uk

Guibing Guo
Northeastern University
Shenyang, China
guogb@swc.neu.edu.cn

Jian Xiong, Zhezhao Xu, Yilin Xiong
Tencent
Shenzhen, China
{janexiong,zhezhaoxu,plutoxiong}@tencent.com

ABSTRACT

Session-based recommender systems have attracted much attention recently. To capture the sequential dependencies, existing methods resort either to data augmentation techniques or left-to-right style autoregressive training. Since these methods are aimed to model the sequential nature of user behaviors, they ignore the future data of a target interaction when constructing the prediction model for it. However, we argue that the future interactions after a target interaction, which are also available during training, provide valuable signal on user preference and can be used to enhance the recommendation quality.

Properly integrating future data into model training, however, is non-trivial to achieve, since it disobeys machine learning principles and can easily cause data leakage. To this end, we propose a new encoder-decoder framework named *Gap-filling based Recommender* (GRec), which trains the encoder and decoder by a gap-filling mechanism. Specifically, the encoder takes a partially-complete session sequence (where some items are masked by purpose) as input, and the decoder predicts these masked items conditioned on the encoded representation. We instantiate the general GRec framework using convolutional neural network with sparse kernels, giving consideration to both accuracy and efficiency. We conduct experiments on two real-world datasets covering short-, medium-, and long-range user sessions, showing that GRec significantly outperforms the state-of-the-art sequential recommendation methods. More empirical studies verify the high utility of modeling future contexts under our GRec framework.

KEYWORDS

Sequential Recommendation, Encoder and Decoder, Seq2Seq Learning, Gap-filling, Data Leakage

ACM Reference Format:
Fajie Yuan, Xiangnan He, Haochuan Jiang, Guibing Guo, and Jian Xiong, Zhezhao Xu, Yilin Xiong. 2020. Future Data Helps Training: Modeling Future Contexts for Session-based Recommendation. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380116

1 INTRODUCTION

Session-based Recommender system (SRS) has become an emerging topic in the recommendation domain, which aims to predict the next item based on an ordered history of interacted items within a user session. While recent advances in deep neural networks [7, 17, 26, 27] are effective in modeling user short-term interest transition, it remains as a fundamental challenge to capture the sequential dependencies in long-range sessions [14, 25, 35]. In practice, long-range user sessions widely exist in scenarios such as micro-video and news recommendations. For example, users on TikTok[1] may watch 100 micro-videos in 30 minutes as the average playing time of each video takes only 15 seconds.

Generally speaking, there are two popular strategies to train recommender models from sequential data: data augmentation [4, 13, 24, 24, 26, 27] and autoregressive training [11, 35]. Specifically, the data augmentation approach, such as the improved GRU4Rec [24], performs data preprocessing and generates new training sub-sessions by using prefixes of the target sequence, and the recommender then predicts the last item in the sequence. The autoregressive approach models the distribution of an entire sequence in an end-to-end manner, rather than only the last item. This idea results in a typical left-to-right style unidirectional generative model, referred to as NextItNet [35]. The two strategies share similar intuition in that when constructing the prediction function for a target interaction, only its past user behaviors (which we also term as "contexts" in this paper) are taken into account.

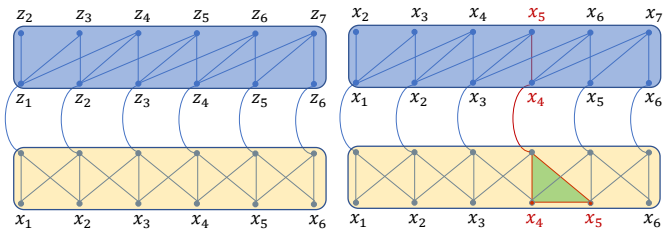
In standard sequential data prediction, it is a straightforward and reasonable choice to predict a target entry based on the past

*Work mostly done at Tencent.

[1] https://www.tiktok.com

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380116



(a) Typical seq2seq learning (√). **(b)** Seq2seq learning for SRS (×).

Figure 1: Examples of ED architecture to model sequential data (encoder as yellow and decoder as blue). (a) is a standard ED architecture where the input x and output z are from two different domains. E.g., in English-to-Chinese machine translation, x and z represent English and Chinese words respectively. (b) is a direct application of ED on SRS with future data modeled. As the predicted items (e.g., x_5 with red color) by the decoder can be observed from the encoder's input, it causes data leakage in training.

entries [10, 28]. However, in sequential recommendation, we argue that such a choice may limit the model's ability. The key reason is that although user behaviors are in the form of sequence data, the sequential dependency may not be strictly held. For example, after a user purchases a phone, she may click phone case, earphone, and screen protector in the session, but there is no sequential dependency among the three items — in other words, it is likely that the user clicks the three items in any order. As such, it is not compulsory to model a user session as a strict sequence. Secondly, the objective of recommendation is to accurately estimate a user's preference, and using more data is beneficial to the preference estimation. As the future data after a target interaction also evidences the user's preference, it is reasonable to believe that modeling the future data can help build better prediction model for the target interaction.

Nevertheless, it is challenging to model with the future data well, since it disobeys machine learning principles and can cause data leakage if not handled properly. Taking the encoder-decoder (ED) neural architecture as an example, which has been extensively used in sequential data modeling [2, 9, 23]. As illustrated in Figure 1 (a), in machine translation, when predicting a target word in a sequence (i.e., sentence), the encoder takes the words from both sides as the input source. Since the source and target words are from different domains, there is no issue of data leakage. However, if we apply the same ED architecture to user session modeling, as illustrated in Figure 1 (b), the data leakage issue arises inevitably. This is because the source and target entries are from the same domain, such that a target entry (to be predicted by the decoder) exactly occurs in the input of the encoder.

To address the above issues, we propose a new SRS method that models the future contexts: *Gap-filling based encoder-decoder framework for sequential Recommendation*, or GRec for short. GRec revises the ED design by tailoring it for future data modeling without data leakage: the encoder and decoder are jointly trained by a gap-filling mechanism [19], which is inspired by the recent development of pretrained language model [3]. Specifically, a portion of items in a user session are deleted by filling in the gap symbols (e.g., "__"). The encoder takes the partially-complete sequence as the input, and the decoder predicts the items of these gaps conditioned on the encoded representation model. Through this way, GRec can force the encoder to be aware of the general user preference, represented by unmasked actions, and simultaneously force the decoder to perform next item generation conditioned on both the past contexts and the encoded general user preference.

The contributions of the work are listed as follows:

- We highlight the necessity of modeling future contexts in session-based recommender system, and develop a general neural network framework GRec that works without data leakage.
- We specify GRec using convolutional neural network with sparse kernels [35], unifying the advantages of both autoregressive mechanism for sequence generation and two-side contexts for encoding.
- We propose a projector neural network with an inverted bottleneck architecture in the decoder, which can enhance the representational bandwidth between the encoder and the decoder.
- We conduct extensive experiments on two real-world datasets, justifying the effectiveness of GRec in leveraging future contexts for session-based recommender system.

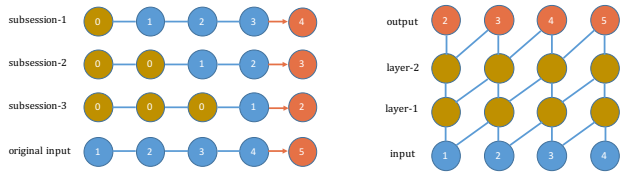
The paper is organized as follows. In Section 2, we review recent advancements in using sequential neural network models for SRS. Particularly, we recapitulate two widely used unidirectional training approaches. In Section 3, we first investigate the straight ways to model bidirectional contexts within a user session, and point out the drawbacks of them for the item recommendation task. After that, we describe in detail the framework and architecture of our proposed GRec. In Section 4, we conduct experiments and ablation tests to verfiy the effectiveness of GRec in the SRS task. In Section 5, we draw conclusions and future work.

