Behavior Sequence Transformer for E-commerce Recommendation in Alibaba

Qiwei Chen, Huan Zhao*
Wei Li, Pipei Huang, Wenwu Ou
Alibaba Search&Recommendation Group
Beijing&Hangzhou, China
{chenqiwei.cqw,chuanfei.zh,rob.lw,pipei.hpp,santong.oww}@alibaba-inc.com

ABSTRACT

Deep learning based methods have been widely used in industrial recommendation systems (RSs). Previous works adopt an Embedding&MLP paradigm: raw features are embedded into low-dimensional vectors, which are then fed on to MLP for final recommendations. However, most of these works just concatenate different features, ignoring the sequential nature of users' behaviors. In this paper, we propose to use the powerful Transformer model to capture the sequential signals underlying users' behavior sequences for recommendation in Alibaba. Experimental results demonstrate the superiority of the proposed model, which is then deployed online at Taobao and obtain significant improvements in online Click-Through-Rate (CTR) comparing to two baselines.

1 INTRODUCTION

During the last decade, Recommender Systems (RSs) have been the most popular application in industry, and in the past five years, deep learning based methods have been widely used in industrial RSs, e.g., Google [2, 3] and Airbnb [5]. In Alibaba, the largest e-commerce platform in China, RSs have been the key engine for its Gross Merchandise Volume (GMV) and revenues, and various deep learning based recommendation methods have been deployed in rich e-commerce scenarios [1, 8, 10, 11, 14, 15, 17, 18]. As introduced in [15], the RSs in Alibaba are a two-stage pipeline: match and rank. In match, a set of similar items are selected according to items users interacted with, and then a fine-tuned prediction model is learned to predict the probability of users clicking the given set of candidate items.

In this paper, we focus on the rank stage at Alibaba's Taobao, China's largest Consumer-to-Consumer (C2C) platform owned by Alibaba, where we are have millions of candidate items, and we need to predict the probability of a user clicking the candidate items given his/her historical behaviors. In the era of deep learning, embedding and MLP have been the standard paradigm for industrial RSs: large numbers of raw features are embedded into low-dimensional spaces as vectors, and then fed into fully connected layers, known as multi layer perceptron (MLP), to predict whether a user will click an item or not. The representative works are wide and deep learning (WDL) networks [2] from Google and Deep Interest networks (DIN) from Alibaba [17].

At Taobao, we build the rank model on top of WDL, where various features are used in the embedding&MLP paradigm , e.g., the category and brand of an item, the statistical numbers of an item, or the user profile features. Despite the success of this framework, it is inherently far from satisfying since it ignores one type of very important signals in practice, i.e., the sequential signal underlying the users' behavior sequences, i.e., users' clicked items in order. In reality, the order matters for predicting the future clicks of users. For example, a user tends to click a case for a cellphone after he or she bought an iphone at Taobao, or tries to find a suitable shoes after buying a pair of trousers. In this sense, it is problematic without considering this factor when deploying a prediction model in the rank stage at Taobao. In WDL [2], they simply concatenates all features without capturing the order information among users' behavior sequences. In DIN [17], they proposed to use attention mechanism to capture the similarities between the candidate item and the previously clicked items of a user, while it did not consider the sequential nature underlying the user's behavior sequences.

Therefore, in this work, to address the aforementioned problems facing WDL and DIN, we try to incorporate sequential signal of users' behavior sequences into RS at Taobao. Inspired by the great success of the Transformer for machine translation task in natural language processing (NLP) [4, 13], we apply the self-attention mechanism to learn a better representation for each item in a user's behavior sequence by considering the sequential information in embedding stage, and then feed them into MLPs to predict users' responses to candidate items. The key advantage of the Transformer is that it can better capture the dependency among words in sentences by the self-attention mechanism, and intuitively speaking, the "dependency" among items in users' behavior sequences can also be extracted by the Transformer. Therefore, we propose the user behavior sequence transformer (BST) for e-commerce recommendation at Taobao. Offline experiments and online A/B test show the superiority of BST comparing to existing methods. The BST has been deployed in rank stage for Taobao recommendation, which provides recommending service for hundreds of millions of consumers everyday.

