Session-based Recommendation with Transformers

Yichao Lu
Layer 6 AI
Toronto, Canada
yichao@layer6.ai

Zhaolin Gao*
Layer 6 AI
Toronto, Canada
zhaolin.gao@mail.utoronto.ca

Zhaoyue Cheng*
Layer 6 AI
Toronto, Canada
joey@layer6.ai

Jianing Sun*
Layer 6 AI
Toronto, Canada
jianing@layer6.ai

Bradley Brown
Layer 6 AI
Toronto, Canada
bcabrown@uwaterloo.ca

Guangwei Yu
Layer 6 AI
Toronto, Canada
guang@layer6.ai

Anson Wong
Layer 6 AI
Toronto, Canada
anson@layer6.ai

Felipe Pérez
Layer 6 AI
Toronto, Canada
felipe.perez.ds@gmail.com

Maksims Volkovs
Layer 6 AI
Toronto, Canada
maks@layer6.ai

ABSTRACT

Large item catalogs and constantly changing preference trends make recommendations a critically important component of every fashion e-commerce platform. However, since most users browse anonymously, historical preference data is rarely available and recommendations have to be made using only information from within the session. In the 2022 ACM RecSys challenge, Dressipi released a dataset with 1.1 million online retail sessions in the fashion domain that span an 18-month period. The goal is to predict the item purchased at the end of each session. To simulate a common production scenario all sessions are anonymous and no previous user preference information is available. In this paper, we present our approach to this challenge. We leverage the Transformer architecture with two different learning objectives inspired by the self-supervised learning techniques to improve generalization. Our team, LAYER 6, achieves strong results placing 2'nd on the final leaderboard out of over 300 teams.

CCS CONCEPTS

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

KEYWORDS

Recommender Systems, Deep Learning, Session-based Recommendation

ACM Reference Format:
Yichao Lu, Zhaolin Gao, Zhaoyue Cheng, Jianing Sun, Bradley Brown, Guangwei Yu, Anson Wong, Felipe Pérez, and Maksims Volkovs. 2022. Session-based Recommendation with Transformers. In *RecSys Challenge 2022 (RecSysChallenge22), September 18–23, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3556702.3556844

*Authors contributed equally to this work.

RecSysChallenge22, September 18–23, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9856-5/22/09…$15.00
https://doi.org/10.1145/3556702.3556844

1 INTRODUCTION

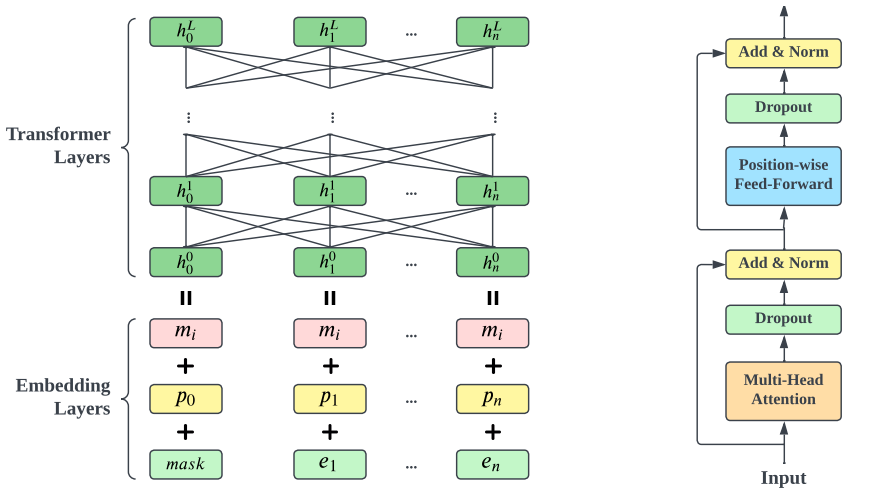
With rapidly changing preference trends and mostly anonymous browsing, the fashion industry presents a significant challenge to recommender systems. To properly address these difficulties, it is critically important to develop a recommender system that can deliver accurate recommendations using only information within the session.

The 2022 ACM RecSys challenge [1], organized by Dressipi, brings these challenges to light. Data on user sessions together with corresponding product information is provided, and the goal is to predict the purchased item at the end of each session. The dataset contains 1.1 million online user sessions over the 18-month period, and all sessions are anonymous with no previous user information available. The dataset is partitioned forward in time, with the last month corresponding to the test period (100k sessions). The test data is further split randomly into two halves for public and final leaderboards respectively. Final leaderboard is used to determine the final team ranking in the competition. To encourage correct prediction of the bought item as early as possible, a random cut is applied to each test session where up to 50% of items are dropped at the end of the session. All teams are required to provide a ranking of 100 items from the candidate items set for each test session, and the submissions are evaluated using the Mean Reciprocal Rank (MRR) metric. There is a very strong temporal pattern in the data where over 30% of items in the candidate set only appear in the last three months of the training period.