2 PRELIMINARIES

In this section, we first define the problem of session-based recommendations. Then, we recapitulate two state-of-the-art left-to-right style sequential recommendation methods. At last, we review previous work of SRS.

2.1 Top-N Session-based Recommendation

The formulation of top-N session-based recomendation in this paper closely follows that in [24, 26, 35]. In SRS, the concept "session" is defined as a collection of items (referring to any objects e.g., videos, songs or queries) that happened at one time or in a certain period of time [13, 31]. For instance, both a list of browsed webpages and a collection of watched videos consumed in an hour or a day can be regarded as a session. Formally, let $\{x_1, ..., x_{t-1}, x_t\}$ be a user session with items in the chronological order, where $x_i \in \mathbb{R}^n$ $(1 \leq i \leq t)$ denotes the index of a clicked item out of a total number of n items in the session. The task of SRS is to train a model so that for a given prefix session data, $x = \{x_1, ..., x_i\}$, it can generate the distribution $\hat{\boldsymbol{y}}$ for items which will occur in the future, where $\hat{\boldsymbol{y}} = [\hat{y}_1, ..., \hat{y}_n] \in \mathbb{R}^n$. $\hat{y}_j$ represents probablity value of item $i+1$ occurring in the next clicking event. In practice, SRS typically makes more than one recommendation by selecting the top-N (e.g., $N = 10$) items from $\hat{\boldsymbol{y}}$, referred to as the top-N session-based recommendations.



(a) Data augmentation (DA). **(b)** Autoregressive models (AR).

Figure 2: Two techniques to train sequential recommendation models. The numbers represent observed itemIDs in each user session. "0" is the padding token. The red token represents the items to be predicted by SRS. (a) The typical data augmentation approach with a number of new subsessions created by spliting the original input session. (b)The typical left-to-right style autoregressive approach. The item that is being predicted is only determined by its previous timesteps, i.e., $p(x_t) = p(x_t|x_1, \ldots, x_{t-1})$. For instance, item "4" is predicted by "1, 2, 3" which achieves the same effect with session-1 in (a). The overall training objectives in (b) can be regarded as the sum of the separate objective of all subsessions in (a).

2.2 The Left-to-Right-style Algorithms

In this section, we mainly review the sequential recommendation models that have the left-to-right fashions, including but not limited to Improved GRU4Rec [24] (short for IGRU4Rec), Caser [26], and NextItNet [35]. Among these models, IGRU4Rec and Caser fall in the line of data augmentation methods, as shown Figure 2 (a), while NextItNet is a typical AR-based generative model, as shown in Figure 2 (b). Note, GRU4Rec, NextItNet can be trained by both DA and AR methods.

2.2.1 ***Data Augmentation*** . The authors in [24] proposed a generic data augmentation method to improve recommendation quality of SRS, which has been further applied in a majority of future work, such as [13, 24, 26, 27]. The basic idea of DA in SRS is to treat all prefixes in the user session as new training sequences [7]. Specifically, for a given user session $\{x_1, \ldots, x_{t-1}, x_t\}$, the DA method will generate a collection of sequences and target labels $\{(x_2|x_1), (x_3|x_1, x_2), \ldots, (x_t|x_1, x_2, \ldots, x_{t-1})\}$ as illustrated in Figure 2 (a). Following this processing, the sequential model is able to learn all conditional dependencies rather than only the last item x_t and the prefix sequence $\{x_1, x_2, \ldots, x_{t-1}\}$. Due to more information learned by additional subsessions, data augmentation becomes an effective way to reduce the overfitting problem especially when the user session is longer and the user-item matrix is sparse. Even though the data augmentation method has been successfully applied in numerous SRS work, it may lead to a break regarding the integrity of the entire user session and significantly increase training times [35].

2.2.2 ***Autoregressive Models (AR)***. The AR-style learning methods [11, 35] propose to optimizing all positions of the original input sequence rather than only the final one. Specifically, the generative model takes $\{x_1, \ldots, x_{t-1}\}$ (or $x_{1:t-1}$) as the input and output probabilities (i.e., softmax) over $x_{2:t}$ by a seq2seq (sequence-to-sequence) manner. Mathematically, the joint distribution of a user session $\{x_1, \ldots, x_{t-1}, x_t\}$ can be factorized out as a product of conditional distributions following the chain rule:

$$p(x) = \prod_{i=1}^{t} p(x_i|x_1, \ldots, x_{i-1}; \Theta) \quad (1)$$

where $p(x_i|x_1, \ldots, x_{i-1})$ denotes the probability of i-th item x_i conditioned on its all prefix $x_{1:i-1}$, Θ is the parameters. With this formulation, each predicted item can be conditioned on all items that are clicked earlier. Correspondingly, the AR method does not rely on the data augmentation technique any more.

As mentioned, both the data augmentation and AR approaches train the user session in an order from left to right. Though it conforms to the generation law of sequential data with natural orders, the way of modeling inevitably neglects many useful future contexts that associate with the target interaction. Particularly in the field of recommendation, user behaviors in the sequence may not obey rigid order relations. Hence, these methods may limit the ability of sequential recommendation models. Moreover, leveraging the addittional future contexts can also be regarded as a way of data augmentation that helps models alleviate the sparsity problem in SRS. Motivated by this, we believe that it is crucial to investigate the impact to sequential recomendation models by taking into account both directional contexts.

2.3 Related Work

Recently, the powerful deep neural network based sequential models have almost dominated the field of session-based recommender systems (SRS). Among these models, GRU4Rec [7] is regarded as the pioneering work that employs the recurrent neural network (RNN) to model the evolution of user preference. Inspired by the success, a class of RNN-based models has been developed. For example, an improved RNN variant in [24] showed promising improvements over standard RNN models by proposing data augmentation techniques. Hidasi et al [6] further proposed a family of alternative ranking objective functions along with effective sampling tricks to improve the cross-entropy and pairwise losses. [17] proposed personalized SRS, while [5, 20] explored how to use content and context features to enhance the recommendation accuracy.

Another line of research work is based on convolutional neural networks (CNN) and attention mechanisms. The main reason is that RNN-based sequential models seriously depend on a hidden state from all the past interactions that cannot fully utilize parallel processing power of GPUs [35]. As a result, their speeds are limited in both training and evaluation. Instead, CNN and purely attention based models are inherently easier to be parallelized since all timesteps in the user session are known during training. The most typical CNN models for SRS is Caser [26], which treats the item embedding matrix as an image and then performs 2D convolution on it. In NextItNet [35], authors argued that the standard CNN architecture and max pooling operation of Caser were not well-suited to model long-range user sequence. Correspondingly, they proposed using stacked dilated CNN to increase the receptive field of higher layer neurons. Moreover, authors claimed that the data augmentation techniques widely used previous work could be simply omitted by developing a seq2seq style objective function. They showed that the autoregressive NextItNet is more powerful

than Caser and more efficient than RNN models for top-N session-based recommendation task. Inspired by the success of Caser and NextItNet, several extended work, e.g., [32, 33], were proposed by employing (or improving) the 1D dilated CNN or 2D CNN to model user-item interaction sequence. Meanwhile, transformer-based self-attention [11, 22, 37] models also demonstrated promising results in the area of SRS. However, it is known that the self-attention mechanism is computationally more expensive than the stacked dilated CNN structure since calculating self-attention of all timesteps requires quadratic complexity. More recently, [14, 25] introduced gating networks to improve SRS by capturing both short- and long-term sequential patterns.

