SASRec: A PyTorch Reimplementation and Ablation Study on MovieLens-1M

Kevin Bretz[1], Michail Protonotarios[1], and Maftukhatul Q. Virati[1]

Leiden University, Leiden, The Netherlands

Abstract. We rebuild SASRec [1] from scratch in PyTorch and run it on MovieLens-1M with a leave-one-out protocol and *full-corpus ranking* which are for every held-out target we score it against every other item the user has not yet interacted with. Our default configuration reaches NDCG@10 = 0.1008 and Recall@10 = 0.1929 on the test set. A controlled sweep over the four dimensions (number of blocks, hidden size, number of heads and maximum sequence length) shows that hidden size is the only direction in which extra capacity still pays off in our setup, that depth saturates almost immediately and that truncating user histories is by far the most damaging change.

Keywords: Sequential recommendation · Self-attention · Transformer · MovieLens.

1 Introduction

Sequential recommendation tries to predict what next item a user will interact with based on their chronology of order history. Early work used first-order Markov chains (FPMC [9]), which were eventually replaced by recurrent (GRU4Rec [7]) and convolutional (Caser [8]) architectures. SASRec [1] took a different route and transplanted the causal self-attention of the Transformer [2] into the setting, so that every position can attend adaptively to any earlier item; BERT4Rec [10] later generalised this with bidirectional masking. Our goal is to reimplement SASRec in PyTorch, run it on MovieLens-1M under the leave-one-out protocol the assignment specifies and compare the four architectural dimensions for ablation.

2 Methodology

2.1 Data preprocessing and sequence construction

We treat any MovieLens-1M rating of four or five stars as a positive feedback signal, after that we order each user's interactions based on the timestamp (break ties by movie id), drop users with fewer than five movie interactions and remap users and items to contiguous ids with 0 reserved everywhere as the padding

token. A *leave-one-out* split per user then gives us a training sequence (all but the last two interactions), a validation target (the penultimate interaction) and a test target (the final interaction). We also make sure that the split is applied to positive interactions only. From the data preprocessing and sequence construction we have a total of 6,034 users, 3,533 items and 575,272 positive interactions, with the average of sequence length of around 95. Comparing the results to the paper [1], they report roughly 1.0M interactions on the same raw dataset, this difference is caused by our stricter rating rules.

2.2 Model

Let d denote the latent dimensionality and n the maximum sequence length. The model has two embedding tables, an item embedding $\mathbf{M} \in \mathbb{R}^{|\mathcal{I}| \times d}$ that is shared between input and output and a learnable positional embedding $\mathbf{P} \in \mathbb{R}^{n \times d}$. For a left-padded input the representation at position t is $\widehat{\mathbf{E}}_t = \sqrt{d}\,\mathbf{M}_{s_t} + \mathbf{P}_t$. The sequence then passes through b stacked self-attention blocks, each of which applies the scaled dot-product attention of Vaswani et al. [2],

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\Big(\mathbf{Q}\mathbf{K}^\top / \sqrt{d_h}\Big)\,\mathbf{V}, \tag{1}$$

with $d_h = d/h$ per head, followed by a point-wise feed-forward $\mathrm{FFN}(\mathbf{x}) = \mathbf{W}_2\mathrm{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$ with dropout. A single boolean mask combines the causal lower-triangular constraint with key-padding so that attention can only flow to past, non-padding tokens. Both sub-layers use a pre-LayerNorm residual, $\mathbf{x} \leftarrow \mathbf{x} + \mathrm{Dropout}(g(\mathrm{LayerNorm}(\mathbf{x})))$, and after each one we explicitly zero the rows for padding positions so their activations cannot leak into the LayerNorm statistics of the next block. To score candidate items we reuse the same embedding matrix, $r_{i,t} = \mathbf{F}_t^{(b)\top}\mathbf{M}_i$. This weight tying halves the item-dependent parameter count and acts as a strong regulariser.

