Scaling Session-Based Transformer Recommendations using Optimized Negative Sampling and Loss Functions

Timo Wilm [1] Philipp Normann [1] Sophie Baumeister [1] Paul-Vincent Kobow [1]

[1]OTTO (GmbH & Co KG), Hamburg, Germany

Abstract

This work introduces **TRON**, a scalable session-based **T**ransformer **R**ecommender using **O**ptimized **N**egative-sampling. Motivated by the scalability and performance limitations of prevailing models such as SASRec and GRU4Rec$^+$, TRON integrates top-k negative sampling and listwise loss functions to enhance its recommendation accuracy and overall training time. Evaluations on relevant large-scale e-commerce datasets show that TRON improves upon the recommendation quality of current methods while maintaining training speeds similar to SASRec. A live A/B test yielded an 18.14% increase in click-through rate over SASRec, highlighting the potential of TRON in practical settings.

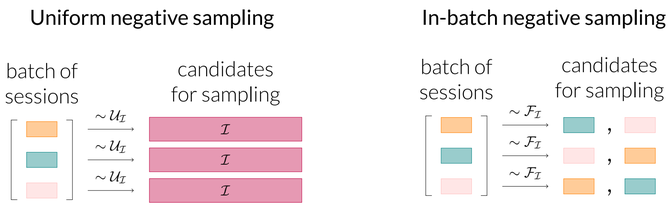
What is TRON?

TRON improves upon SASRec, enhancing scalability and performance for large scale ecommerce datasets. The enhancements include:

1. **Negative sampling**: uniform and in-batch
2. Backpropagate only on **top-k negatives**
3. Loss functions: adopt **sampled softmax**

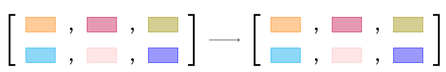
1. Negative Sampling: Uniform and In-batch

We use a combination of negative sampling from a uniform distribution $\mathcal{U}_\mathcal{I}$ and an empirical density distribution $\mathcal{F}_\mathcal{I}$ over the item set $\mathcal{I}$. Consider a batch $\mathcal{B} := [S_1, S_2, \ldots, S_b]$ that consists of b user sessions. At each time step t in each user session s, we sample $\mathcal{UN}_s^t := [U_1, U_2, \ldots, U_k]$ and $\mathcal{FN}_s^t := [F_1, F_2, \ldots, F_m]$. These samples are then concatenated to form a $k+m$ dimensional random vector $\mathcal{N}_s^t := concat[\mathcal{FN}_s^t, \mathcal{UN}_s^t]$. The negative samples for the entire batch are represented as $\mathcal{N}$.



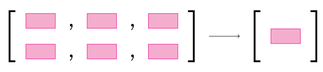
This sampling process can be performed in different ways, which results in a different tensor shape of negatives with significant impact on data transfers between CPU and GPU:

- **elementwise** negatives per element $shape(\mathcal{N}) = [b, T, k+m]$



- **sessionwise** negatives per session $shape(\mathcal{N}) = [b, 1, k+m]$

- **batchwise** negatives per batch $shape(\mathcal{N}) = [1, 1, k+m]$



2. Backpropagate only on Top-K Negatives

Additionally, we sample a set of negative items $\mathcal{N}_s^t$ and obtain scores $r_{s,j}^t$ for each item j of session s at time step t in $\mathcal{N}_s^t$. Applying the top-k function to the scored items, we select the top-k negatives $\mathcal{KN}_s^t := topk([r_{s,1}^t, r_{s,2}^t, \ldots, r_{s,|\mathcal{N}_s^t|}^t])$. These top-k items are then used for updates in the backpropagation step, while the rest are discarded.

3. Loss Functions: Adopt Sampled Softmax

For our experiments we use the following three loss functions, where TRON utilizes the sampled softmax loss:

- Pointwise loss: **Binary cross-entropy (BCE)** [3]
- Pairwise loss: **Bayesian personalized ranking max (BPR)** [2]
- Listwise loss: **Sampled softmax (SSM)** [1]

Experimental Study On Public Datasets

We only use click events for our experiments, maintaining a minimum item support of five and a session length of at least two for all datasets [2]. We use a temporal train/test split method, using the last day (Yoochoose) or the last week of data (Diginetica and OTTO) to form the test sets. The remaining data is used for training.

Table 1. Statistics of the datasets used in our experiments.

Data	Train set sessions	Train set events	Test set sessions	Test set events	items
Yoochoose	7,9M	31,6M	15k	71k	37k
OTTO	12,9M	194,7M	1,6M	12,3M	1,8M

SASRec and TRON models were trained for 10 epochs for both Yoochoose and OTTO, while GRU4Rec$^+$ was trained for 3 epochs on Yoochoose, and 1 epoch on OTTO. The results are shown in table 2.