The above mentioned sequential recommenders are built on either an encoder or a decoder architecture. Jointly training an encoder and decoder to model two directional contexts as well as maintain the autoregressive generative mechanism has not been explored in the existing recommendation literature. A relatively relevant work to this paper is NARM [13], which proposed an attention-based 'ED mechanism' for SRS. However, NARM is, in fact, a sequence-to-one architecture rather than the typical seq2seq manner in its decoder network. In other words, NARM decodes the distribution only for the final item, whereas the standard ED model decodes distributions of a complete sequence. By contrast, our proposed GRec is a pseq2pseq (partial-sequence-to-partial-sequence) ED paradigm where its encoder & decoder focus on encoding and decoding incomplete sequences. With the design, GRec combines the advantages of both autoregressive mechanism for sequence generation and two side contexts for encoding.

3 METHODOLOGIES

Before introducing the final solution, we first need to investigate some conventional ways to incorporate future contexts. Then, we shed light on the potential drawbacks of these methods when applying them for the generating task. Motivated by the analysis, we present the gap-filling (or fill-in-the-blank) based encoder-decoder generative framework, namely, GRec. In the following, we instantiate the proposed methods using the dilated convolutional neural network used in NextItNet, giving consideration to both accuracy and efficiency.

3.1 Two-way Data Augmentation

A straightforward approach to take advantage of future data is to reverse the original user input sequence and train the recommendation model by feeding it both the input and reversed output. This type of two-way data augmentation approach has been effectively verified in several NLP tasks [23]. The recommendation models based on both data augmentation and AR methods can be directly applied without any modification. For instance, we show this method by using NextItNet (denoted by NextItNet+), as illustrated below.

$$\begin{aligned} NextItNet+ : & \underbrace{\{x_1, ..., x_{t-1}\}}_{input} \Rightarrow \underbrace{\{x_2, ..., x_t\}}_{output} \\ & \underbrace{\{x_t, ..., x_2\}}_{input} \Rightarrow \underbrace{\{x_{t-1}, ..., x_1\}}_{output} \end{aligned} \tag{2}$$

Issues: The above two-way data augmentation may have two potential drawbacks if using for the item generating task: (1) the left and right contexts of item x_i are modeled by the same set of parameters or same convolutional kernels of NextitNet. While in practice the impact of the left and right contexts to x_i can be very different. That is, the same parameter representation is not accurate from this perspective. (2) The separate training process of the left and right contexts easily results in suboptimal performance since the parameters learned for the left contexts may be largely modified when the model trains the right contexts. In view of this, a better solution is that (1) a single optimization objective consists of both the left and right contexts simultaneously, and (2) the left and right contexts are represented by different sets of model parameters.

3.2 Two-way NextItNets (tNextItNets)

Here, we introduce two-way NextItNets that model the past contexts in the forward direction and model the future contexts in the backward direction. Similar to the forward NextItNet, the backward NextItNet runs over a user session in reverse, predicting the previous item conditioned on the future contexts. The claim here is different from [35], where both the predicted items and its future contexts require to be masked. we only guarantee that the item being predicted will not be accessed by higher-layer neurons. The formulation of backward NextItNet is $p(x) = \prod_{i=1}^{t} p(x_i|x_t, .x_{t-1}, .., x_{i+1}; \overleftarrow{\Theta})$.

Both the forward and backward NextItNets will produce a hidden matrix for a user session in each convolutional layer. Let $\vec{\boldsymbol{h}}_{x_i}$ and $\overleftarrow{\boldsymbol{h}}_{x_i}$ be the item hidden vector x_i calculated by the top layer NexitItNet from the forward and backward directions respectively. To form the two-way NextItNets, we concatenate $\vec{\boldsymbol{h}}_{x_i}$ and $\overleftarrow{\boldsymbol{h}}_{x_i}$, i.e., $\boldsymbol{h}_{x_i} = [\vec{\boldsymbol{h}}_{x_i}; \overleftarrow{\boldsymbol{h}}_{x_i}]$. To combine both directions in the objective function, we maximize the joint log likelihood of both directions.

$$\begin{aligned} p(x) = \prod_{i=1}^{t} & p(x_i|x_1, .x_2, .., x_i; \Theta_e, \vec{\Theta}_{NextItNet}, \Theta_s) \\ & p(x_i|x_t, .x_{t-1}, .., x_{i+1}; \Theta_e, \overleftarrow{\Theta}_{NextItNet}, \Theta_s) \end{aligned} \tag{3}$$

The parameters Θ consist of four parts: the bottom layer item embedding Θ_e, convolutional kernels of NextItNet $\vec{\Theta}_{NextItNet}$ & $\overleftarrow{\Theta}_{NextItNet}$ and weights of softmax layer Θ_s. The idea here has similar spirit with the recent deep contextualized word representation (ELMo) model [16] with the exception that ELMo was designed for word understanding or feature extraction tasks via a Bi-RNN encoder, while we apply the two-way NextItNets to solve the generating task.

Issues: Though tNextItNets can address the training issues mentioned in Section 3.1, the future contexts are actually unapproachable during the generating phase. That is, the backward NextItNet is useless when it is used for inference. The discrepancies between training and predicting may seriously hurt the final recommendation performance since the optimal parameters learned for the two-way NextItNets may be largely suboptimal for the unidirectional NextItNet. Another downside is that two-way NextItNets are essentially a shallow concatenation of independently trained left-to-right and right-to-left models, which have limited expresiveness in modeling complex contextual representations. So, it is unknown

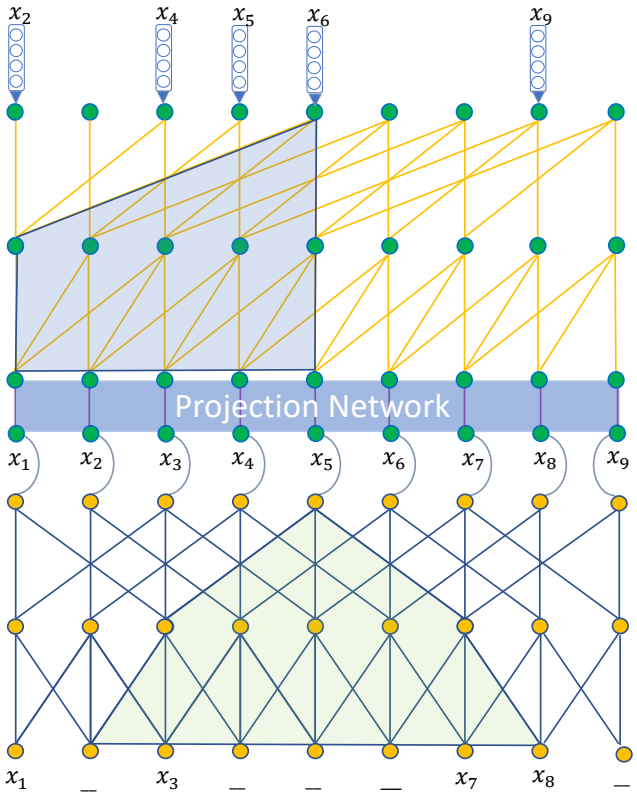


Figure 3: The graphical illustration of GRec with two convolutional layers. The decoder (grean neurons) is stacked on top of the encoder (yellow neurons). The light blue & green areas are the receptive field of x_6. Note the first position is not considered for masking.

whether the proposed two-way NextItNets perform better or not than NextItNet, even though it utilize more contexts.

