Deep Multifaceted Transformers for Multi-objective Ranking in Large-Scale E-commerce Recommender Systems

Yulong Gu[1], Zhuoye Ding[1], Shuaiqiang Wang[2], Lixin Zou[3], Yiding Liu[2], Dawei Yin[2]
[1]JD.com, Beijing, China [2]Baidu Inc., Beijing, China [3]Tsinghua University, Beijing, China
{guyulongcs,zoulixin15,liuyiding.tanh}@gmail.com
dingzhuoye@jd.com,wangshuaiqiang@baidu.com,yindawei@acm.org

ABSTRACT

Recommender Systems have been playing essential roles in e-commerce portals. Existing recommendation algorithms usually learn the ranking scores of items by optimizing a single task (*e.g.,* Click-through rate prediction) based on users' historical click sequences, but they generally pay few attention to simultaneously modeling users' multiple types of behaviors or jointly optimize multiple objectives (*e.g.,* both Click-through rate and Conversion rate), which are both vital for e-commerce sites. In this paper, we argue that it is crucial to formulate users' different interests based on multiple types of behaviors and perform multi-task learning for significant improvement in multiple objectives simultaneously. We propose Deep Multifaceted Transformers (DMT), a novel framework that can model users' multiple types of behavior sequences simultaneously with multiple Transformers. It utilizes Multi-gate Mixture-of-Experts to optimize multiple objectives. Besides, it exploits unbiased learning to reduce the selection bias in the training data. Experiments on JD real production dataset demonstrate the effectiveness of DMT, which significantly outperforms state-of-art methods. DMT has been successfully deployed to serve the main traffic in the commercial Recommender System in JD.com. To facilitate future research, we release the codes and datasets at https://github.com/guyulongcs/CIKM2020_DMT.

CCS CONCEPTS

• **Information systems** → **Personalization**; **Recommender systems**;

KEYWORDS

Recommender Systems; E-commerce; Multi-task Learning; Click-Through Rate Prediction; Conversation Rate Prediction

ACM Reference Format:
Yulong Gu[1], Zhuoye Ding[1], Shuaiqiang Wang[2], Lixin Zou[3], Yiding Liu[2], Dawei Yin[2]. 2020. Deep Multifaceted Transformers for Multi-objective Ranking in Large-Scale E-commerce Recommender Systems. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3340531.3412697

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412697

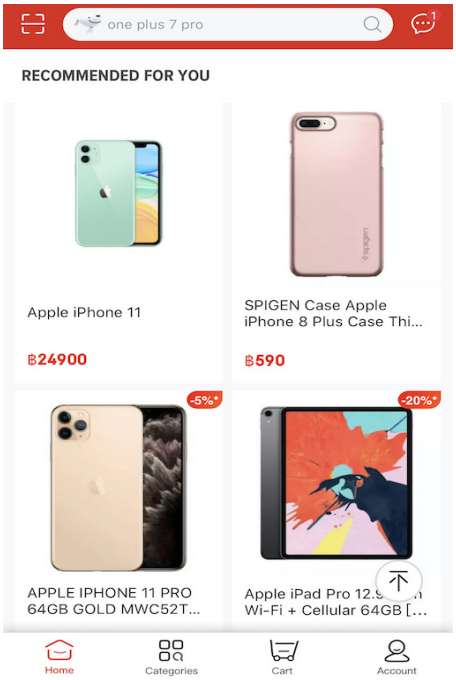

Figure 1: A real example in the Recommender Systems in JD.

1 INTRODUCTION

Recommender Systems, which aim to recommend potentially interested items for users and solve the information explosion problem, are playing critical roles in E-commerce sites (*e.g.,* Amazon, JD.com, Alibaba) [10, 20, 41, 42], videos sharing sites (*e.g.,* YouTube) [7], picture sharing sites (*e.g.,* Pinterest) [36], social networks (*e.g.,* Facebook) [11, 13] and so on. For example, in JD.com, one of the largest E-commerce sites in the world, the Recommender System serves more than 0.3 billion users in China, Thailand, Malaysia and other countries, and contributes billions of dollars for the Gross Merchandise Volume (GMV) (*i.e.,* the total sales value for merchandise sold) each year. Industrial Recommender Systems are usually consisted of two stages: The first stage is Candidate generation [4, 7, 21, 25, 32], which selects hundreds or thousands of products as candidates from millions or even billions of items; The second stage is Ranking [7, 41], which ranks the selected candidates and returns several top-ranked items for each user. In this paper, we focus on the ranking stage, which is critical for improving both the satisfaction of the users and the revenue of the sites.

