# Revisiting Self-Attentive Sequential Recommendation

**Zan Huang** *
huangzan@gatech.edu

**Abstract**

Recommender systems are ubiquitous in on-line services to drive businesses. And many sequential recommender models were deployed in these systems to enhance personalization. The approach of using the transformer decoder as the sequential recommender was proposed years ago and is still a strong baseline in recent works. But this kind of sequential recommender model did not scale up well, compared to language models. Quite some details in the classical self-attentive sequential recommender model could be revisited, and some new experiments may lead to new findings, without changing the general model structure which was the focus of many previous works. In this paper, we show the details and propose new experiment methodologies for future research on sequential recommendation, in hope to motivate further exploration to new findings in this area.

## 1 Problem Setting

Given the user set $U = \{u_1, u_2, ..., u_m\}$, the item set $I = \{i_1, i_2, ..., i_n\}$, and the interaction history represented by the $(u_x, i_y, t_z)$ records. The target of recommender models is to accurately predict the best-fit item $i_y^*$ when $u_x$ requests for service, so $i_y^*$ could be actively and better served to maximize the user's or the platform's interest.

Sequential recommenders work in a further simplified setting, by sorting user-interacted items into a sequence $S_{u_x} = [i_{x_1}, i_{x_2}, ..., i_{x_l}]$ of length $l$ according to the interaction timestamp. And try to predict $i_y^*$ based on $S_{u_x}$.

In the offline experiment, we can hide part of the interaction history and take $i_{hidden}$ as $i^*$ to train or evaluate the recommender model. Different metrics can be applied in this phase to guide the model to effectively uncover $i_{hidden}$.

In the online experiment, the performance of the recommender model would be evaluated systematically by A/B testing. Under the assumption that good $i^*$ predictions lead to more interactions or other observable platform growth signals.

## 2 Related Work

SASRec[1] was proposed following Transformer[2] in which the decoder structure could be used for sequential recommendation, by replacing tokens with item IDs. Then BERT4Rec[3] following BERT[4] used the encoder for sequential recommendation and claimed to achieve a better result.

More recent research shows that SASRec could be further improved by elaborating details such as the loss function [5], and mask usage [6], making it a strong long-lived baseline.

Referring to the current trend of applying large language models on the recommender system, the scalability issue of SASRec was paid more attention. HSTU[7] try to resolve the issue by customizing the attention layers for the recommendation task, HLLM[8] try to use pre-trained language model directly in recommendation.

Both of these approaches are promising, as language models have been used extensively in industry recommender systems for years, and attention layer customization helps to better fit the data set and utilize the compute resource.

Recommender systems were successfully scaled out to hyperscale intelligent services over the years, mainly by trade-off storage resources for better personalization. But why did recommender models like SASRec fail in scaling up by trade-off compute for more precise prediction has not been well addressed, which is the focus of this work.

*Doing research on personal devices in sparse time, for internet users welfare, opinions are on my own.

## 3 The Details

In this section, we present some details of the self-attentive sequential recommender model to lay the foundation for further exploration and explain why we propose new experiments to revisit this paradigm.

### 3.1 Personalization

There is no personalization in the original SASRec[1] model, as it is without explicit use of user embeddings. The claim was that the user could be well represented by the interaction history. But in fact, anyone who interacts with $[i_x, i_y]$ would receive the same recommended item $i_z$ deterministically, if without the randomness introduced in sampling.

This setting may force the model to prefer the common choice instead of a more personalized choice for users, as it is still a non-personalized recommendation algorithm, but in a higher-dimensional sequential space to perform a "dimensional collapse strike". (Items are well represented by embeddings, as long as we support embeddings CRUD in the vector database to track item set dynamics. While users are more like random walkers in the item space, SASRec better captured the user-side dynamics compared to the approaches do vanilla user-item embedding distance computation. Something like the fourier transform may be needed to bridge user embeddings and item embeddings and preserve the dynamics considering the relationship like the one between the time domain and the frequency domain. )

Learning *average face* by the transformer-based model may make something beautiful enough and quite useful in fields such as language processing, but may limit the effectiveness of recommendation due to the lack of personalization and fail the scaling up attempts when stacking more self-attention layers.