The remainder of this paper is organized as follows: the architecture is elaborated in Section 2, and then the experimental results including offline and online ones are presented in Section 3. Related work are reviewed in Section 4, and finally we conclude our work in Section 5.

2 ARCHITECTURE

In the rank stage, we model the recommendation task as Click-Through Rate (CTR) prediction problem, which can be defined as follows: given a user's behavior sequence $S(u) = \{v_1, v_2, ..., v_n\}$ clicked by a user u, we need to learn a function, $\mathcal{F}$, to predict the probability of u clicking the target item v_t, i.e., the candidate one. Other Features include user profile, context, item, and cross features.

*Qiwei Chen and Huan Zhao contribute equally to this work, and Pipei Huang is the corresponding author.

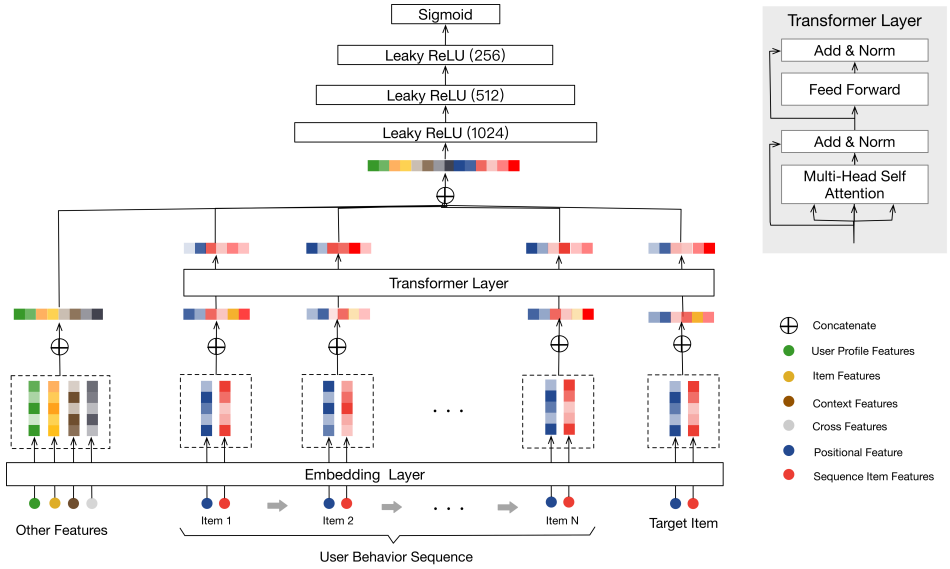

Figure 1: The overview architecture of the proposed BST. BST takes as input the user's behavior sequence, including the target item, and "Other Features". It firstly embeds these input features as low-dimensional vectors. To better capture the relations among the items in the behavior sequence, the transformer layer is used to learn deeper representation for each item in the sequence. Then by concatenating the embeddings of Other Features and the output of the transformer layer, the three-layer MLPs are used to learn the interactions of the hidden features, and sigmoid function is used to generate the final output. Note that the "Positional Features" are incorporated into "Sequence Item Features".

We build BST on top of WDL [2], and the overview architecture is shown in Figure 1. From Figure 1, we can see that it follows the popular embedding&MLP paradigm, where the previously clicked items and related features are firstly embedded into low-dimensional vectors before fed on to MLP. The key difference between BST and WDL is that we add transformer layer to learn better representations for users' clicked items by capturing the underlying sequential signals. In the following parts, we introduce in a bottom-up manner the key components of BST: embedding layer, transformer layer, and MLP.

2.1 Embedding Layer

The first component is the embedding layer, which embeds all input features into a fixed-size low-dimensional vectors. In our scenarios, there are various features, like the user profile features, item features, context features, and the combination of different features, i.e., the cross features[1]. Since this work is focused on modeling the behavior sequence with transformer, we denote all these features as "Other Features" for simplicity, and give some examples in Table 1. As shown in Figure 1, we concatenate "Other features" in left part and embed them into low-dimensional vectors. For these features, we create an embedding matrix $\mathbf{W}_o \in \mathbb{R}^{|D| \times d_o}$, where d_o is the dimension size.