2.3 Training and evaluation

At every non-padded position we draw one negative item j uniformly from the items the user has never interacted with and minimise the masked binary cross-entropy

$$\mathcal{L} = -\sum_{(u,t)} \Big[\log \sigma(r_{o_t,t}) + \log(1 - \sigma(r_{j,t}))\Big], \tag{2}$$

using Adam [3] with $\beta = (0.9, 0.98)$, batch size 128, dropout 0.2 and at most 200 epochs. The learning rate follows a linear-warmup-then-cosine-decay schedule (ramping from 0 to 10^{-3} over 10 epochs, then decaying smoothly to 0). We validate every 20 epochs, keep the checkpoint with the best validation NDCG@10 and stop training if that metric fails to improve for 40 consecutive epochs. For evaluation we follow the assignment specification and use *full-corpus ranking*, for each held-out target we score every item in the catalogue against the user's encoded sequence in a single matrix multiplication $\mathbf{F}_n^{(b)}\mathbf{M}^\top$, mask out items

the user has already interacted with (so they cannot outscore the target) and report the rank of the held-out target among the surviving candidates. This is harder than the sampled-negative protocol used in the original paper as the candidate set is now the entire item catalogue of $\approx$ 3,533 items rather than 101 but the absolute numbers are directly meaningful and comparable across configurations. We report Recall@$K = \mathbb{1}[\text{rank} < K]$ and NDCG@$K = \mathbb{1}[\text{rank} < K]/\log_2(\text{rank} + 2)$ for $K \in \{10, 20\}$, averaged over users.

3 Experiments

3.1 Setup

The default hyperparameters in Table 1 are based on the original paper's [1] MovieLens-1M settings.

Table 1: Default hyperparameters.

max sequence length n	200	peak learning rate	10^{-3}
hidden dim d	50	batch size	128
# blocks b	2	max epochs	200
# heads h	1	early-stop patience	40 epochs
dropout	0.2	evaluation cadence	every 20 epochs
optimiser	Adam, $\beta = (0.9, 0.98)$		
LR schedule	linear warmup (10 epochs) $\rightarrow$ cosine decay		

3.2 Main results

Our default configuration reaches NDCG@10 = 0.1008 and Recall@10 = 0.1929 on the test set (Table 2). We did not do direct comparison as Kang and McAuley evaluate against 100 sampled negatives per user, whereas under full-corpus ranking each target is ranked against the entire catalogue, which is a much harder task. This result to drop of the value substantially at approximately 5$\times$ on NDCG@10 relative to the paper's 0.5905 but relative comparisons between configurations (Section 3.3) remain reliable because every run scores against exactly the same candidate set.

Table 2: Main results on MovieLens-1M (test set, full-corpus ranking).

Model	NDCG@10	NDCG@20	Recall@10	Recall@20
SASRec (this work, default)	0.1008	0.1251	0.1929	0.2887

3.3 Ablation across the four required dimensions

We vary one architectural dimension (number of blocks b, hidden size d, number of heads h and maximum sequence length n) at a time and leave everything else at the default. Since every run ranks against the same full item catalogue with each user's own history masked out identically, the gaps in Table 3 reflect

real architectural effects rather than noise from a different choice of candidates. Figure 1 plots the resulting NDCG@10 values grouped by ablation dimension.

Table 3: Ablation study on MovieLens-1M (test set, full-corpus ranking). Each row changes only one hyperparameter relative to the default. **Bold** marks the best value per column.

Configuration	NDCG@10	NDCG@20	Recall@10	Recall@20
default (b=2, d=50, h=1, n=200)	0.1008	0.1251	0.1929	0.2887
$b = 1$	0.1026	0.1271	0.1961	0.2938
$b = 3$	0.1027	0.1286	0.1990	0.3021
$d = 32$	0.0918	0.1162	0.1777	0.2758
$d = 64$	**0.1047**	**0.1299**	**0.2043**	**0.3046**
$h = 2$	0.0976	0.1218	0.1904	0.2865
$n = 50$	0.0901	0.1145	0.1770	0.2739

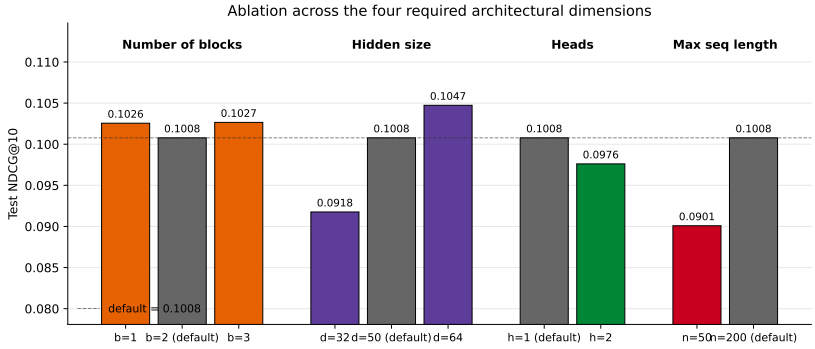

Fig. 1: Test NDCG@10 across the four required ablation dimensions. For each group, grey is the default configuration as reference and the dashed line marks its NDCG@10. The depth and hidden-size sweeps saturate near the default, h=1 $\leftrightarrow$ h=2 is essentially a tie and changing the history (n=50) is by far the most worst change.