Many ranking methods for recommendations have been proposed, including tree-based methods [9], deep neural networks [5, 7, 10, 40, 41], reinforcement learning [42, 43] based models and so on. However, designing a real-world large-scale E-commerce recommender system still faces many challenges, including:

- **Multi-objective ranking.** We need to simultaneously optimize multiple objectives, which could be related but sometimes need a trade-off. In E-commerce portals, for example, the recommended products are expected to be clicked and purchased by the users. Thus here both Click-Trough Rate (CTR) and Conversion Rate (CVR) are two important objectives. The most straightforward method that always returns the most popular products for all users may leads to high CTR but low CVR resulting from its poor personalization of the recommendations to the users.
- **Multiple types of users' behaviors.** Users usually have many types of behaviors (*e.g.,* click, add to cart, order) in E-commerce sites. They can imply users' real-time intents in different aspects. What's more, they may play different roles for different ranking objectives. For example, empirically recommending some products that are similar to purchased products by the target user will lead to high CTR but low CVR. Currently, most existing studies aim to model users' click sequences and optimize the CTR-based objective for recommendation [40, 41]. It still remains as an open problem to effectively formulate users' multiple types of behaviors for multi-objective ranking.
- **Biased implicit feedbacks.** Existing recommender systems usually exploit users' implicit feedback (e.g. click or not) to learn the ranking model. However, selection bias exists in the implicit feedback data. A user might click a product simply because it is ranked high, and such observation is called "position bias". Besides, given a target item, the neighboring items near to it could also influence its CTR, which is refferred as "neighboring bias" in this paper. A user may less likely to click a product if it is surrounded by many similar ones in the same category. For example, in Figure 1, the product "iPhone 11"" may have higher probability of being clicked because it is ranked in the first position, and the three "iPhones" may have strong competition for being clicked. Which types of bias information are more influential? How to effectively model and reduce the bias? They are both open questions in E-commerce scenarios.

Some pioneering studies have investigated the multi-task learning problem for recommender systems [19, 22, 23, 33, 38], where the Multi-gate Mixture-of-Experts (MMoE) [23, 38] method has achieved state-of-the-art performance. However, these works only model one type of behaviors (*i.e.,* clicks). We argue that it is crucial to formulate users' different interests based on multiple types of behaviors for significant improvement in multiple objectives simultaneously. Furthermore, existing works [7, 40, 41] usually ignore the bias issue when they model users' behaviors, which is extremely important in real-world systems.

To address these challenges, we propose a novel framework Deep Multifaceted Transformers (DMT) for the multi-objective ranking in large-scale E-commerce Recommender Systems. In particular, DMT exploits multiple Transformers [30] to model users' multiple types of behavior sequences, and represent users' real-time intents in different aspects as multiple low dimensional interest vectors. Then the MMOE modules can model the relationship and conflict between multi tasks, and optimize multiple objectives simultaneously based on these interest vectors. To consider the bias in the implicit feedbacks, DMT utilizes a Bias Deep Neural Network to estimate the propensity score based on the selection bias features, which further improves the performance of the framework.

To summarize, our major contributions are listed as follows:

- We investigate the multi-objective E-commerce recommendation problem by modeling users' multiple types of behavior sequences.
- We present DMT, which exploits multiple transformers to model users' diverse behavior sequences, utilizes Multi-gate Mixture-of-Experts to jointly optimize multi-objectives, and uses a Bias Deep Neural Network for reducing the bias in E-commerce Recommender Systems.
- We conduct extensive experiments and demonstrate that DMT outperforms state-of-the-art baselines greatly for both click and conversion tasks. DMT has been deployed in the commercial Recommender System in JD.com, contributing billions of dollars in revenue each year.