Besides, we also obtain the embeddings for each item in the behavior sequence, including the target item. As shown in Figure 1, we use two types of features to represent an item, "Sequence Item Features"(in red) and "Positional Features" (in dark blue), where "Sequence Item Features" include *item_id* and *category_id*. Note that an item tends to have hundreds of features, while it is too expensive to choose all to represent the item in a behavior sequence. As introduced in our previous work [15], the *item_id* and *category_id* are good enough for the performance, we choose these two as sparse features to represent each item in embedding the user's behavior sequences. The "Positional Features" corresponds the following "positional embedding". Then for each item, we concatenate Sequence Item Features and Positional Features, and create an embedding matrix $\mathbf{W}_V \in \mathbb{R}^{|V| \times d_V}$, where d_V is the dimension size of the embedding, and $|V|$ is the number of items. We use $\mathbf{e}_i \in \mathbb{R}^{d_V}$ to represent the embedding for the i-th item in a given behavior sequence.

Table 1: The "Other Features" shown in left side of Figure 1. We use much more features in practice, and show a number of effective ones for simplicity.

User	Item	Context	Cross
gender	category_id	match_type	age * item_id
age	shop_id	display position	os * item_id
city	tag	page No.	gender * category_id
...	...	...	...

Positional embedding. In [13], the authors proposed a positional embedding to capture the order information in sentences. Likewise, the order exists in users' behavior sequences. Thus, we add the "position" as an input feature of each item in the bottom layer before it is projected as a low-dimensional vector. Note that the position value of item v_i is computed as $pos(v_i) = t(v_t) - t(v_i)$,

[1]Though the combination of features can be automatically learned by neural networks, we still incorporate the some hand-crafted cross features, which have been demonstrated useful in our scenarios before the deep learning era.

where $t(v_t)$ represents the recommending time and $t(v_i)$ the timestamp when user click item v_i. We adopt this method since in our scenarios it outperforms the *sin* and *cos* functions used in [13].

2.2 Transformer layer

In this part, we introduce the Transformer layer, which learns a deeper representation for each item by capturing the relations with other items in the behavior sequences.

Self-attention layer. The scaled dot-product attention [13] is defined as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}, \tag{1}$$

where $\mathbf{Q}$ represents the queries, $\mathbf{K}$ the keys and $\mathbf{V}$ the values. In our scenarios, the self-attention operations takes the embeddings of items as input, and converts them to three matrices through linear projection, and feeds them into an attention layer. Following [13], we use the multi-head attention:

$$\mathbf{S} = \text{MH}(\mathbf{E}) = \text{Concat}(head_1, head_2, \cdots, head_h)\mathbf{W}^H, \tag{2}$$

$$head_i = \text{Attention}(\mathbf{E}\mathbf{W}^Q, \mathbf{E}\mathbf{W}^K, \mathbf{E}\mathbf{W}^V), \tag{3}$$

where the projection matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d}$, and $\mathbf{E}$ is the embedding matrices of all items, and h is the number of heads.

Point-wise Feed-Forward Networks. Following [13], we add point-wise Feed-Forward Networks (FFN) to further enhance the model with non-linearity, which is defined as follows:

$$\mathbf{F} = FFN(\mathbf{S}). \tag{4}$$

To avoid overfitting and learn meaningful features hierarchically, we use dropout and LeakyReLU both in self-attention and FFN. Then the overall output of the self-attention and FFN layers are as follows:

$$\mathbf{S}' = \text{LayerNorm}(\mathbf{S} + \text{Dropout}(MH(\mathbf{S})), \tag{5}$$

$$\mathbf{F} = \text{LayerNorm}\big(\mathbf{S}' + \text{Dropout}(\text{LeakyReLU}(\mathbf{S}'\mathbf{W}^{(1)} + b^{(1)})\mathbf{W}^{(2)} + b^{(2)})\big), \tag{6}$$

where $\mathbf{W}^{(1)}, b^{(1)}, \mathbf{W}^{(2)}, b^{(2)}$ are the learnable parameters, and $LayerNorm$ is the standard normalization layer.