2 APPROACH

2.1 Data Partitioning

To partition the data, we first sort all session by the time of the first item view. We then partition the ordered sessions forward in time to preserve temporal information in the data. Specifically, the last month of the 17-month training period is used as the validation set with over 81 thousand sessions and the rest as the training set. We



(a) Model Architecture (b) Transformer Layer

Figure 1: Model architecture for our Transformer model. The input item sequence (in reverse order) for a given session is first embedded with item e, positional p, and month m embeddings followed by L Transformer layers to form the final hidden representation. Note that the first item e_0 in the session is the unboserved purchased item encoded by the [mask] embedding that the model needs to predict.

also perform a random cut to the validation set where up to 50% of items are drooped at the end of each session. The accuracy with this partitioning generalizes well to the public leaderboard as it directly corresponds to the data partitioning for the leaderboard and final sets. The remaining 16-month training set is further divided into 16 subsets where each subset contains all the sessions in the same month.

2.2 Feature Construction

We conduct extensive feature engineering to describe item and session contexts, with particular focus on item-item similarities based on co-occurrence in history. In total we extract 88 features that can be organized into 3 main groups:

Item Categorical Features: summarize item content attributes. Instead of directly using the raw attributes as features, we represent them with learnable embeddings (see Section 2.3).

Item Count Features: extract the number of historical purchases and impressions (viewed but not purchased) for each item. We observe a strong temporal dependency in the dataset and such information reflects the changes in item popularity. We create separate features by aggregating over last month, last 2 months, and all time, to capture both short and long term item trends.

Item Co-occurrence Features: summarize the item-item similarity by computing the co-occurrence between the (candidate) purchased item and impression items, as well as the co-occurrence between every item pair in the session. These features are computed using the previous month sessions only for training at each month because of the strong time dependency.

2.3 Embedding Layer

To provide meaningful inputs to the Transformer, we create a learnable embedding layer. The embedding layer consists of item embeddings extracted from categorical content features, positional embeddings based on item position in the session, and month embeddings from session month. We note that as the computational complexity of self-attention layers grows quadratically with the sequence length, we set the maximum session length to be n. Any session with a length longer than n is truncated to the latest n items.

Item Embedding. For every item i we are provided a set of category-value pairs that summarize attributes such as color, fit and style. Some categories have multiple values associated with them (e.g. multiple colors). Let F^i be the set of categories associated with item i, and for each category c let F_c^i be the corresponding values associated with c. If $f_{c,v} \in \mathbb{R}^d$ is a learnable representation for the category-value pair (c, v), we define the item category embedding f_c^i as:

$$f_c^i = \begin{cases} \frac{1}{|F_c^i|} \sum_{v \in F_c^i} f_{c,v} & \text{if } c \in F^i \\ [unk] & \text{otherwise} \end{cases} \quad (1)$$

where f_c^i is set to a learnable [unk] embedding if category c is not present for item i. Note that while f_c^i summarizes item attribute information, it does not capture other intrinsic properties of the item such as popularity and trendiness. To this end, we let d^i be a learnable vector representation uniquely associated with item i. We then set the input *feature embedding* for item i to be $(d^i, f_{c_1}^i, f_{c_2}^i, ..., f_{c_k}^i)$ where $c_1, \ldots, c_k$ are the top-k most popular categories that appear in the dataset. The feature embedding contains rich information from both item content and contextual attributes extracted by the model. The corresponding *item embedding* is obtained by feeding the feature embedding through a linear layer to project to the Transformer input dimension d. We denote the full set of item embeddings after this linear projection as $E \in \mathbb{R}^{I \times d}$ where I is the total nubmer of items.

Positional Embedding. Order of the items in the session is very important for accurate prediction, in particular items viewed later in the session typically reveal more about user intent and purchased

item. To encode this information we add positional embeddings to the input embeddings. We find that learnable positional embeddings achieve better performance than sinusoidal embeddings.

Month Embedding. Fashion has a strong seasonality aspect where users typically shop for very different types of items in the summer than in winter. We incorporate seasonality information by encoding each of the twelve months with a separate learnable embedding of size d. For each session we then extract month from the timestamp of the first item and add the corresponding month embedding to all inputs.

