Augmenting Sequential Recommendation with Pseudo-Prior Items via Reversely Pre-training Transformer

Zhiwei Liu*, Ziwei Fan*, Yu Wang, Philip S. Yu
Department of Computer Science, University of Illinois at Chicago
{zliu213,zfan20,ywang617,psyu}@uic.edu

ABSTRACT

Sequential Recommendation characterizes the evolving patterns by modeling item sequences chronologically. The essential target of it is to capture the item transition correlations. The recent developments of transformer inspire the community to design effective sequence encoders, *e.g.,* SASRec and BERT4Rec. However, we observe that these transformer-based models suffer from the cold-start issue, *i.e.,* performing poorly for short sequences. Therefore, we propose to augment short sequences while still preserving original sequential correlations. We introduce a new framework for **A**ugmenting **S**equential **Re**commendation with **P**seudo-prior items (ASReP). We firstly pre-train a transformer with sequences in a reverse direction to predict prior items. Then, we use this transformer to generate fabricated historical items at the beginning of short sequences. Finally, we fine-tune the transformer using these augmented sequences from the time order to predict the next item. Experiments on two real-world datasets verify the effectiveness of ASReP. The code is available on https://github.com/DyGRec/ASReP.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Sequential Recommendation, Transformer, Cold-start, Augmentation, Pre-training

ACM Reference Format:
Zhiwei Liu*, Ziwei Fan*, Yu Wang, Philip S. Yu. 2021. Augmenting Sequential Recommendation with Pseudo-Prior Items via Reversely Pre-training Transformer. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3404835.3463036

1 INTRODUCTION

Recommender systems, which predict the potential interests of users to items, are widely applied in online platforms [9–11, 24, 27, 33] nowadays. Because of the dynamic characteristics of users' behavior, characterizing the evolving patterns of users' historical records is necessary [4, 5, 8, 23, 26, 32]. Among them, the Sequential Recommendation (SR) [4, 7, 16, 17, 30] is studied a lot for decades. It models the sequential patterns of users' transactions on items. Therefore, the next item can be inferred.

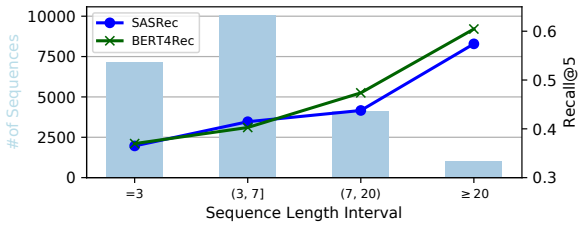

Figure 1: The sequence length distribution (bar) and the corresponding Recall@5 (line) on Amazon Beauty dataset.

The essential target in SR is to capture the item transition correlations [12]. The recent developments of transformer [18, 28] provide a powerful backbone to embed sequence, which can effectively model item correlations. SASRec [4] is the pioneering work adopting transformer to complete SRs. It employs a dot-product self-attention module to learn the importance of items at different positions in a sequence. BERT4Rec [17], which is inspired by bi-direction transformer [1], models the item transition correlations from both left-to-right and right-to-left directions. Other recent works [7, 14, 19] also justify the efficacy of transformer in revealing item correlations in sequences.

However, transformer-based sequence encoders fail to achieve satisfying performance when sequences are very short, *i.e.,* the *cold-start* issue [6]. We illustrate the sequence length distribution and the corresponding Recall@5 for next item prediction on Amazon Beauty dataset [13] in Figure 1. The transformers in both models are trained with the length being 100. The observations are twofold. Firstly, most of the sequences are rather short. Nearly 75% sequences consist of less than 7 items. Moreover, the performance of very short sequences (*e.g.*, length = 3) is much poorer than that of long sequences (*e.g.*, length ≥ 20). This discrepancy implies the necessity of informative contexts for encoding sequences [20, 21], which are limited in short sequences. Therefore, it motivates us to devise *lengthening augmentation for short sequences.*

The challenges for this augmentation are threefold. Firstly, since the target in sequential recommendation is to predict the next item, it is required to maintain the original sequential correlations after augmenting those short sequences. Additionally, though some recent works leverage item attributes [25, 31] as additional contexts, similar items might not reflect complex items transition correlations.