Stacking the self-attention blocks. After the first self-attention block, it aggregates all the previous items' embeddings, and to further model the complex relations underlying the item sequences, we stack the self-building blocks and the b-th block is defined as:

$$\mathbf{S}^b = SA(F^{(b-1)}), \tag{7}$$

$$\mathbf{F}^b = FFN(\mathbf{S}^b), \forall i \in 1, 2, \cdots, n. \tag{8}$$

In practice, we observe in our experiments $b = 1$ obtains better performance comparing to $b = 2, 3$ (see Table 4). For the sake of efficiency, we did not try larger b and leave this for future work.

2.3 MLP layers and Loss function

By concatenating the embeddings of Other Features and the output of the Transformer layer applying to the target item, we then use three fully connected layers to further learn the interactions among the dense features, which is standard practice in industrial RSs.

To predict whether a user will click the target item v_t, we model it as a binary classification problem, thus we use the sigmoid function as the output unit. To train the model, we use the cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{(x,y)\in\mathcal{D}} \big(y \log p(x) + (1-y)\log(1-p(x))\big), \tag{9}$$

where $\mathcal{D}$ represent all the samples, and $y \in \{0, 1\}$ is the label representing whether user have click an item or not, $p(x)$ is the output of the network after the sigmoid unit, representing the predicted probability of sample x being clicked.

Table 2: Statistics of the constructed Taobao dataset.

Dataset	#Users	#Items	#Samples
Taobao	298,349,235	12,166,060	47,556,271,927

3 EXPERIMENTS

In this section, we present the experimental results.

3.1 Settings

Dataset. The dataset is constructed from the log of Taobao App [2]. We construct an offline dataset based on users' behaviors in eight days. We use the first seven days as training data, and the last day as test data. The statistics of the dataset is shown in Table 2. We can see that the dataset is extremely large and sparse.

Baselines. To show the effectivene of BST, we compare it with two models: WDL [2] and DIN [17]. Besides, we create a baseline method by incorporating sequential information into WDL, denoted as WDL(+Seq), which aggregates the embeddings of the previously clicked items in average. Our framework is built on top of WDL by adding sequential modeling with the Transformer, while DIN is proposed to capture the similarities between the target item and the previous clicked items with attention mechanisms.

Evaluation metric. For the offline results, we use Area Under Curve (AUC) score to evaluate the performance of different models. For online A/B test, we use the CTR and average RT to evaluate all models. RT is short for response time (RT), which is the time cost of generating recommending results for a given query, i.e., one request of a user at Taobao. We use average RT as the metric to evaluate the efficiency of different in online production environment.

Settings. Our model is implemented with Python 2.7 and Tensorflow 1.4, and the *"Adagrad"* is chosen as the optimizer. Besides, we give the detail of the model parameters in Table 3.

3.2 Results Analysis

The results are shown in Table 4, from which, we can see the superiority of BST comparing to baselines. In specific, the AUC of offline experiment is improved from 0.7734 (WDL) and 0.7866 (DIN) to 0.7894 (BST). When comparing WDL and WDL(+Seq), we can see that the effectiveness of incorporating sequential information in an simple averaging manner. It means with the help of self-attention, BST provides a powerful capability to capture the sequential signal underlying users' behavior sequences. Note that from our practical experience, even the small gain of offline AUC can lead to huge gain in online CTR. A similar phenomenon is reported by the researchers from Google in WDL [2].

[2] https://www.taobao.com/

Table 3: The configuration of BST, and the meaning of the parameters can be inferred from their names.

Configuration of BST.			
embedding size	4 ∼64	batch size	256
head number	8	dropout	0.2
sequence length	20	#epochs	1
transformer block	1	queue capacity	1024
MLP Shape	1024 * 512 * 256	learning rate	0.01

Besides, in terms of efficiency, the average RT of BST is close to those of WDL and DIN, which guarantees the feasibility of deploying a complex model like the Transformer in real-world large-scale RSs.