2.4 Transformer Architecture

The overall model architecture is shown in Figure 1(a). Given a session with n items, we first add a $[mask]$ token at the front of the sequence which is be used for prediction. The input embeddings, $(h_0^0, h_1^0, ..., h_n^0)$, are formed by adding the item, positional, and month embeddings together. Since the attention function is computed on all inputs simultaneously, we stack the input embeddings to form input matrix $H^0 \in \mathbb{R}^{n \times d}$. We then iteratively compute the hidden representation h_i^l for item i at layer l with the Transformer layer [6]. As illustrated in Figure 1(b), the Transformer layer consists of two parts: a Multi-Head Attention module and a Position-wise Feed-Forward network.

Multi-Head Attention. Following the previous work [3, 5, 6], we adopt Multi-Head Attention by projecting H^l at layer l into h subspaces where h is the number of attention heads. Each head has a different set of learnable parameters and the output representations are concatenated and projected:

$$\begin{aligned} \text{MultiHead}(H^l) &= \text{Concat}(\text{head}_1, .., \text{head}_h)W^o \\ \text{head}_i &= \text{Attention}(H^l W_i^Q, H^l W_i^K, H^l W_i^V) \\ \text{Attention}(Q, K, V) &= \text{softmax}\left(\frac{QK^T}{\sqrt{d/h}}\right)V \end{aligned} \tag{2}$$

where the projection matrices, $W_i^Q \in \mathbb{R}^{d \times d/h}$, $W_i^K \in \mathbb{R}^{d \times d/h}$, $W_i^V \in \mathbb{R}^{d \times d/h}$, and $W^o \in \mathbb{R}^{d \times d}$ are learnable parameters.

Position-wise Feed-Forward Network. To introduce non-linearity into the model, we adopt Position-wise Feed-Forward network from [3, 5, 6]:

$$\text{FFN}(h_i) = \max(0, h_i W_1 + b_1)W_2 + b_2 \tag{3}$$

While the same linear tranformations, $W_1 \in \mathbb{R}^{d \times 2d}$, $b_1 \in \mathbb{R}^{2d}$, $W_2 \in \mathbb{R}^{2d \times d}$, and $b_2 \in \mathbb{R}^d$, are applied at each position, different tranformations are used from layer to layer.

2.5 Training Objectives

Given that there are fewer than 1M sessions in the training set, we find that even relatively small Tansformer configurations quickly overfit when trained in a purely supervised way. Inspired by the recent work on self-supervised training [3], we propose to combine two tasks for session-based recommendation: Masked Session Modeling (MSM) and Purchased Item Prediction (PIP). MSM regularizes the model with a self-supervised loss, and PIP aims to accurately predict the purchased items.

Masked Session Modeling. MSM is similar to the language model pre-training in NLP, where contextual token representations are obtained via a self-supervised objective consisting of masking out portions of the sentence. In our case, we randomly mask 15% of the items in the session by replacing them with the token $[mask]$, and predict the masked items given the rest of the context. For example:

Session: $[5, 4, 3, 2, 1]$ **Purchased Item:** $[6]$

Input: $[[mask]_0, e_5, e_4, e_3, e_2, e_1]$ —Random Mask→ $[[mask]_0, e_5, e_4, [mask]_1, e_2, [mask]_2]$

Label: $[mask]_0 = e_6, [mask]_1 = e_3, [mask]_2 = e_1$

Here, the first mask, $[mask]_0$, is used for purchased item, and the rest are masked session items to be predicted via MSM. Dot product is performed between the final hidden state of the masked token, h_m^L, and the item embeddings, $E \in \mathbb{R}^{I \times d}$, to produce the item prediction scores $P_m \in \mathbb{R}^I$. We adapt weight-tying [2], a commonly used technique in the NLP community, by tying with the output layer projection matrix to the input item embedding. This is motivated by the fact that the inputs and outputs both represent items and should thus be in the same space. We apply softmax on the item prediction scores to generate the probabilities:

$$\begin{aligned} \text{if } m = 0 \quad f(P_m)_i &= \begin{cases} \frac{e^{P_{m,i}}}{\sum_{j \in C} e^{P_{m,j}}} & \text{if } i \in C \\ 0 & \text{otherwise} \end{cases} \\ \text{else} \quad f(P_m)_i &= \begin{cases} \frac{e^{P_{m,i}}}{\sum_{j \in V} e^{P_{m,j}}} & \text{if } i \in V \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

where C is the set of candidate items for the purchased item, and V is the union of all the items that have been viewed in the same month as current session. In the challenge, C is set to the union of all items that were purchased during the same month. Both C and V thus preserve the temporal dynamics and force the model to discriminate between items that are relevant at the time of the session. We use cross entropy objective to match predictions for masked items with ground truth:

$$L_{MSM} = -\frac{1}{|M|} \sum_{m \in M} \sum_{i \in I} y_{mi} \log f(P_m)_i \tag{5}$$

where M contains all masked items (including purchased item), and y_{mi} is 1 for the item that was masked with mask m and 0 for all other items.