3.3 Gap-filling Based ED framework

In this subsection, we first present the general framework and neural architecture of GRec. Then, we discuss the relation between GRec and other popular sequential models.

3.3.1 Seq2seq for SRS. First, we introduce the basic concepts of the seq2seq learning for SRS. We denote $(x, z) \in (\mathcal{X}, \mathcal{Z})$ as a sequene pair, where $x = \{x_1, ..., x_t\} \in \mathcal{X}$ represents the user input session sequence with t items, and $z = \{z_2, ..., z_{g+1}\} \in \mathcal{Z}$ represents the output sequence, and $(\mathcal{X}, \mathcal{Z})$ are regarded as source and target domains. Unlike the standard seq2seq scenario (i.e., Figure 1 (a)), we have the following special relations in the SRS task (see Figure 1 (b)): (1) $g = t$; (2) $\{z_1, ..., z_g\}$=$\{x_1, ..., x_t\}$. The goal of a seq2seq model is to learn a set of parameters Θ to describe the conditional probablity $P(z|x, \Theta)$, and usually employs the log likelihood as the objective function [21, 23]: $G(\mathcal{X}, \mathcal{Z}; \Theta) = \sum_{(x,z)\in(\mathcal{X},\mathcal{Z})} \log p(z|x; \Theta)$. Following the decomposition of the chain rule, the probability can be further expressed as an autoregressive manner:

$$P(z|x, \Theta) = \prod_{i=2}^{g} P(z_i|z_{1:i-1}, x; \Theta) = \prod_{i=2}^{t} P(x_i|x_{1:i-1}, x; \Theta) \quad (4)$$

3.3.2 General Framework of Pseq2pseq. As can be seen, it is non-trivial to design a seq2seq learning model using Eq. (4) since the item that is being predicted, e.g., x_i, could be indirectly seen from the encoder network by x. To address this issue, we present the masked-convolution operations by applying the idea of gap-filling (originally designed for the language [19] task) in the ED architecture. Here, we assume items in a user session as words in a sentence. Correspondingly, we could randomly replace some tokens in the sequence with the gap symbol "__". The goal of gap-filling is to predict the truth of these missing tokens.

GRec consists of a modified version of encoder & decoder, and a projector module which is injected into the decoder network. Both the encoder and decoder are described by using the dilated convolutional neural network, although they can be simply replaced with the recurrent [7] and attention [11] networks. The main difference of the encoder and decoder is that the encoder network is built upon the deep bidirectional CNN, while the decoder network is built upon the deep causal CNN. To enhance the brandwidth between the encoder and decoder, we place the decoder on top of the represenation computed by the encoder, and inject a projector network between them. This is in contrast to models that compress the encoder representation to a fixed-length vector [23] or align them by attention mechanism[2] [2, 21].

Formally, given a user session sequence $x = \{x_1, ..., x_t\} \in \mathcal{X}$, we denote $\tilde{x}$ as a partial x, where portions of the items, i.e., $x_\triangle = \{x_{\triangle_1}, ..., x_{\triangle_m}\}$ $(1 \le m < t)$, are randomly replaced with blank mask symbols ("__"). GRec optimizes a pseq2pseq model by predicting $x_\triangle$ in each user session, taking the modified item sequence $\tilde{x}$ as input sequence. The objective function $G(\mathcal{X}; \Theta)$ of GRec is defined as

$$\begin{aligned} G(\mathcal{X}; \Theta) &= \sum_{x\in\mathcal{X}} \log p(x_\triangle|\tilde{x}; \Theta) \\ &= \sum_{x\in\mathcal{X}} \log \prod_{i=1}^{m} p(x_{\triangle_i}|x_{1:\triangle_{i-1}}, \tilde{x}; \Theta) \end{aligned} \quad (5)$$

where Θ consists of the item embeddings of encoder Θ_{en} and decoder Θ_{de}, the convolution weights of encoder Θ_{cnn} and decoder $\vec{\Theta}_{cnn}$, the weights of the projector module Θ_p and softmax layer Θ_s. One may find that there is overlapped data between $\tilde{x}$ and $x_{1:\triangle_{i-1}}$. In fact, since the item embeddings of encoder and decoder are not shared, the overlapped tokens in the encoder and decoder can represent different meanings.

We show the graphical example of Eq. (5) using Figure 3. The decoder of GRec will predict items (i.e., $x_\triangle$) that are masked in the encoder part. As shown in Figure 3, GRec takes an input sequence "x_1, x_3, x_7, x_8" and produces "x_2, x_4, x_5, x_6, x_9" as the output sequence. Taking the generation of item "x_6" as an example, when it is predicted, GRec can leverage the causal relations of the partial sequence "x_1, x_2, x_3, x_4, x_5", and meanwhile leverage the representations of item "x_3, x_7, x_8" via the encoder, where "x_7, x_8" are the future contexts of "x_6". For clarity, we show the comparison of NextItNet (seq2seq) and GRec (pseq2pseq) in terms of model generation

[2] We did not find the basic attention mechanisms introduced in [2, 29] help GRec yield any better results.

as below:

$$\begin{aligned} &NextItNet: \underbrace{\{x_1, x_2, x_3, ..., x_7, x_8\}}_{decoder\ input} \Rightarrow \underbrace{\{x_2, x_3, x_4, ..., x_8, x_9\}}_{decoder\ output} \\ &GRec: \underbrace{\{x_1, \_\_, x_3, \_\_, \_\_, \_\_, x_7, x_8, \_\_, \}}_{encoder\ input} + \underbrace{\{x_1, x_2, x_3, ..., x_9\}}_{decoder\ input} \\ &\Rightarrow \underbrace{\{x_2, x_4, x_5, x_6, x_9\}}_{decoder\ output} \end{aligned} \tag{6}$$

With this design, GRec can take advantage of both the past and future contexts without causing data leakage.

3.3.3 GRec Architecture. In the following, we describe the components of GRec: the embedding layers, the encoder, the decoder, the projector and the softmax layer.

Embedding Layers. The proposed GRec has two distinct embedding layers, namely, the encoder embedding matrix $\widetilde{E} \in \mathbb{R}^{n \times d}$ and decoder embedding matrix $\widehat{E} \in \mathbb{R}^{(n-1) \times d}$, where $n-1$ is the number of items and d is the embedding dimension. Specifically, the encoder of GRec embeds the masked user input sequence $\tilde{x}$ via a look-up table from $\widetilde{E}$, denoted by $\widetilde{E}^{\tilde{x}} \in \mathbb{R}^{t \times d}$, while the decoder embeds the original input sequence x from $\widehat{E}$, denoted by $\widehat{E}^{x} \in \mathbb{R}^{t \times d}$. After the embedding look-up operation, we denote the embeddings of the encoder and decoder as below:

$$\begin{aligned} \widetilde{E}_L^{\tilde{x}} &= \begin{bmatrix} \widetilde{E}_L^{\tilde{x}_1} & \widetilde{E}_L^{\tilde{x}_0} & \widetilde{E}_L^{\tilde{x}_3} & \widetilde{E}_L^{\tilde{x}_0} & \cdots & \widetilde{E}_L^{\tilde{x}_0} \end{bmatrix} \\ \widehat{E}_L^{x} &= \begin{bmatrix} \widehat{E}_L^{x_1} & \widehat{E}_L^{x_2} & \widehat{E}_L^{x_3} & \widehat{E}_L^{x_4} & \cdots & \widehat{E}_L^{x_t} \end{bmatrix} \end{aligned} \tag{7}$$

where L represents the L-th user sequence, and $\widetilde{E}_e^{\tilde{x}_0}$ represents the embedding vector of blank symbol, i.e., '__', in the encoder embedding.