2 RELATED WORK

2.1 CTR prediction

Click-through rate (CTR) prediction, which aims to estimate the probability of a user clicking on the item, is one of the long-standing core tasks in industrial applications, such as Recommender Systems [5, 7, 37], Search Engine [35], Advertising [13, 40, 41] and so on. GBDT [9] is one of the most popular and successful methods for CTR prediction in industrial systems [13, 35]. It has the advantage of simplicity, effectiveness, good explainability, flexible extensibility and so on. In recent years, deep learning based methods have achieved appealing performance for CTR prediction. These methods [5, 7] follow the Embedding&MLP paradigm: large-scale sparse input features are firstly mapped into low dimensional embedding vectors, and then concatenated together to fed into the multilayer perceptron (MLP) to learn the nonlinear relations among features. Wide&Deep [5] combines wide linear models and deep neural networks for recommender systems. Some work [2, 12, 28, 29] focus on the feature interaction problem. They can be regarded as complement work with our approach. State-of-the-art methods have found the effectiveness of modeling users' historical behaviors for CTR prediction [8, 10, 16, 17, 26, 27, 31, 40, 41]. DIN [41] notices that a user may have multiple interests and uses attention mechanism to learn the representation of user interests from historical behaviors with respect to a certain candidate item. It achieves better performance than the simple Embedding&MLP method [7]. To further consider the sequential information in users' click sequence, DIEN [40] uses two layers of GRU to model the click sequence and capture the evolution of the user's interest. HUP [10] exploits Pyramid Recurrent Neural Networks to model users' hierarchical interests in categories and items. Recently, some works [3, 8, 18, 39] attempt to exploit Self-Attention Neural Networks to model user's behavior sequence. However, existing methods usually focus on modeling a single type of user's behavior sequence or consider a single objective. How to effectively model users' multiple types of behaviors sequences on items for multiple objectives in industrial Recommender Systems still remains as an open problem.

2.2 Multi-task Learning for Recommendation

Many existing multi-task recommender systems are designed for specific types of applications or features. Lu et al. [22] jointly learns to perform rate prediction using matrix factorization and recommendation explanation using sequence to sequence learning on textual data. Wang et al. [33] proposes a multi-objective evolutionary algorithm for long tail items recommendation to suggest accurate and novel items. General Multi-task ranking systems commonly follows a Shared-bottom [38] architecture, where different tasks share the same bottom layers (*i.e.,* multilayer perceptron layers). However, the hard-parameter sharing mechanisms may harm the learning of multiple objectives when correlation between task is low. To better model the relations and conflicts among multiple objectives, recently Multi-gate Mixture-of-Experts (MMoE) [23, 38] is adopted for multi-task ranking and has achieved state-of-the-art performance. However, existing work usually applying the MoE layer on the input layer [23] or lower-level shared multi perceptron layers [38]. In this paper, we propose the idea of applying the MoE layer on the multiple types of interests vectors, which are learnt by multiple Transformers based on corresponding behavior sequences. This is significantly different with existing work. To our best knowledge, this is the first work that demonstrates the effectiveness of applying MMoE on users' multifaceted interest vectors, which are extracted by sequential models based on users' multiple types of behavior sequences, for multi-objective ranking.

2.3 Unbiased Learning to Rank

Ranking models are usually trained based on logs (*e.g.,* users' implicit feedback) in human-interactive systems. The training logs naturally have selection bias. For example, position bias in search rankings strongly influence how many clicks a result receives [1, 14]. Directly using the implicit feedback data for training may affect the learning of ranking models in estimating relevance between users and items, and lead to sub-optimal results [14]. Unbiased Learning to Rank [14] aims to reduce such biases in learning ranking models. Many works [1, 14, 34] have been proposed to model the position bias in search ranking. These methods usually need to learn a separate model to estimate the propensity scores [1, 14]. Recently, there are some initial work [38] that attempts to learn a single Deep Neural Network based model that can simultaneously solve the click prediction and bias modeling problems in Recommender Systems. This approach is more suitable for industrial Recommender Systems where the training data distribution (*e.g.,* users' interests or behaviors, item popularities) may change frequently.

3 PROBLEM FORMULATION

The Multi-objective Ranking problem in E-commerce Recommender Systems. Given a set of M candidate items, the Multi-objective Ranking problem aims to predict the ranking score of each candidate item based on multiple objectives. In this paper, we consider two objectives: Click-through Rate (i.e. click prediction) and Click-through Conversation Rate (i.e. order prediction), which denote the probability of clicking and purchasing the item respectively. It should be noted that, Click-through Conversation Rate is the product of Click-through Rate and Conversation Rate [24].