*Both authors contribute equally.

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3463036

Meanwhile, the fact that sequences are the only available data for most SRs further hinders the augmentation for short sequences.

To this end, we propose to augment those short sequences with pseudo-prior items. Intuitively, we append a fabricated sub-sequence of items at the beginning of short sequences, which provide additional contexts [3, 29]. To generate these pseudo-prior items, we pre-train a transformer from a reverse (i.e., right-to-left) direction of the original sequence to predict the prior item. As such, it can preserve the sequential correlations among these pseudo-prior items and the original items in sequences. Given the augmented sequences, we fine-tune the transformer from the original direction (i.e., left-to-right) to predict the next item in a sequence. We name this framework as **A**ugmenting **S**equential **Re**commendation with **P**seudo-prior items (ASReP). The contributions are as follows:

- To the best of our knowledge, we are the first work that investigates the possibility of improving the performance of SR by augmenting short sequences with pseudo-prior items.
- We design a novel framework ASReP that pre-trains over reverse-direction sequences and fine-tunes on augmented sequences.
- The ASReP significantly outperforms existing transformer-based SR models (*e.g.,* 18.1% for Recall@5). Detailed analyses verify the effectiveness of augmenting short sequences.

2 PRELIMINARY

2.1 Problem Definition

In SR problem, we denote the user set and item set as $\mathcal{U}$ and $\mathcal{V}$, respectively, whose user and item element are denoted as u and v, respectively. Each user is associated with a sequence of items $\mathcal{S}^u = [v_1^u, v_2^u, \ldots, v_{T_u}^u]$, which are in chronological order of the interaction time with the user u. $T_u = |\mathcal{S}^u|$, denoting the total number of items in the sequence of user u. The SR problem is commonly evaluated as the next-item prediction, which is formulated as:

$$p\left(v_{T_u+1}^{(u)} = v \,\middle|\, \mathcal{S}^u\right), \tag{1}$$

which is interpreted as calculating the probability of all candidate items, given the training sequence $\mathcal{S}^u$.

2.2 Embedding

We maintain an embedding table $\mathbf{E} \in \mathbb{R}^{d \times |\mathcal{V}|}$ for all the items, whose elements $\mathbf{e}_i \in \mathbb{R}^d$ denotes the embedding for item v_i. Besides, we should train embeddings for the position indices in a sequence. Since a transformer is trained with a fixed length sequence, we hold a position embedding table $\mathbf{P} \in \mathbb{R}^{d \times n}$, where n is the maximum length. Its element $\mathbf{p}_i$ denotes the position embedding for i-th position in a sequence. For a sequence $\mathcal{S}$, we truncate it to be the last n items if it is longer than n. And, we employ zero-padding [4] if it is shorter than n. Therefore, given a sequence $\mathcal{S} = [v_1, v_2, \ldots, v_n]$, the input embedding is

$$\mathbf{E}_{\mathcal{S}} = [\mathbf{e}_1 + \mathbf{p}_1, \mathbf{e}_2 + \mathbf{p}_2, \ldots, \mathbf{e}_n + \mathbf{p}_n]. \tag{2}$$