Purchased Item Prediction. In this approach, we consider session-based recommendation as a purely supervised task. We randomly sample k "negative" items per session from $C \setminus \{i^p\}$, where i^p is the purchased item. For each sampled item, we place it in the $[mask]_0$ position of the purchased item and make forward pass through the Transformer to obtain the corresponding representation h_0^L. Extracted item features (see Section 2.2) are then concatenated with h_0^L and passed through two fully connected layers with batchnorm and ReLU activations to output the sigmoid prediction score for the item. This score represents the likelihood that the item was purchased at the end of the session. The goal is to maximize the score for purchased item i^p and lower it for all

negative items so we again leverage the cross entropy objective:

$$L_{PIP} = -\frac{1}{(k+1)}\left(\log(\hat{y}_p) + \sum_k \log(1-\hat{y}_k)\right) \quad (6)$$

Here, $\hat{y}_p$ and $\hat{y}_k$ are prediction scores for purchased and sampled negative items respectively. Note that PIP allows to incorporate arbitrary item features into the classifier on top of h_0^L and has greater flexibility. But this comes at the cost of slower inference since instead of fast dot product inference in MSM, we now have to make forward passes through the Transformer for every item in the candidate set.

In practice, we find that it is beneficial to combine the two tasks, and our best model uses both losses with hyper-parameter α controlling the contribution between them:

$$L = \alpha L_{MSM} + L_{PIP} \quad (7)$$

2.6 Inference

We use a different inference method for each of the training tasks. For models trained with MSM, we take the final hidden state of the first token, h_0^L, to perform dot product on the item embeddings from the candidate set. Dot-product scores are then sorted to get the top-k item list. For models trained with PIP, we pass each item from the candidate set through the Transformer by replacing $[mask]_0$ with that item's embedding. Sigmoid scores from the classifier are then similarly sorted to get the top-k list. For models trained with both tasks, we find that PIP inference leads to better performance and use that in all experiments.

3 EXPERIMENTS

3.1 Training Details

All experiments are conducted on Ubuntu servers with Intel Xeon(R) E5-2686 v4 @ 2.30GHz CPUs, 500GB RAM, and NVIDIA Tesla V100 GPUs. By using the first 16 months for training set and the last month as the validation set, we generate 918, 382 training sessions and 81, 618 validation sessions. After finding the optimal set of parameters, we retrain the model on the full 17 month period that includes both training and validation sets, and then run inference on the test set. Item features include 73 categorical features with embedding size 8, 4 normalized co-occurrence features for purchased and impressed items, and 6 item count features for last month, last two months, and all months for both purchases and impressions. The Transformer has input dimension 128 and feed-forward dimension 256 with 4 layers, 4 attention heads and 0.1 dropout rate. The maximum session length is set to 100 and the number of negatives for the PIP objective is 19. The model is jointly trained using AdamW optimizer [4] with a batch size of 512. The learning rate is warmed up for 2000 iterations until 1e-3 and decayed with cosine decay after. Following the challenge rules [1], we use the Mean Reciprocal Rank (MRR) objective to evaluate model performance.

3.2 Results

In addition to Transformer, we also implement three other models: XGBoost, VAE, and DAE. The results for all four models on the leaderboard set are shown in Table 1. We see that the Transformer outperforms all other models with a large improvement. The key difference between Transformer and the other models is in the objective function. XGBoost, VAE, and DAE only perform supervised learning by predicting the purchased item, while Transformer incorporates a self-supervised objective. This allow to use a much larger model size, around 10x larger than the baselines, without overfitting. Final leaderboard results are shown in Table 3. We use a linear ensemble of all four models and achieve highly competitive performance placing 2'nd.

Table 1: Leaderboard results for different model types.

Model	MRR
XGBoost	0.1993
VAE	0.2026
DAE	0.2035
Transformer	0.2121

Table 2: Leaderboard ablation results for different values of α.