### 3.2 Recommendation

Unlike other tasks, the data used in recommender model training is not the ground-truth, but the output generated when conditioned on the baseline model, by which I mean the interacted item may be one of the recommended items by the baseline model instead of a random pick. So, the offline training mainly shows the model's data-fitting capability. And personalization could be used as an interchangeable term of overfitting sometimes for the recommendation task.

Moreover, for the sequential recommendation task, turning a movie rating dataset into which-movie-to-be-rated-nextly dataset may be using it in an unexpected way. We should double-check these details before trying to scale up models.

### 3.3 Embedding

Learning the embeddings for recommender models is like adjusting the item positions in a high-dimensional supermarket. And the Transformer[2] model feels like an embedding factory, with self-attention layers being production lines.

In SASRec[1], there are two kinds of embedding used: item embedding $e_i$ and positional embedding $e_p$. There were some issues with embedding use in the PyTorch implementation of the model[2]. When the embedding values of the padding item are not initialized as $0$, it may work more like a bias term instead of padding. Positional embeddings added to padding items may also be mistakenly used to have *nothing* incorrectly contributing *something* to the final output. These issues affect the model performance but are sometimes hidden by randomness in negative sampling.

More importantly, there seems to be something wrong in the SASRec[1] use of positional embedding. The position of the interacted item should be relative to the position of the prediction, but the absolute position was used instead.

More concretely, given the interaction history represented by a sequence $S = [i_{x_1}, i_{x_2}, i_{x_3}, ..., i_{x_l}]$ of length $l$, there are $l-1$ items to be masked for training, predicting $i_y$ given sequence $[i_{x_1}]$ and label $i_{x_2}$, predicting $i_y$ given sequence $[i_{x_1}, i_{x_2}]$ and label $i_{x_3}$, lastly predicting $i_y$ given sequence $[i_{x_1}, i_{x_2}, ..., i_{x_{l-1}}]$ and label $i_{x_l}$.

When turning the input sequences into embedding sequences, we expect transformations like $[i_{x_1}] \rightarrow [(e_{i_{x_1}} + e_{p_1})]$, $[i_{x_1}, i_{x_2}] \rightarrow [(e_{i_{x_1}} + e_{p_2}), (e_{i_{x_2}} + e_{p_1})]$, ..., $[i_{x_1}, i_{x_2}, ..., i_{x_{l-1}}] \rightarrow [(e_{i_{x_1}} + e_{p_{l-1}}), (e_{i_{x_2}} + e_{p_{l-2}}), ..., (e_{i_{x_{l-1}}} + e_{p_1})]$.

But the trick was used in SASRec to pack the whole sequence into a batch of training samples using masks for causality in self-attention layers, causing the positional embedding use not following the expected way except for the last position prediction, e.g. $[i_{x_1}, i_{x_2}, ..., i_{x_{l-2}}] \rightarrow [(e_{i_{x_1}} + e_{p_{l-2}}), (e_{i_{x_2}} + e_{p_{l-3}}), ..., (e_{i_{x_{l-2}}} + e_{p_1})]$ was expected for the $i_{x_{l-1}}$ prediction, but $[i_{x_1}, i_{x_2}, ..., i_{x_{l-2}}] \rightarrow [(e_{i_{x_1}} + e_{p_{l-1}}), (e_{i_{x_2}} + e_{p_{l-2}}), ..., (e_{i_{x_{l-2}}} + e_{p_2})]$ was used. This problem may decrease the effectiveness of positional embedding and cause disorders in the learning procedure.

These details show the room for better positional embedding use of SASRec, corrections may help to further scale it up.

[2] https://github.com/pmixer/SASRec.pytorch

## 4 Proposed Experiments

According to the observed details presented. Here are some experiments worth trying, to fix related issues, and further improve or scale-up the SASRec-like model. We list the proposed experiments as in the following sections.