2.3 Transformer for SR

We illustrate the basic structure of a transformer [18] as in the Figure 2(a). It can encode a sequence as in Eq. (2) to an output embedding. To be more specific, a transformer consists of two components, *multi-head attention* and *feed-forward network*. The multi-head attention adopts scaled dot-product attention at each head to learn the importance of items in sequences. The multi-head attention on the input embeddings is:

$$\mathbf{H} = \text{concat}\{\text{head}_1, \text{head}_2, \ldots, \text{head}_h\}\mathbf{W}^O, \tag{3}$$

$$\text{head}_i = \text{Attention}(\mathbf{W}_i^Q \mathbf{E}_{\mathcal{S}}, \mathbf{W}_i^K \mathbf{E}_{\mathcal{S}}, \mathbf{W}_i^V \mathbf{E}_{\mathcal{S}}), \tag{4}$$

where the *Attention* is the scaled dot-product attention [4, 17, 18], and its input are query, key and value which are linearly mapped from $\mathbf{E}_{\mathcal{S}}$ with $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$ and $\mathbf{W}_i^V$, respectively. Then, we use a non-linear Feed-Forward Network (FFN) to position-wisely map $\mathbf{H}_i$ to the sequence embedding at position i. The FFN consists of two linear transformation layers with a ReLU activation function in between [4, 18]. A transformer layer is denoted as Trm. More details about the transformer for SR can be found in [4, 17, 22].

3 PROPOSED MODEL

In this section, we present the framework of ASReP, which consists of *reversely pre-training*, *short sequence augmentation*, and *left-to-right fine-tuning*. Since this paper's main purpose is to verify the efficacy of augmenting short sequences, we directly adopt the transformer backbone in SASRec [4] as the sequence encoder. The framework is illustrated in Figure 2(b).

3.1 Reversely Pre-training

To generate pseudo-prior items for short sequences, we train the transformer from the right-to-left reverse direction of sequences, which is presented in the left-hand side of Figure 2(b). As such, the transformer is able to predict the prior item of a sequence $\mathcal{S} = [v_1, v_2, \ldots, v_n]$ as:

$$p(v_0 = v \,|\mathcal{S}), \tag{5}$$

where $v_0 \in \mathcal{V}$ denotes the prior item of the sequence. In the implementation, we mask the left prior item and enforce the transformer to predict it, which is exactly the reverse direction of the training in SASRec. Note that, though the model is trained in a reverse direction, the self-attention module can also reveal the item correlations, which is why the previous work [17] endows a sequence encoder with a bi-directional transformer.

3.2 Short Sequences Augmentation

Next, we use the reversely pre-trained transformer to recursively generate pseudo-prior items. It involves two hyper-parameters:

- k, denoting the number of pseudo-prior items ahead the sequence.
- M, augmenting a sequence if its length $\leq M$.

We denote these pseudo-prior items as $[v_{-k+1}, \ldots, v_{-1}, v_0]$ and put them in the beginning of the original short sequence $\mathcal{S}$. We illustrate the augmentation for all short sequences as in the middle of Figure 2(b). Note that these pseudo-prior items are generated recursively. For example, we generate v_0 and append it ahead of $\mathcal{S}$, which constitutes a new sequence for the inference of v_{-1}.

3.3 Left-to-Right Fine-tuning

Finally, we fine-tune the transformer using the augmented sequences from the left-to-right direction to predict the next item. We illustrate this on the right-hand side of Figure 2(b). We use