Interpretation. Depth is essentially flat. NDCG@10 reads 0.1026/0.1008/0.1027 for b=1, 2, 3, all within seed noise, so even a single block captures the relevant sequential structure. *Hidden size* still has room to grow. d=32 costs −8.9% on NDCG@10 and d=64 *outperforms* the d=50 default by +3.9%, winning every column of Table 3. Under full-corpus ranking d=50 is no longer a sweet spot, likely because full ranking exposes finer-grained item distinctions than the sampled-101 setting the paper tuned d=50 for. *Number of heads* hurts here. h=2 drops NDCG@10 by 3.2%; splitting d=50 into two 25-dim subspaces leaves each head too narrow to learn a distinct attention pattern. *Maximum sequence length* is by far the most consequential knob. n=50 costs 10.6% on NDCG@10 and 8.2% on Recall@10, because the average user history is $\approx$ 95 items and that trunca-

tion strips many users of their early interactions. This long-context sensitivity is exactly what was supposed to separate SASRec from the first-order FPMC.

4 Discussion and Challenges

A few small implementation choices mattered more than we expected. We pad sequences on the *left* so the inference feature is always $\mathbf{F}_n^{(b)}$ and the scoring code can read the final position unconditionally. The causal and key-padding masks are merged into one tensor handed to `F.scaled_dot_product_attention`, sidestepping the version-dependent mask semantics of `nn.MultiheadAttention`. After each residual we zero the padding rows so they cannot drift into the LayerNorm statistics of the next block. The item embedding is shared between input and scoring head (the paper reports $> 20\%$ NDCG@10 loss without tying), which also lets us score every item in a single matmul $\mathbf{F}_n^{(b)}\mathbf{M}^\top$ at evaluation time which are crucial for keeping full-corpus ranking tractable. We mask the user's training history from the candidate set so already-seen items cannot outscore the target.

Limitations and future work. Our absolute numbers sit well below Kang and McAuley's figures, but this is expected as full-corpus ranking is fundamentally harder than their 101-candidate sampled protocol. All numbers come from a single seed and a more accurate study would average over several. The natural next step is a head-to-head comparison with BERT4Rec [10] under the same protocol.

5 Conclusion

We rebuilt SASRec in PyTorch and ran it on MovieLens-1M with leave-one-out splits and full-corpus ranking. The default reaches NDCG@10 = 0.1008 and Recall@10 = 0.1929. The ablations show that depth saturates almost immediately, hidden size still has room to grow (d=64 beats the default on every metric), a second attention head hurts at this hidden size and truncating histories to n=50 costs the most by a wide margin which confirms SASRec's reliance on long context.

References

1. Kang, W.-C., McAuley, J.: Self-Attentive Sequential Recommendation. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), pp. 197–206 (2018). https://doi.org/10.1109/ICDM.2018.00035
2. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I.: Attention Is All You Need. In: Advances in Neural Information Processing Systems (NeurIPS) 30, pp. 5998–6008 (2017). http://arxiv.org/abs/1706.03762

3. Kingma, D. P., Ba, J.: Adam: A Method for Stochastic Optimization. In: Proceedings of the 3rd International Conference on Learning Representations (ICLR) (2015). https://arxiv.org/abs/1412.6980
4. Ba, J. L., Kiros, J. R., Hinton, G. E.: Layer Normalization. arXiv preprint arXiv:1607.06450 (2016). https://arxiv.org/abs/1607.06450
5. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research **15**(56), 1929–1958 (2014). http://jmlr.org/papers/v15/srivastava14a.html
6. Harper, F. M., Konstan, J. A.: The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems **5**(4), Article 19 (2015). https://doi.org/10.1145/2827872
7. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based Recommendations with Recurrent Neural Networks. In: Proceedings of the 4th International Conference on Learning Representations (ICLR) (2016). https://arxiv.org/abs/1511.06939
8. Tang, J., Wang, K.: Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In: Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM), pp. 565–573 (2018). https://doi.org/10.1145/3159652.3159656
9. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing Personalized Markov Chains for Next-Basket Recommendation. In: Proceedings of the 19th International Conference on World Wide Web (WWW), pp. 811–820 (2010). https://doi.org/10.1145/1772690.1772773
10. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., Jiang, P.: BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM), pp. 1441–1450 (2019). https://doi.org/10.1145/3357384.3357895