Encoder: Deep Bidirectional CNNs by Gap-filling. We implement the encoder network with a series of stacked 1D dilated convolutional layers inspired by NextItNet. To alleviate gradient vanishing issues, we wrap every two dilated layers by a residual block. Unlike NextItNet, the convolutional operations of the encoder are not causal. Each higher-layer neurons can see both its left and right contexts. With the gap-filling design, these neurons are forced to understand the unmasked contexts in the sequence. It is also worth mentioning that the proposed gap-filling mechanism is dynamic and random, which masks different portions of the item sequence in different training batches. Formally, we define the output of the encoder network with two stacked layers in Figure 3 as:

$$\mathcal{F}_{encoder}(\widetilde{E}_L^{\tilde{x}}) = \widetilde{E}_L^{\tilde{x}} + \mathcal{F}_{non\_cauCNN}(\widetilde{E}_L^{\tilde{x}}) \tag{8}$$

where $\mathcal{F}_{non\_cauCNN}(\widetilde{E}_L^{\tilde{x}})$ denotes the block function of non-causal CNNs defined as

$$\mathcal{F}_{non\_cauCNN}(\widetilde{E}_L^{\tilde{x}}) = RELU(\mathcal{L}_n(\psi_2(RELU(\mathcal{L}_n(\psi_1(\widetilde{E}_L^{\tilde{x}})))))) \tag{9}$$

where $RELU$ and $\mathcal{L}_n$ denote non-linear activation function [15] and layer-normalization [1], ψ_1 and ψ_2 are non-causal CNNs with 1-dilated and 2-dilated filters respectively. In practice, one can repeat the basic encoder structure several times to capture long-term and complex dependencies.

Decoder: Deep Causal CNNs by Gap-predicting. The decoder is composed of the embedding layer $\widehat{E}_L^{x}$, the projector and the causal CNN modules by which each position can only attend leftward. The CNN component strictly follows NextItNet with the exception that it is allowed to estimate the probabilities of only the masked items in the encoder, rather than the entire sequence in NextItNet. Meanwhile, before performing the causal CNN operations, we need to aggregate the final ouput matrix of the encoder and the embedding matrix of the decoder, and then pass them into the projector network, which is described later. Formally, the final hidden layer (before softmax layer) of the decoder can be represented as

$$\mathcal{F}_{decoder}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x}) = \mathcal{F}_{PR}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x}) + \mathcal{F}_{cauCNN}(\mathcal{F}_{PR}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x})) \tag{10}$$

where

$$\begin{aligned} &\mathcal{F}_{cauCNN}(\mathcal{F}_{PR}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x})) \\ &= RELU(\mathcal{L}_n(\phi_2(RELU(\mathcal{L}_n(\phi_1(\mathcal{F}_{PR}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x}))))))) \end{aligned} \tag{11}$$

where $\mathcal{F}_{PR}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x})$ and $\mathcal{F}_{cauCNN}(\mathcal{F}_{PR}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x}))$ are the outputs of projection layers and causal CNN layer respectively, ϕ_1 and ϕ_2 are causal CNNs with 1-dilated and 2-dilated filters respectively.

Projector: Connecting Encoder & Decoder. Although the output hidden layer of the encoder, i.e., $\mathcal{F}_{encoder}(\widetilde{E}_L^{\tilde{x}})$ and the input embedding layer of the decoder, i.e., $\widetilde{E}_L^{\tilde{x}_0}$ have the same tensor shape, we empirically find that directly placing the decoder on top of the encoder by element-wise addition may not offer the best results. To maximize the representational brandwith between the encoder and decoder, we propose an additional projection network (or projector in short) in the decoder. Specifically, the projector is an inverted bottleneck residual architecture, which consists of the projection-up layer, the activation function layer, the projection-down layer and a skip connection between the projection-up and projection-down layers. The projector first projects the original d-dimensional channels into a larger dimension with the $1 \times 1 \times d \times f$ ($f = 2d$ in this paper) convolutional operations. Following by the non-linearity, it then projects the f channels back to the original d dimensions with the $1 \times 1 \times f \times d$ convolutional operations. The output of the projector is given as

$$\mathcal{F}_{PR}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x}) = \mathcal{F}_{agg}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x}) + \phi_{down}(RELU(\phi_{up}(\mathcal{F}_{agg}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x})))) \tag{12}$$

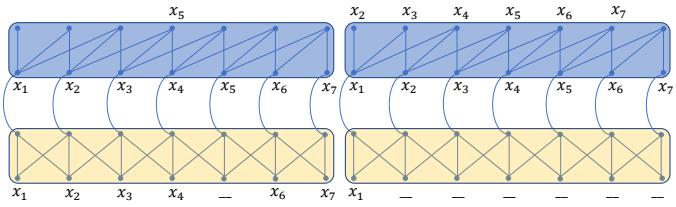
where

$$\mathcal{F}_{agg}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x}) = \mathcal{F}_{encoder}(\widetilde{E}_L^{\tilde{x}}) + \widehat{E}_L^{x} \tag{13}$$

where ϕ_{up} and ϕ_{down} represent the projection-up and projectin-down operations.

Model Training & Generating. As mentioned in Eq. (5), GRec only takes the masked positions into consideration rather than the complete sequence. Hence, we first perform the look-up table by retrieving the hidden vectors of the masked positions from $\mathcal{F}_{decoder}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x})$, denoted by $\mathcal{F}_{decoder}^{x_\triangle}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x})$. Then, we feed these vectors into a fully-connected neural network layer which projects them from the d-dimentional latent space to the n-dimentional softmax space. The calculated probabilities of the masked items $x_\triangle$ are given as

$$p(x_\triangle | \tilde{x}; \Theta) = softmax(\mathcal{F}_{decoder}^{x_\triangle}(\widetilde{E}_L^{\tilde{x}}, \widehat{E}_L^{x})\boldsymbol{W} + \boldsymbol{b}) \tag{14}$$



(a) GRec with one masked item (b) GRec with t masked items.

Figure 4: GRec variants by changing the gap-filling strategy

where $\boldsymbol{W} \in \mathbb{R}^{d \times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$ are weight matrix and the corresponding bias term. Finally, we are able to optimize Eq. (5) by gradient ascent (or gradient descent on the negative of Eq. (5)) . Since GRec only estimates portions of items in each batch, it needs more training steps to converge compared to NextItNet-style models (which estimate the entire sequence), but much fewer steps compared to the left-to-right data augmentation based models (e.g., Caser) (which estimate only the last item).

Once the model is well trained, we can use it for item generation. Unlike the training phase during which the encoder has to mask a certain percentage of items, GRec is able to directly compute the softmax of the final position of the final layer at the inference time without performing mask operations.

3.3.4 Connection to Existing Models. Our work is closely related to NextItNet and the well-known bidirectional language model BERT [3, 22]. In this subsection, we show the connections of GRec to NextItNet-style and BERT-style models. For clarity, we omit the projection layers during discussion.

As shown in Figure 4 (a), when $m = 1$, the encoder masks only one position, i.e., x_5, from the input sequence, and correspondingly the decoder only predicts this masked token, conditioned on all other items in this sequence. Mathematically, we have $p(x_\triangle|\tilde{x}) = p(x_5|x \backslash x_5)$. If we further mask the input of the decoder with '__', GRec reduces to a standard encoder with one softmax output, and is very similar to the well-known bidirectional language model BERT, with the only exception that BERT applies the Transformer [29] architecture while GRec uses the stacked 1D dilated CNNs. In this simple case, GRec reduces to a sequence-to-one model and loses its autoregressive property. In fact, The DA-based recommendation mdoels, such as Caser, IGRU4Rec and NARM, can also be seen as sequence-to-one models which just apply different neural network infrastructures.