Finally, we also shown the influences of stacking the self-attention layers in Section 2.2. From Table 4, we can see that $b = 1$ obtains the best offline AUC. This may be due to the fact that the sequential dependency in users' behavior sequence is not as complex as that in sentences in machine translation task, thus smaller number of blocks are enough to obtain good performance. Similar observation is reported in [7]. Therefore we choose $b = 1$ to deploy BST in production environment, and only report the online CTR gain for $b = 1$ in Table 4.

4 RELATED WORK

In this section, we briefly review the related work on deep learning methods for CTR prediction. Since the proposal of WDL [2], a series of works have been proposed to improve the CTR with deep learning based methods, e.g., DeepFM [6], XDeepFM [9], Deep and Cross networks [16], etc. However, all these previous works focus on feature combinations or different architectures of neural networks, ignoring the sequential nature of users' behavior sequence in real-world recommendation scenarios. Recently, DIN [17] was proposed to deal with users' behavior sequences by an attention mechanism. The key difference between our model and DIN lies in that we propose to use the Transformer [13] to learn a deeper representation for each item in users' behavior sequences, while DIN tried to capture different similarities between the previously clicked items and the target item. In other words, our model with transformer are more suitable for capturing the sequential signals. In [7, 12], the Transformer model is proposed to solve the sequential recommendation problem in sequence-to-sequence manner while the architectures are different from our model in terms of CTR prediction.

5 CONCLUSION

In this paper, we present the technical detail of how we apply the Transformer [13] to Taobao recommendation. By using the powerful capability of capturing sequential relations, we show the superiority of the Transformer in modeling user behavior sequences for recommendation by extensive experiments. Besides, we also present the detail of deploying the proposed model in production environment at Taobao, which provides recommendation service for hundreds of millions of users in China.

6 ACKNOWLEDGMENTS

We would like to thank colleagues of our team - Jizhe Wang, Chao Li, Zhiyuan Liu, Yuchi Xu and Mengmeng Wu for useful discussions and supports on this work. We are grateful to Jinbin Liu, Shanshan Hao and Yanchun Yang from Alibaba Distributed Computing Team, and Kan Liu, Tao Lan from Alibaba Online Inference Team, who help deploy the model in production. We also thank the anonymous reviewers for their valuable comments and suggestions that help improve the quality of this manuscript.

Table 4: Offline AUCs and online CTR gains of different methods. Online CTR gain is relative to the control group.

Methods	Offline AUC	Online CTR Gain	Average RT(ms)
WDL	0.7734	-	13
WDL(+Seq)	0.7846	+3.03%	14
DIN	0.7866	+4.55%	16
BST($b = 1$)	**0.7894**	**+7.57%**	20
BST($b = 2$)	0.7885	-	-
BST($b = 3$)	0.7823	-	-

REFERENCES

[1] Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. 2019. POG: Personalized Outfit Generation for Fashion Recommendation at Alibaba iFashion.
[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. , 7–10 pages.
[3] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*. 191–198.
[4] 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).
[5] Mihajlo Grbovic and Haibin Cheng. 2018. Real-time personalization using embeddings for search ranking at Airbnb. In *KDD*. 311–320.
[6] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *IJCAI*. 1725–1731.
[7] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. 197–206.
[8] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Pipei Huang, Huan Zhao, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall.
[9] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining explicit and implicit feature interactions for recommender systems. In *KDD*. 1754–1763.
[10] Yabo Ni, Dan Ou, Shichen Liu, Xiang Li, Wenwu Ou, Anxiang Zeng, and Luo Si. 2018. Perceive Your Users in Depth: Learning Universal User Representations from Multiple E-commerce Tasks. In *KDD*. 596–605.
[11] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Wenwu Ou, and Dan Pei. 2019. Personalized Context-aware Re-ranking for E-commerce Recommender Systems. *arXiv preprint arXiv:1904.06813* (2019).
[12] 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.
[13] 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.
[14] Chenglong Wang, Feijun Jiang, and Hongxia Yang. 2017. A hybrid framework for text modeling with convolutional rnn. In *KDD*. 2061–2069.
[15] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. 2018. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *KDD*. 839–848.
[16] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD'17 (ADKDD'17)*.
[17] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *KDD*. 1059–1068.
[18] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning Tree-based Deep Model for Recommender Systems. In *KDD*. 1079–1088.