4 METHOD

In this section, we introduce DMT, a Deep Multifaceted Transformers based framework for multi-objective ranking in E-commerce Recommender Systems. As illustrated in Figure 2, DMT is composed of Deep Multifaceted Transformers, Multi-gate Mixture-of-Experts (MMoE) layers and a Bias Deep Neural Network.

4.1 Input and Embedding Layers

The inputs of DMT can be divided into two parts: categorical features and dense features.

4.1.1 Categorical features. The most useful categorical features in e-commerce recommender systems are users' diverse behaviors.

Users' diverse behavior sequences. In E-commerce portals, the uses usually have rich and diverse behaviors, such as click products, add products to cart, order products and so on. The sequence of each type of users' behaviors is represented by a variable-length sequence of items. Given a target user u, the input of our model is the target item and the user's multiple types of behavior sequences $S = \langle x_1, x_2, \ldots, x_T \rangle$, where T is the length of the sequence. The ith element $x_i = (t_i, p_i)$ indicates that u performs a behavior on the item p_i at the time t_i. In this paper, we consider three types of users' behavior sequences: click sequences S_c, add to cart sequences S_a and order sequences S_o, which are users' main behaviors in e-commerce sites.

Embedding Layer. For each product p_i, we use its product id p_i, category id c_i, brand id b_i and shop id s_i information to represent it. As previous did [7, 41], we use the embedding layer to transform the high dimensional sparse ids into low dimensional dense representations. Specifically, the embedding layer firstly uses embedding tables of products, categories, brands and shops to transform the sparse ids p_i, c_i, b_i, s_i into low-dimensional dense vectors (i.e., e_{p_i},e_{c_i},e_{b_i},e_{s_i}) respectively and then concatenates these vectors into a single embedding vector e_i. The embedding tables are initialized as random numbers and jointly learnt with our model.

4.1.2 Dense features. The last generation Recommender System in JD.com exploits GBDT [9] to learn the ranking models based on some dense features, which have been designed and improved for more than five years. We empirically find that it will bring significant gains by integrating these dense features into the DNN model. In the JD Recommender System, we use 615 dense features, which can mainly be divided into three types: item profile features (*e.g.,* number of clicks, CTR, CVR, rating) , use profile features (*e.g.,* purchase power, preferred categories and brands), user-item matching features (*e.g.,* whether the item matches the user's gender or age) and user-item interaction features (*e.g.,* number of clicks on the category of the item within a time window). As Deep Neural Networks are sensitive to the scaling of their inputs, we use the Z-score Normalization method to normalize the dense features.

4.2 Deep Multifaceted Transformers Layer

4.2.1 Deep Multifaceted Transformers. To capture each user's multifaceted interest, we use three separate Deep Interest Transformers (they have different parameters) to model the user's click sequence, cart sequence and order sequence, and learn the user's short-term, middle-term and long-term interest vectors respectively.