When $m = t-1$, all items (except the first position) in the encoder will be masked, and the decoder will predict these masked items from x_2 to x_t, as illustrated in Figure 4 (b). In this case, the encoder of GRec becomes almost ineffective. If we remove the encoder, GRec becomes exactly NextItNet. Note that GRec with $m = t - 1$ is very likely to perform worse than NextItNet. This is because in such case the encoder of GRec introduces many additional noises, which makes the decoder much harder to be optimized.

In summary, our proposed GRec can be seen as a pseq2pseq model that jointly trains the encoder and decoder for the sequential recommendation tasks. In contrast to NextItNet-style models, GRec is able to model both the past and future contexts. In contrast to BERT-style models, GRec is more suitable to the generation task due to its autoregressive process. In contrast to the standard seq2seq encoder-decoder models, GRec does not have the data leakage problem when the encoder and decoder are fed into the same input sequence.

Table 1: Statistics of the datasets. "M" is short for million.

DATA	*#actions*	*#sequences*	*#items*	k
TW10	9,986,953	1,048,575	65,997	10
ML30	25,368,155	858,160	18,273	30
ML100	25,240,741	300,624	18,226	100

4 EXPERIMENTS

As the key contribution of this work is to improve the existing left-to-right style learning algorithms for SRS, we evaluate GRec on real-world datasets with short-, medium- and long-range sessions, and conduct extensive ablation studies to answer the following research questions:

(1) **RQ1:** Whether the three proposed approaches perform better than the existing left-to-right sytle models? Which way performs best?
(2) **RQ2:** How does GRec perform with different gap-filling strategies?
(3) **RQ3:** What are the effects of other key modules of GRec? For example, does it benefit from the proposed projector module?
(4) **RQ4:** Is GRec a general framework or does it work well by replacing the encoder and decoder with other types of neural networks?

4.1 Experimental Settings

4.1.1 ***Datasets.*** We conduct experiments on two real-world datasets with three different session lengths.

ML-latest[3]. This dataset was created on September 26, 2018 by MovieLens. Since the original dataset contains cold items, we perform a basic preprocessing by filtering out items that appear less than 20 times, similar to [26]. We then generate the interaction sequence of the same user according to the chronological order. We split the sequence into subsequence every k movies. If the length of the subsequence is less than k, we pad with zero in the beginning of the sequence to reach k. For those with length less than l, we simply remove them in our experiments. In our experiments, we set k=30 with l=10 and k=100 with l=20, which results in two datasets, namely, ML30 and ML100.

TW10[4]. This is a private dataset which was created on October, 2018 by the Weishi Team at Tencent Inc.. TW10 is a short video dataset, in which the averaging playing time of each video is less than 30 seconds. Since the cold users and items have been trimmed by the official provider, we do not need to consider the cold-start problem. Each user sequence contains 10 items at maximum. Table 1 summarizes the statistics of evaluated datasets in this work.

[3] http://files.grouplens.org/datasets/movielens/
[4] https://weishi.qq.com

Table 2: Accuracy comparison. MostPop returns item lists ranked by popularity. For each measure, the best result is indicated in bold.

DATA	*Models*	*MRR@5*	*MRR@20*	*HR@5*	*HR@20*	*NDCG@5*	*NDCG@20*
TW10	*MostPop*	0.0055	0.0127	0.0203	0.0970	0.0091	0.0305
	Caser	0.0780	0.0916	0.1330	0.2757	0.0916	0.1317
	GRU4Rec	0.0786	0.0926	0.1325	0.2808	0.0919	0.1335
	NextItNet	0.0848	0.0992	0.1408	0.2931	0.0986	0.1414
	NextItNet+	0.0698	0.0844	0.1214	0.2775	0.0825	0.1218
	tNextItNet	0.0813	0.0958	0.1376	0.2896	0.0953	0.1380
	GRec	**0.0901**	**0.1046**	**0.1498**	**0.3021**	**0.1049**	**0.1477**
ML30	*MostPop*	0.0030	0.0058	0.0098	0.0405	0.0047	0.0132
	Caser	0.0622	0.0739	0.1074	0.2323	0.0733	0.1083
	GRU4Rec	0.0652	0.0788	0.1156	0.2589	0.0776	0.1179
	NextItNet	0.0704	0.0849	0.1242	0.2756	0.0837	0.1263
	NextItNet+	0.0564	0.0711	0.1051	0.2609	0.0685	0.1121
	tNextItNet	0.0658	0.0795	0.1164	0.2605	0.0782	0.1188
	GRec	**0.0742**	**0.0889**	**0.1300**	**0.2850**	**0.0879**	**0.1315**
ML100	*MostPop*	0.0025	0.0045	0.0076	0.0301	0.0037	0.0099
	Caser	0.0492	0.0605	0.0863	0.2074	0.0584	0.0922
	GRU4Rec	0.0509	0.0632	0.0909	0.2211	0.0608	0.0974
	NextItNet	0.0552	0.0687	0.1007	0.2411	0.0664	0.1059
	NextItNet+	0.0487	0.0615	0.0942	0.2321	0.0600	0.0983
	tNextItNet	0.0518	0.0642	0.0927	0.2239	0.0619	0.0986
	GRec	**0.0588**	**0.0720**	**0.1057**	**0.2477**	**0.0702**	**0.1101**

4.1.2 ***Evaluation Protocols.*** We randomly split all user sequences into training (80%), validation (10%) and testing (10%) sets. We evaluate all models by three popular top-N metrics, namely MRR@N (Mean Reciprocal Rank), HR@N (Hit Ratio) and NDCG@N (Normalized Discounted Cumulative Gain) [34–36]. N is set to 5 and 20 for comparison. The HR intuitively measures whether the ground truth item is on the top-N list, while the NDCG & MRR account for the hitting position by rewarding higher scores to hit at a top rank. For each user sessions in the testing sets, we evaluate the accuracy of the *last* (i.e., next) item following [11, 35].

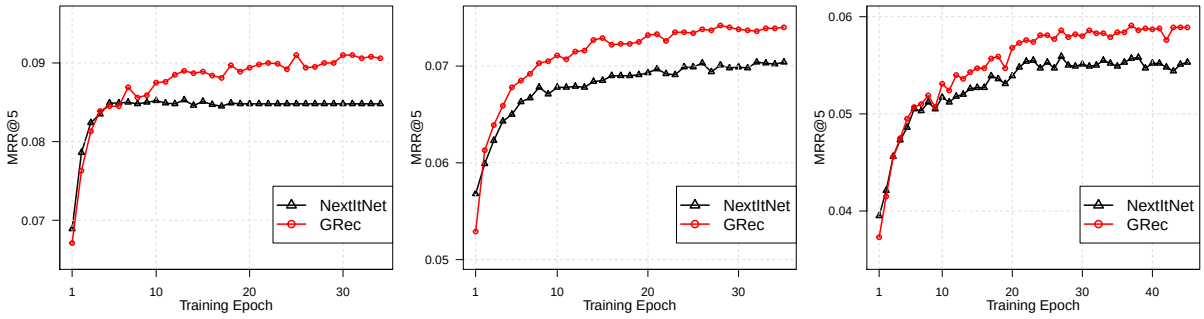
4.1.3 ***Compared Methods.*** We compare the proposed augmentation methods with three typical sequential recommendation baselines, namely, GRU4Rec [7], Caser [26] and NextItNet [35], particularly with NextItNet since GRec can be seen as an extension of NextItNet. We train Caser using the data augmentation method, and train GRU4Rec and NextItNet based on the AR method. For fair comparisons, all methods use the cross-entropy loss function.