### 4.1 Correction

As claimed before, the positional embedding practice needs to be corrected and could be further improved, which may require reorganizing the training samples, e.g. batching all $length = 1$ sequences together, then all $length = 2$ sequences and so on, to make sure the learned positional embedding got applied in the correct way semantically.

The padding operation and the truncation operation could also be removed after the modification, to make better use of sequences, though it may require more advanced Transformer engineering practice to ensure efficiency.

Moreover, as items in the sequences are just hidden instead of being unknown in the model training phase, the prediction accuracy or the self-attention layer generated embedding quality of known items in a sequence could be used for adjusting the weights for the next item prediction or constructing more advanced positional embedding techniques.

### 4.2 Autoregression

Although sequential recommender models are designed for next item prediction tasks, they could be used to do predictions in the autoregressive way, like the practice in language or speech research.

When longer outputs are generated, the more we know about the model's characteristics, so to further optimize it. The previous practice of leaving the last item for the test set and the second last item for validation is relatively conservative.

### 4.3 Tokenization

The structure of the model could be shaped by the task and the data used. Although the SASRec model was based on the Transformer model developed for language processing, the dataset used in language model training and sequential recommender model training have not been thoroughly compared according to our survey.

Considering the simplified setting in sequential recommendation, the comparison of token sequences and item-id sequences can be performed to study the dataset differences in two different domains. We expect the item-id sequences to be more random, and much more sparse than language token sequences, challenging the model learning capability in a different way, and may require *the reverse process of tokenization*(taking tokenization as turning combination of basic items to new items) in production level recommender systems to breakdown and reduce trillions of items into combinations of more basic elements to really scale up the model, while it may suffer "ADHD" otherwise.

### 4.4 Duality

Based on *"you are what you interacted"* assumption, if the user can be well represented by a sequence of items, encoded by another model generated embedding from a sequence of item embeddings, we can also try to use a sequence of these user embeddings to represent the item, and enforce the high-order item embeddings to be consistent with the original item embedding, to improve the consistency and robustness of embeddings and models.(If you have ever tried machine translators to convert the text between two languages back and forth to see how the outputs gradually out of control, that is what I mean by the need of consistency and robustness.)

We may also start from user embeddings, try to use user sequences to represent items, and predict the next user for an item for interaction prediction. It would taste more like an advertisement system, to recommend users to items.

### 4.5 Sampling

The negative sampling used in SASRec introduced the randomness into the model training and inference process, but it is also somehow biased, as interacted items got excluded from the pool, and the ground-truth item was manually inserted into the pool. We could try more sampling algorithms for items retrieval, referring to the practice in industry.

For user-generated content datasets, we could even try to apply the constraint like *"how many items you interacted, how many times your items get recommended"*, to check the balance between fairness and business.

## 5 Future Work

We need to do these experiments systematically and may publish the work named *Scaling Up Self-Attentive Sequential Recommenders* if it goes well. Please feel free to do your own exploration based on the presented stuff if interested.

## References


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

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[3] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1441–1450, New York, NY, USA, 2019. Association for Computing Machinery.

[4] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 6. long and short papers: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2019), 2-7 June 2019, Minneapolis, Minnesota, USA*, 2019.

[5] Anton Klenitskiy and Alexey Vasilev. Turning dross into gold loss: is bert4rec really better than sasrec? In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23, page 1120–1125, New York, NY, USA, 2023. Association for Computing Machinery.

[6] Huiyuan Chen, Yusan Lin, Menghai Pan, Lan Wang, Chin-Chia Michael Yeh, Xiaoting Li, Yan Zheng, Fei Wang, and Hao Yang. Denoising self-attentive sequential recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, page 92–101, New York, NY, USA, 2022. Association for Computing Machinery.

[7] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, Yinghai Lu, and Yu Shi. Actions speak louder than words: trillion-parameter sequential transducers for generative recommendations. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[8] Junyi Chen, Lu Chi, Bingyue Peng, and Zehuan Yuan. Hllm: Enhancing sequential recommendations via hierarchical large language models for item and user modeling. *arXiv preprint arXiv:2409.12740*, 2024.