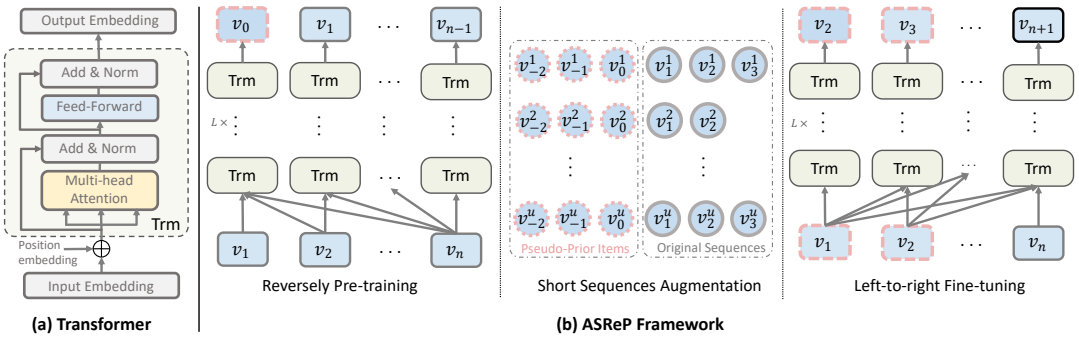

Figure 2: Transformer and ASReP framework. ASReP consists of three components: Firstly, *reversely pre-training* aims to predict prior items (pink dash blocks) of sequences. Secondly, *short sequences augmentation* is to generate k pseudo-prior items for short sequences ($M = 3$), where $k = 3$ in this figure. Finally, *left-to-right fine-tuning* is to train the model for next-item (v_{n+1}) prediction with augmented sequences.

the augmented sequence to predict the next item v_{n+1}. The first k items for short sequences are all pseudo-prior items, which are those pink dash boxes. Long sequences without augmentation have no pseudo-prior items. Note that we mask the right next item to maintain the causality and sequential transition correlations for the attention module in Trm layers. Again, since the transformer requires a fixed length of input, we truncate it to the last item if the sequence (both augmented and without augmentation) is longer than n, and zero-pad it if it is still shorter than n.

4 EXPERIMENTS

In this section, we present experimental settings and results. We will answer the following Research Questions (**RQs**):

- **RQ1**: Is ASReP effective in improving the performance of sequential recommendation?
- **RQ2**: What is the best choice of the number of pseudo-prior items (k) and the length of short sequences (M) of ASReP?
- **RQ3**: How does the ASReP perform with respect to the length of original sequences?

4.1 Datasets

In our experiments, we use two publicly available datasets [13]: (1) Beauty: Amazon Beauty 5-core includes 22,363 users (sequences), 12,101 items, and 198,502 ratings with density as 0.07%. The shortest sequence length is 5 while more than 75% of sequences are less than 9, and only 1,019 sequences are longer than 20. (2) Phones: Amazon Cell Phones and Accessories 5-core includes 27,879 users (sequences), 10,429 items, and 194,439 ratings with density as 0.06%. The shortest sequence length is 5 while more than 75% of sequences are less than 7, and only 284 sequences are longer than 20.

Following [4, 17], we transform datasets with explicit ratings into implicit feedbacks by treating the presence of a review as positive feedback, and use timestamps to determine the order of items within each sequence (per user). We use the most recent item of each user for testing and the second most recent item for validation.

Table 1: Performance Comparison in Recall@5, NDCG@5, and MRR. The best and second-best results are boldfaced and underlined, respectively.

	Beauty			Phones		
	Recall@5	NDCG@5	MRR	Recall@5	NDCG@5	MRR
BPRMF	0.3737	0.2712	0.2682	0.3862	0.2849	0.2831
LightGCN	0.3852	0.2927	0.2906	0.4498	0.3394	0.3218
SASRec	0.3963	0.2949	0.2907	0.4646	0.3379	0.3314
BERT4Rec	0.4143	0.3128	0.3098	0.5077	0.3812	<u>0.3720</u>
ItemCor	0.4053	0.3067	0.3039	0.4755	0.3526	0.3428
re-train	<u>0.4311</u>	<u>0.3257</u>	<u>0.3209</u>	<u>0.5213</u>	<u>0.3833</u>	0.3667
ASReP	**0.4684**	**0.3547**	**0.3458**	**0.5573**	**0.4193**	**0.4026**
v.s. SASRec	+18.1%	+20.2%	+18.9%	+19.9%	+24.0%	+21.4%
Improve.	+8.6%	+8.9%	+7.7%	+6.9%	+9.3%	+8.2%

4.2 Experimental Settings

Evaluation Protocols. We evaluate all models in three metrics: Recall@5, NDCG@5, and Mean Reciprocal Rank (MRR). Recall@5 measures the fraction of relevant items being retrieved at top-5 recommendations out of all relevant items, NDCG@5 evaluates their top-5 ranking performance, while MRR measures the ranking performance of the entire ranking list. For each user, we randomly sample 100 negative items for ranking with the ground-truth item.