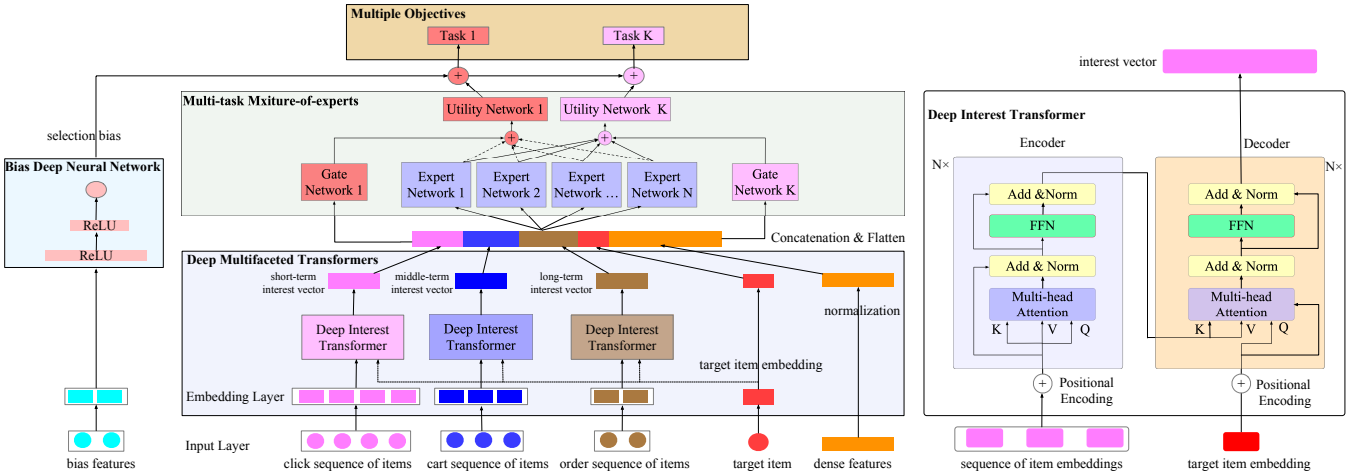

Figure 2: The architecture of our framework DMT. It utilizes Deep Multifaceted Transformers (bottom), which is consisted of multiple Deep Interest Transformers (right), to extract users' multifaceted interests from their diverse behavior sequences, exploits Multi-gate Mixture-of-Experts (MMoE) (top) to simultaneously optimize multiple objectives, and uses a Bias Deep Neural Network (left) to reduce the bias in training data.

The basic idea is that users' multiple types of behavior sequences on items (*e.g.,* click, add to cart and order) are significantly different and they have different timescales. For example, a user may have many click behaviors but usually have only a few cart or order behaviors in a short time range (*e.g.,* within a week). In this paper, for each user, the click sequence is her recent 50 clicked products within 7 days, the cart or order sequence is her recent 10 carted or ordered products within a year.

4.2.2 Deep Interest Transformer. For each behavior sequence, we exploit Deep Interest Transformer (right side in Figure 2) to model user's real-time interest and represent it as an interest vector. In the Deep Interest Transformer, the encoder models the relationships among items in the sequence, and the decoder learns user's interest vector corresponding to the target item. The encoder and decoder in the interest extractor are both based on self-attention blocks.

Self-attention blocks. Self-attention [30], which is an attention mechanism relating different positions of a single sequence in order to compute a new representation of the sequence, has achieved state-of-the-art performance for sequence modeling in many tasks [3, 30]. An attention function mappings a query and a set of key-value pairs to an output, which is a weighted sum of the values, where the weight assigned to each value is computed based on the query and corresponding key. The Self-attention block uses the Scaled Dot-Product Attention defined in Equation 1:

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

where $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ present the queries, keys and values respectively, d_k, d_k and d_v are the dimensions of the queries, keys and values. Each Self-attention block is consisted of Multi-head Self-attention Layers and Point-wise Feed-Forward Networks.

(1) Multi-head Self-attention Layers. To capture the relationships between queries and keys from different subspaces [30], we use multiple heads self-attention. Firstly, it linearly projects the queries $\mathbf{Q} \in \mathbb{R}^{T_q \times d}$, keys $\mathbf{K} \in \mathbb{R}^{T_k \times d}$ and values $\mathbf{V} \in \mathbb{R}^{T_k \times d}$ for h times with different linear projection matrices and performs Scaled Dot-Product Attention on the results to yield each head. Then, it concatenates the results from each head and projects them to the final values with the matrix $\mathbf{W}^O$. The process can be formulated as follows:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)\mathbf{W}^O$$
$$head_i = Attention(Q\mathbf{W}_i^{\mathbf{Q}}, K\mathbf{W}_i^{\mathbf{K}}, V\mathbf{W}_i^{\mathbf{V}})$$

where h is the number of heads, $\mathbf{W}_i^Q \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d \times d_k}$ and $\mathbf{W}_i^V \in \mathbb{R}^{d \times d_v}$ are parameter projection matrices, $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d}$ and $d_k = d_v = \frac{d}{h}$. The output of the Multi-head Self-attention Layers has the dimension of $\mathbb{R}^{T_q \times d}$.

(2) Point-wise Feed-Forward Networks. To increase the representation ability of the model, each Self-attention block applies fully connected feed-forward networks (FFN) to each position in the output of the Multi-head Self-attention Layers separately and identically. As shown in Equation (2), the networks consists of two linear transformations with a ReLU activation in between:

$$FFN(x) = ReLU(x\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \tag{2}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d * d_f}$, $\mathbf{W}_2 \in \mathbb{R}^{d_f * d}$, $\mathbf{b}_1 \in \mathbb{R}^{d_f}$ and $\mathbf{b}_2 \in \mathbb{R}^d$.