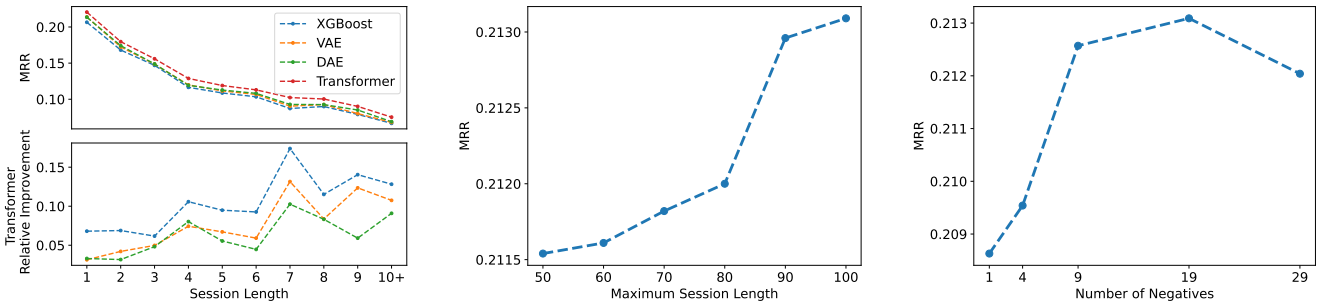
Objective	MRR
MSM	0.2109
PIP	0.2030
MSM+PIP (α=0.1)	0.2121
MSM+PIP (α=0.2)	0.2115
MSM+PIP (α=0.5)	0.2108
MSM+PIP (α=1.0)	0.2104

Table 3: Final challenge results from the top-5 teams, our team is LAYER 6.

Team	MRR
1. zzh	0.2160
2. LAYER 6	0.2148
3. NVIDIA RAPIDSAI	0.2086
4. MooreWins	0.2076
5. THLUO	0.2062

Objective Analysis. We investigate the effects of MSM and PIP tasks on model performance by varying the α parameter. The results are shown in Table 2. MSM outperforms PIP, and joint learning with two objectives (MSM+PIP) outperforms each objective individually. MSM allows the model to learn correlations within the viewed items in the session and between viewed and purchased items by masking the purchased item and 15% of the viewed items. On the other hand, the input for PIP has no masking, so the model can take advantage of the full information. Combining the two tasks consistently produces the best performance.

Session Length Analysis. To understand the effect of session length on performance, we compute MRR for each session length from 1 to 10+ across the four models. We also compute relative improvement for Transformer over XGBoost, VAE, and DAE at each length. The results evaluated on the validation set are shown in



(a) (b) (c)

Figure 2: (a) Model performance by session length and the relative improvement for Transformer with respect to XGBoost, VAE, and DAE. (b) Transfomer performance for different maximum input session lengths. (c) Transformer performance for different number of negatives in the PIP task.

Figure 2(a). For all four models the performances drops on longer sessions. One possible explanation is that longer sessions contain multiple intents where user is browsing around before settling on what to buy, making inference harder. However, we can also observe that the relative performance improvement for Transformer is generally increasing at longer session lengths. The attention mechanism in Transformer enables the model to only focus on the relevant portions of the item sequence which can be particularly effective for longer sessions.

Maximum Session Length Analysis. We analyze the effect of maximum input session length on Transformer, and the results are shown in Figure 2(b). The MRR steadily increases with longer maximum session length. As we discussed in the previous section, Transformer self-attention can effectively extract information from longer sessions so increasing maximum length is beneficial. However, self-attention also scales quadratically (both runtime and memory) with input length, and we find it prohibitively expensive to increase maximum length beyond 100. In the future work, a promising direction is to explore methods for efficient training with longer sequence lengths.

Number of Negatives Analysis. An important component of the PIP object is the number of negative samples used during training. Here, we need to find a balance between having enough negative samples to provide a rich training signal, but at the same avoid skewing the model towards predicting everything as negative. Figure 2(c) shows the effect of the number of negatives on the model accuracy. We can observe that MRR peaks a 19 so 20:1 ratio of negative to positive samples is close to optimal for this dataset.

4 CONCLUSION

In this paper, we present our approach to the 2022 ACM RecSys Challenge organized by Dressipi. Our best model consists of feature extraction followed by embedding and Transformer self-attention layers. This model is trained with a combination of self-supervised and fully supervised tasks, and we demonstrate the effectiveness of incorporating self-supervised learning for session modelling. We achieve highly competitive performance, placing 2'nd on the final leaderboard out of over 300 teams.

REFERENCES

[1] 2022. *RecSys Challenge 2022.* http://www.recsyschallenge.com/2022/
[2] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. 2021. Transformers4Rec: Bridging the Gap between NLP and Sequential/Session-Based Recommendation. In *ACM Conference on Recommender Systems.*
[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics.*
[4] Ilya Loshchilov and Frank Hutter. 2017. Fixing Weight Decay Regularization in Adam. *CoRR* abs/1711.05101 (2017).
[5] 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 *International Conference on Information and Knowledge Management.*
[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *Neural Information Processing Systems.*