Table 2. Recall@20 and training speed are reported using various negative sampling strategies and loss functions. The best result for each dataset is highlighted in bold. b: batchwise, s: sessionwise, e: elementwise.

Method	Loss	# Negatives uniform	# Negatives in-batch	Yoochoose R@20	Yoochoose Epochs/h	OTTO R@20	OTTO Epochs/h
GRU4Rec$^+$	BPR	2048_b	31_s	0.725	0.478	0.443	0.019
SASRec	BCE	1_e	0	0.573	2.573	0.307	**0.248**
SASRec M-Negs	BCE	512_s	16_s	0.607	**2.603**	0.269	0.246
SASRec L-Negs	BCE	8192_s	127_s	0.571	1.245	0.226	0.204
SASRec BPR-Max	BPR	8192_s	127_s	0.722	1.049	0.377	0.194
SASRec SSM	SSM	8192_s	127_s	0.722	1.268	0.432	0.209
TRON L-Negs	SSM	8192_b	127_s	0.730	2.117	0.460	0.233
TRON XL-Negs	SSM	16384_b	127_s	**0.732**	1.912	**0.472**	0.227
TRON XL vs. SASRec				27.7%	-25.7%	53.7%	-8.5%
TRON XL vs. GRU4Rec$^+$				0.97%	299.8%	6.5%	1094.7%

Real-World Evaluation of TRON

We trained TRON XL-Negs, SASRec SSM, and SASRec on a private OTTO dataset from the two most recent weeks. The Recall@20 for each epoch and model on the test set can be seen in Figure 1. The live improvement of TRON XL-Negs and SASRec SSM relative to SASRec measured from May 9 to May 17 2023 is shown in Figure 2.

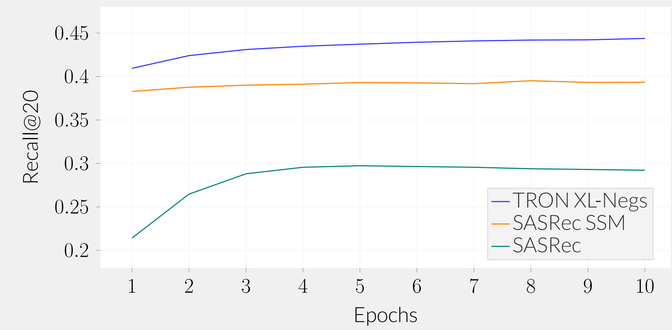


Figure 1. Offline evaluation results on our private OTTO dataset used for the online A/B test of our three groups.

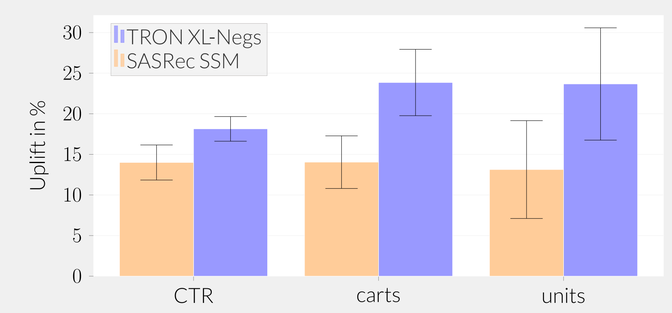


Figure 2. Online results of our A/B test relative to the SASRec baseline. The error bars indicate the 95% confidence interval.

Conclusion

Our proposed TRON model significantly improves the accuracy and training time of transformer-based recommendation systems on large e-commerce datasets. This enhancement is achieved through the strategic optimization of negative sampling methods, utilization of listwise loss functions, and focusing on the most misranked negatives. The results validate the effectiveness of TRON in a real-world e-commerce setting, showing an increase of 18.14% in click-through rate, 23.85% increase in add-to carts and 23.67% uplift in units compared to SASRec.

References

[1] Y. Bengio and J.-S. Senecal. Adaptive Importance Sampling to Accelerate Training of a Neural Probabilistic Language Model. *IEEE Transactions on Neural Networks*, 19(4):713–722, April 2008.

[2] Balázs Hidasi and Alexandros Karatzoglou. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 843–852, October 2018. arXiv:1706.03847 [cs].

[3] Wang-Cheng Kang and Julian McAuley. Self-Attentive Sequential Recommendation. 2018.

https://doi.org/10.1145/3604915.3610236 Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, Singapore

timo.wilm@otto.de, philipp.normann@otto.de, sophie.baumeister@otto.de