4.1.4 ***Implementation Details.*** For comparison purpose, we follow the common practice in [11, 14, 18, 30] by setting the embedding dimension d to 64 for all models. The hidden dimensions are set the same value as embedding dimension d. Note that methods with other d (e.g., $d = 16, 256, 512$) yield very different results, but their performance behaviors keep similar. The learning rate is set to 0.001 in this paper. Other hyper-parameters of baseline methods are empirically tuned according to performance on validation sets. NextItNet+, tNextItNets and GRec use exactly the same hyper-parameters ($q = 128$ for TW10, $q = 256$ for ML30 and ML100) as NextItNet since they can be regarded as variants of NextItNet. The dilated convolution kernels for both the encoder and decoder are set to 3. The dilated convolutional layers are stacked using dilation factors $\{1, 2, 2, 4, 1, 2, 2, 4, 1, 2, 2, 4, \}$ (6 residual blocks with 12 CNN layers), $\{1, 2, 4, 8, 1, 2, 4, 8\}$ (4 blocks with 8 CNN layers), and $\{1, 2, 4, 8, 1, 2, 4, 8, 1, 2, 4, 8\}$ (6 blocks with 12 CNN layers) on TW10, ML30 and ML100 respectively. We perform sampled softmax [8] on TW10 and full softmax on ML30 and ML100 for NextItNet, NextItNet+, tNextItNets and GRec throughout this paper. All models use the Adam [12] optimizer. All results of GRec use $\gamma = 50\%$ as the gap-filling percentage without special mention.

4.2 Performance Comparison (RQ1)

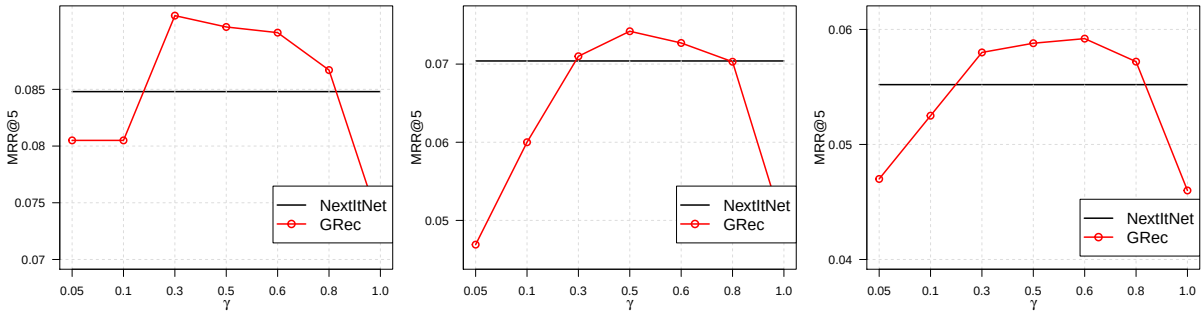
Table 2 presents the results of all methods on three datasets, namely, the short-range session dataset TW10, medium-range ML30, and long-range ML100. We first observe that NextitNet achieves significantly better results than Caser and GRU4Rec on all three datasets. This is consistent with the observation in [35] since (1) with a fair comparison setting, the AR-based optimization method is usually more effective than the data augmentation based method for the sequence generating task; (2) the stacked dilated residual block architecture in NextItNet is capable of capturing more complex and longer sequential dependencies, while the max-pooling operations and shallow structure in Caser inevitably lose many important temporal signals and are far from optimal [25], particularly for modeling long-range sequences.

In what follows, we focus on comparing our proposed methods with NextItNet as they use similar neural network modules and the same hyper-parameters. First, among the three proposed augmentation methods, NextItNet+ and tNextItNet yield consistently worse results than NextItNet, whereas GRec outperforms NextItNet by a large margin. The results of NextItNet+ and tNextItNet indicate that the trivial two-way augmentation methods are not enough to guarantee better recommendation accuracy compared with the unidirectional model, although they are trained with more data or more parameters. The results are predictable since, as we mentioned



(a) TW10 (b) ML30 (c) ML100

Figure 5: Convergence behaviors of GRec and NextItNet. All hyper-parameters are kept the same for the two models. One training epoch in x-axis is 10000∗128, 10000∗256, and 3000∗256 sequences on TW10, ML30 and ML100 respectively, where 128 and 256 are the batch size. Note we perform early stop on TW10 after 20 epoches when NextItNet fully converges and plot the same results in the following epoches, as shown in (a).



(a) TW10 (b) ML30 (c) ML100

Figure 6: Performance trend of GRec by tuning the percentage of masked items in the input sequence. All other hyper-parameters are kept unchanged.

Table 3: Impact of the projector module regarding MRR@5. NextItNetP represents NextItNet with projector. GRecN represents GRec without projector.

DATA	*NextItNet*	*NextItNetP*	*GRec*	*GRecN*
TW10	0.0848	0.0843	0.0901	0.0880
ML30	0.0704	0.0702	0.0742	0.0720
ML100	0.0552	0.0558	0.0588	0.0577

before, the parameters learned by the right contexts in NextItNet+ may be incompatible with those learned from the left contexts using the same convolutional filters. Even though tNextItNet applies two independent networks, the discrepancies during training and inference phases are very harmful for the recommendation accuracy.

Second, we observe that GRec with the pseq2pseq structure signficantly exceeds NextItNet, as demonstrated in Table 2. The results indicate that an appropriate way of modeling by using additional (i.e., future) contextual features does improve the recommendation accuracy for the unidirectional model which attends to only the past contexts. Moreover, we plot the convergence comparison of GRec and NextItNet in Figure 5. The results show that NextItNet converges a bit faster and better than GRec in the first several epoches, but shows poorer results than GRec after more training epoches. The slightly slower convergence behavior is because the loss function of GRec only considers a partial sequence whereas NextItNet loss leverages the loss of complete sequence during training (also refer to Eq. (6)). But obviously, the improved performance gains of GRec far outweigh the marginally increased training cost.

4.3 Impact of the Gap-filling Strategies (RQ2)

Table 6 shows the performance change of GRec with different gap-filling percentages. We fix all other hyper-parameters by tuning γ. As clearly shown, too large or too small γ typically achieves suboptimal performance. The highest recommendation accuracy is obtained when γ is between 30% to 50%. The is because masking too much percentage of items in the user session is very likely to (1) discard important future contexts; (2) introduce noises due to the masked tokens; and (3) bring more discrepancies between training and inference phases, as explained in Section 3.3.4. E.g.,

Table 4: GRec vs. its encoder regarding MRR@5. γ is set to 0.5 for both GRec and its encoder.

DATA	TW10	ML30	ML100
GRec	0.0901	0.0742	0.0588
Encoder	0.0808	0.0592	0.0489

Table 5: GRec vs. NextItNet with $d = 512$ regarding MRR@5.

DATA	TW10	ML30	ML100
NextItNet	0.111	0.105	0.093
GRec	0.117	0.114	0.103

when $\gamma = 1.0$, no future contexts are leveraged, and the encoder of GRec becomes a neural network with only noises. In this case, GRec performs even worse than the standard NextItNet. On the other side, GRec will lose its autoregressive advantage when γ is smaller, and becomes an simple encoder network or a sequence-to-one model when only one item is masked. With this setting, the sequential and recurrent patterns will not be captured any more. Hence, there is a clear trade-off for GRec between taking advantage of future contexts and making use of the autoregressive property.