Baselines. We compare ASReP with two transformer-based SR models SASRec [4] and BERT4Rec [17]. Also, we compare with two static models, the BPR-MF [15] and LightGCN [2]. Additionally, we create two variants of the ASReP. One is *ItemCor*, whose pseudo-prior items are items with similar embeddings from LightGCN. Hence, it augments short sequences without reversely training but only item correlations. The other one is *re-train*, which re-trains a new transformer over the augmented sequences rather than fine-tuning the reversely pre-trained transformer.

Parameter Settings. For all methods, we search the embedding size d from $\{32, 64, 128\}$. The $L2$ regularization term is selected from $\{0.0, 0.0001, 0.001, 0.01, 0.1\}$. For max input sequence length n, we search from $\{50, 100\}$ for all SR methods. We search the number of layers L from $\{1, 2, 3\}$. For BERT4Rec, we also search the masked probability from $\{0.2, 0.3, 0.5, 0.7\}$. We grid search all

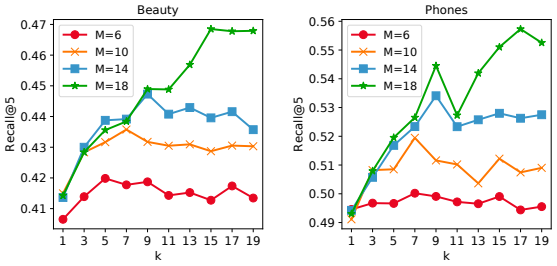

Figure 3: The Recall@5 performance on different values of M and k over two datasets.

possible combinations of hyper-parameters on the validation set. The best hyper-parameters set for Beauty dataset is max length $n = 100$, $d = 128$, $L = 2$, dropout rate is 0.7, $L2 = 0.0$, short sequence length threshold $M = 18$, and the number of augmenting items $k = 15$. For Phones dataset, it is max length $n = 100$, $d = 32$, $L = 2$, dropout rate is 0.5, $L2 = 0.0$, $M = 18$, and $k = 17$.

4.3 Overall Comparison (RQ1)

We report the overall performance comparison of ASReP and other baselines in Table 1. The observations are as follows:

- ASReP outperforms other baselines on all metrics. Compared with SASRec (v.s. SASRec), ASReP has significant relative improvements on all three metrics, in average 19.1% and 21.8% on Beauty and Phones, respectively. Compared with the second-best one (improvements are in the last row), it also has in average 8.4% and 8.1% improvements on Beauty and Phones dataset, respectively. These results prove that our framework is very effective in modeling item sequential correlations. And, the augmentation for short sequences is rather crucial.
- All sequential models are better than static methods, *i.e.,* BPR-MF and LightGCN, which verifies the efficacy of modeling item sequential correlations with transformer. However, they are also worse than ASReP as they suffer from the cold-start issue when predicting the next item for short sequences.
- The two variants of ASReP perform worse than ASReP. Compared with ASReP, *ItemCor* directly append similar items before short sequences as pseudo-prior items. The worse performance of it shows the necessity of reversely pre-training a transformer for generating pseudo-prior items. But it is still better than SASRec, which proves the benefits of augmenting short sequences contexts. Compared with ASReP, *re-train* re-trains a new transformer. It ignores the reversely pre-trained transformer and trains a new transformer for recommendation. Its worse performance compared with ASReP verifies that pre-training helps capture important sequential item correlations, which is also why BERT4Rec performs better than SASRec.

4.4 Parameter Sensitivity (RQ2)

We analyze the performance of ASReP w.r.t. its two hyper-parameters k and M, which denote the number of pseudo-prior items and the threshold for short sequences, respectively. We select $k \in \{1, 3, \ldots, 19\}$ and $M \in \{6, 10, 14, 18\}$. The results are illustrated in Figure 3. On both datasets, the performance improves when M increases, which verifies the importance of augmenting short sequences. Additionally, when M is small, *e.g.* 6, increasing k has little improvements. This is because those generated pseudo-prior items are not informative when we only consider very short sequences. Though performance improves when we increase both k and M, the time cost also increases. Therefore, we should find a trade-off between those hyper-parameters and efficiency.

4.5 Performance w.r.t. Sequence Length (RQ3)

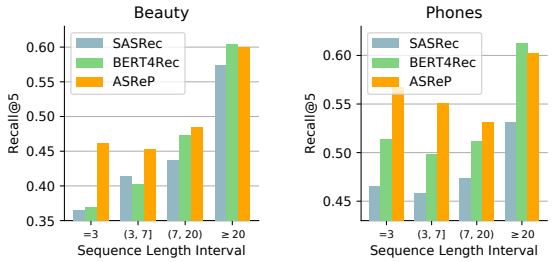

Figure 4: The Recall@5 performance on different sequence lengths over two datasets.

We illustrate the performance of SASRec, BERT4Rec and ASReP w.r.t. the sequence length in Figure 4. Firstly, we observe that ASReP significantly outperforms both SASRec and BERT4Rec when sequence length < 20, i.e., short sequences. This proves that ASReP is effective in augmenting short sequences and provides additional contexts to encode short sequences. Secondly, since we directly adopt the same fine-tuning strategy as SASRec, the better performance of ASReP compared with SASRec on all sequence lengths demonstrates the necessity of augmenting short sequences. Additionally, we observe that BERT4Rec performs the best when sequence length ≥ 20, which shows the benefits of a bi-directional transformer. However, since we focus on augmenting short sequences and most sequences are rather short, ASReP can thus significantly improve the overall performance. It is also worth noting that we could substitute the left-to-right fine-tuning of ASReP to bi-directional fine-tuning, which may also outperform BERT4Rec on long sequences. We leave it for future study.

5 CONCLUSIONS

In this paper, we study improving sequential recommendation via augmenting short sequences. To complete this task, we propose a new framework, ASReP, which employs a reversely pre-trained transformer to generate pseudo-prior items for short sequences. We fine-tune the pre-trained transformer from the left-to-right direction to predict the next item in a sequence. Moreover, we conduct overall comparisons of ASReP with other baselines, verifying the effectiveness of ASReP. Detailed analyses demonstrate that ASReP significantly improves the performance regarding short sequences.

6 ACKNOWLEDGEMENTS

This work is supported in part by NSF under grants III-1763325, III-1909323, and SaTC-1930941.

REFERENCES

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

[2] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 639–648. https://doi.org/10.1145/3397271.3401063

[3] Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. 2020. Sub-graph Contrast for Scalable Self-Supervised Graph Representation Learning. *arXiv preprint arXiv:2009.10273* (2020).

[4] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. IEEE, 197–206.

[5] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *KDD*. ACM.

[6] Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, Yang Yang, and Zi Huang. 2019. From zero-shot learning to cold-start recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4189–4196.

[7] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM*. 322–330.

[8] Xiaohan Li, Mengqi Zhang, Shu Wu, Zheng Liu, Liang Wang, and Philip S. Yu. 2020. Dynamic Graph Collaborative Filtering. In *ICDM*.

[9] Zhiwei Liu, Xiaohan Li, Ziwei Fan, Stephen Guo, Kannan Achan, and Philip S Yu. 2020. Basket Recommendation with Multi-Intent Translation Graph Neural Network. *arXiv preprint arXiv:2010.11419* (2020).

[10] Zhiwei Liu, Mengting Wan, Stephen Guo, Kannan Achan, and Philip S Yu. 2020. Basconv: aggregating heterogeneous interactions for basket recommendation with graph convolutional neural network. In *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 64–72.