4.4 Ablation Studies (RQ3)

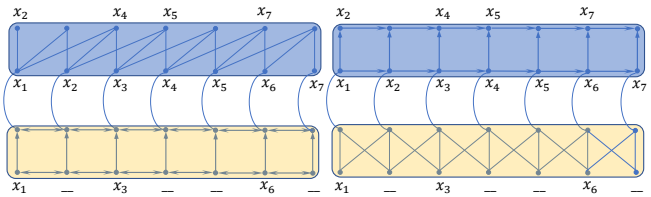
In this subsection, we first investigate the effectiveness of the projector module. One may argue that the improved performance of GRec relative to NextItNet may come from the additional projector module. To clear up the uncertainty, we perform a fair ablation study by removing the projector for GRec as well as injecting it for NextItNet. We have the following observations according to Table 3: (1) The projector indeed helps GRec achieve better performance by comparing GRec & GRecN; (2) NextItNet is inferior to GRec even with the projector by comparing GRec & NextItNetP; (3) GRec still exceeds NextItNet even without the projector by comparing GRecN & NextItNet.

Second, we also report results of GRec with only the encoder network since the encoder itself is also able to leverage two directional contexts. To do so, we remove the decoder of GRec and place the softmax layer on the encoder during training. At the generating phase, we just need to replace the last item by "__", and retrieve the top-N scored items for comparison. With this special case, GRec reduces to a BERT-like bidirectional encoder model. We report the MRR@5 in Table 4. As can be seen, GRec largely exceeds its encoder on all three datasets. The findings confirm our previous analysis since the bidirectional enocder network is not autoregressive and fail to explicitly model the sequential patterns of previous interactions over time. In addition, some of the left contexts are missing because of the gap-filling mechanism, which also results in unsatisfied performance.

In addition, we also report NextItNet & GRec with a very large d in Table 5 (along with much longer training time and larger memory consumption). As shown, GRec obtains significant improvements relative to NextItNet, similar to the observations in Table 2.

Table 6: Recurrent variants of GRec regarding MRR@5..

DATA	*ReCd*	*NextItNet*	*CeRd*	*GRU*
TW10	0.0879	0.0843	0.0876	0.0786
ML30	0.0728	0.0704	0.0712	0.0652
ML100	0.0582	0.0552	0.0571	0.0509



(a) ReCd: BiRNN+ causal CNN. **(b)** CeRd: non-causal CNN + RNN.

Figure 7: GRec variants with recurrent encoder or decoder.

4.5 GRec Variants (RQ4)

Since GRec is a general ED framework, one can simply replace the original dilated convolutional neural network with other types of neural networks, such as RNN. For the sake of completeness, we demonstrate two GRec variants in Figure 7. ReCd represents GRec with recurrent encoder network (Bi-GRU) and convolutional decoder network, while CeRd represents GRec with convolutional encoder network and recurrent decoder network (GRU). We report results on Table 6 and make two observations: (1) GRec still exceeds NextItNet even it utilizes Bi-GRU as encoder by comparing ReCd & NextItNet; (2) GRec outperforms GRU when it utilizes the typical GRU as decoder by comparing CeRd & GRU; (3) in general, the GRec framework using stacks of convolutional blocks for both its encoder and decoder performs better than its variants using RNN for either encoder or decoder. The above observations further verify the generality and flexibility of GRec for processing future contexts.

5 CONCLUSION

In this paper, we perform studies on how to incorporate future contexts for the typical left-to-right style learning algorithms in the task of SRS. The motivation is that the architectures of autoregressive-based sequential recommendation models fail to model the past and future contexts simultaneously. To maintain the autoregressive property as well as utilize two directional contexts, we present GRec, a novel pseq2pseq encoder-decoder neural network recommendation framework with gap-filling based optimization objective. GRec is general and flexible, which jointly trains the encoder and decoder on the same user action sequence without causing the data leakage issue. Through ablations and controlled experiments, we demonstrate that GRec is more powerful than the traditional unidirectional models. For future work, we are interested in studying whether the right contexts or GRec can improve the recommendation diversity for SRS.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (61972372, U19A2079).

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

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

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

[4] Hui Fang, Guibing Guo, Danning Zhang, and Yiheng Shu. 2019. Deep Learning-Based Sequential Recommender Systems: Concepts, Algorithms, and Evaluations. In *International Conference on Web Engineering*. Springer, 574–577.

[5] Youyang Gu, Tao Lei, Regina Barzilay, and Tommi S Jaakkola. 2016. Learning to refine text based recommendations.. In *EMNLP*. 2103–2108.

[6] Balázs Hidasi and Alexandros Karatzoglou. 2017. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. *arXiv preprint arXiv:1706.03847* (2017).

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

[8] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* (2014).

[9] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* (2016).

[10] Nal Kalchbrenner, Aäron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. 2017. Video pixel networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1771–1779.

[11] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[12] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *CIKM*. ACM, 1419–1428.

[14] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical Gating Networks for Sequential Recommendation. *arXiv preprint arXiv:1906.09217* (2019).

[15] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*. 807–814.

[16] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[17] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 130–137.

[18] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 273–282.

[19] Keisuke Sakaguchi, Yuki Arase, and Mamoru Komachi. 2013. Discriminative approach to fill-in-the-blank quiz generation for language learners. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vol. 2. 238–242.

[20] Elena Smirnova and Flavian Vasile. 2017. Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks. *arXiv preprint arXiv:1706.07684* (2017).

[21] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked Sequence to Sequence Pre-training for Language Generation. *arXiv preprint arXiv:1905.02450* (2019).

[22] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1441–1450.

[23] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[24] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 17–22.

[25] Jiaxi Tang, Francois Belletti, Sagar Jain, Minmin Chen, Alex Beutel, Can Xu, and Ed H Chi. 2019. Towards neural mixture recommender for long range dependent user sequences. In *The World Wide Web Conference*. ACM, 1782–1793.

[26] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *ACM International Conference on Web Search and Data Mining*.

[27] Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D Convolutional Networks for Session-based Recommendation with Content Features. In *RecSys*. ACM.

[28] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. 2016. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*. 4790–4798.

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

[30] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 515–524.

[31] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A Survey on Session-based Recommender Systems. *arXiv preprint arXiv:1902.04864* (2019).

[32] An Yan, Shuo Cheng, Wang-Cheng Kang, Mengting Wan, and Julian McAuley. 2019. CosRec: 2D Convolutional Neural Networks for Sequential Recommendation. *arXiv preprint arXiv:1908.09972* (2019).

[33] Jiaxuan You, Yichen Wang, Aditya Pal, Pong Eksombatchai, Chuck Rosenburg, and Jure Leskovec. 2019. Hierarchical Temporal Convolutional Networks for Dynamic Recommender Systems. In *The World Wide Web Conference*. ACM, 2236–2246.

[34] Fajie Yuan, Guibing Guo, Joemon M Jose, Long Chen, Haitao Yu, and Weinan Zhang. 2016. Lambdafm: learning optimal ranking with factorization machines using lambda surrogates. In *CIKM*. ACM, 227–236.

[35] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 582–590.

[36] Fajie Yuan, Xin Xin, Xiangnan He, Guibing Guo, Weinan Zhang, Chua Tat-Seng, and Joemon M Jose. 2018. fBGD: Learning embeddings from positive unlabeled data with BGD. (2018).

[37] Shuai Zhang, Yi Tay, Lina Yao, and Aixin Sun. 2018. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414* (2018).