[11] Zhiwei Liu, Lei Zheng, Jiawei Zhang, Jiayu Han, and S Yu Philip. 2019. JSCN: Joint spectral convolutional network for cross domain recommendation. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 850–859.

[12] Huanrui Luo, Ning Yang, and S Yu Philip. 2019. Hybrid Deep Embedding for Recommendations with Dynamic Aspect-Level Explanations. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 870–879.

[13] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. 43–52.

[14] Ruiyang Ren, Zhaoyang Liu, Yaliang Li, Wayne Xin Zhao, Hui Wang, Bolin Ding, and Ji-Rong Wen. 2020. Sequential recommendation with self-attentive multi-adversarial network. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 89–98.

[15] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (Montreal, Quebec, Canada) *(UAI '09)*. AUAI Press, Arlington, Virginia, USA, 452–461.

[16] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. 811–820.

[17] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*. 1441–1450.

[18] 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 *NIPS*. 5998–6008.

[19] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT: Sequential Recommendation Via Personalized Transformer. In *RecSys*. ACM, 328–337.

[20] Congying Xia, Caiming Xiong, S Yu Philip, and Richard Socher. 2020. Composed Variational Natural Language Generation for Few-shot Intents. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 3379–3388.

[21] Congying Xia, Chenwei Zhang, Hoang Nguyen, Jiawei Zhang, and Philip Yu. 2020. Cg-bert: Conditional text generation with bert for generalized few-shot intent detection. *arXiv preprint arXiv:2004.01881* (2020).

[22] Xu Xie, Fei Sun, Zhaoyang Liu, Jinyang Gao, Bolin Ding, and Bin Cui. 2020. Contrastive Pre-training for Sequential Recommendation. *arXiv preprint arXiv:2010.14395* (2020).

[23] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G Carbonell. 2010. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*. SIAM, 211–222.

[24] Liangwei Yang, Zhiwei Liu, Yingtong Dou, Jing Ma, and Philip S. Yu. 2021. ConsisRec: Enhancing GNN for Social Recommendation viaConsistent Neighbor Aggregation. *Proceedings of the 44th international ACM SIGIR conference on Research and development in information retrieval*.

[25] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Aditya Menon, Lichan Hong, Ed H Chi, Steve Tjoa, Evan Ettinger, et al. 2020. Self-supervised Learning for Deep Models in Recommendations. *arXiv preprint arXiv:2007.12865* (2020).

[26] Wenwen Ye, Shuaiqiang Wang, Xu Chen, Xuepeng Wang, Zheng Qin, and Dawei Yin. 2020. Time Matters: Sequential Recommendation with Complex Temporal Information. In *SIGIR*. 1459–1468.

[27] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *SIGKDD*, Yike Guo and Faisal Farooq (Eds.). 974–983.

[28] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140* (2020).

[29] Yao Zhang, Yun Xiong, Yun Ye, Tengfei Liu, Weiqiang Wang, Yangyong Zhu, and Philip S Yu. 2020. SEAL: Learning Heuristics for Community Detection with Generative Adversarial Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1103–1113.

[30] Lei Zheng, Ziwei Fan, Chun-Ta Lu, Jiawei Zhang, and Philip S Yu. 2019. Gated Spectral Units: Modeling Co-evolving Patterns for Sequential Recommendation. In *SIGIR*. 1077–1080.

[31] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1893–1902.

[32] Yao Zhou, Arun Reddy Nelakurthi, and Jingrui He. 2018. Unlearn what you have learned: Adaptive crowd teaching with exponentially decayed memory learners. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2817–2826.

[33] Yao Zhou, Jianpeng Xu, Jun Wu, Zeinab Taghavi Nasrabadi, Evren Korpeoglu, Kannan Achan, and Jingrui He. 2020. GAN-based Recommendation with Positive-Unlabeled Sampling. *arXiv preprint arXiv:2012.06901* (2020).