Positional encoding. For a behavior sequence $S = \langle x_1, x_2, \ldots, x_T \rangle$, where T is the length of the sequence, and $x_i = (t_i, p_i)$ indicates that the user performs a behavior on the item p_i at the time t_i. To model the positional information in the behavior sequence, we investigate two methods for positional encoding:

- Sinusoidal Positional Embedding (**pos_sincos**) [15, 30]. This method encodes the positions $1, 2, ..., T$ in the sequence into embeddings using sine and cosine functions.

- Learned Positional Embedding (**pos_learn**) [30]. It directly learns the positional embeddings in the model training stage and adds the positional embeddings to corresponding item embeddings in the sequence.

Encoder. The encoder applies a self-attention block on the the embeddings of the behavior sequence and allows each item in the sequence to attend over all items in the input sequence. The self-attention mechanism models the mutual relationships between every two items in the user's behavior sequence, and can better learn the user's interest from historical behaviors.

Decoder. As a user may have diverse interest [41], the decoder uses the target item as query and the output of the encoder as both keys and values. It learns the attention score between the target item and each item in the historical sequence, and learns a unique user interest vector for each target item. The interest vector varies over different target items, improving the representation ability of the model.

4.3 Multi-gate Mixture-of-Experts Layers

In e-commerce recommender systems, there are usually multiple objectives, such as CTR, CVR, GMV and so on. The ranking system should be able to learn and estimate multiple types of users' utilities and combines these estimations to compute a final utility ranking score. The multiple objectives may have complex relationships (*e.g.,* independent, related or conflict) with each other, and the commonly used Shared-bottom [23] architecture may harm the learning of multiple objectives. To capture the relation and conflict of multiple tasks, we adopt Multi-gate Mixture-of-Experts (MMoE) [38] for multi-objective ranking. DMT applies MMoE on top of the concatenation of outputs from the Deep Multifaceted Transformers Layer and the dense features. It uses N expert networks, which are all multi-layer perceptrons with ReLu activations, to model the input x respectively and get the outputs of each expert, which are denoted as $e_1(x), e_2(x), ..., e_N(x)$. For each task k, as shown in Equation 3, firstly, it exploits a gating network ${NN_G}^k$ to learn the weights of each expert as $w^k = (w_1^k, w_2^k, ..., w_N^k) \in \mathbb{R}^N$. Secondly, it calculates the weighted sum of experts outputs as $f^k(x)$. Finally, it feds $f^k(x)$ into a utility network ${NN_U}^k$ to get $u_k \in \mathbb{R}^1$, the utility for task k. The gating networks and utility networks are implemented by multi-layer perceptrons, and their parameters are different for each task.

$$\begin{aligned} w^k &= softmax({NN_G}^k(x)) \\ f^k(x) &= \sum_{i=1}^{N} w_i^k e_i(x) \\ u_k &= {NN_U}^k(f^k(x)) \end{aligned} \tag{3}$$

4.4 Bias Deep Neural Network.

Ranking models are usually trained using implicit feedback, which is biased because the displayed items are generated from the existing ranking system and the user have selection bias (*e.g.,* position bias) [38]. As shown in Figure 1, the user can scroll down and browse products in many pages. In this paper, we investigate two types of selection bias in e-commerce Recommender Systems: Position bias and Neighboring bias.

- **Position bias.** The "position bias" means that users tend to click items which are displayed closer to the top of the list. The position of each item can be defined as the index number or page number in the screen. We denote them as "Position_index" and "Position_page" respectively. For example, in Figure 1, the page numbers of the four products are both 1 because they are all in the first page, and the index numbers of them are 1,2,3,4. We set the maximum number of page and index as 100 and 400 respectively.
- **Neighboring bias.** The "neighboring bias" means that the probability of clicking a item may be influenced by its neighboring products.

DMT uses a Bias Deep Neural Network to model the selection bias. The input of the networks are bias features. For the position bias, the input is the number of index ("Position_index" bias) or page ("Position_page" bias) of the target item. For the neighboring bias, the input is the categories of the target item and its nearest K neighboring items. We experimentally set $K = 6$. We embed the sparse bias features into low dimension vectors and fed them into multi-layer perceptrons with Relu activations. Giving the bias feature x_b, the selection bias for the target item is $y_b \in \mathbb{R}^1$:

$$y_b = NN_B(x_b) \tag{4}$$

where NN_B is the Bias Deep Neural Network.

4.5 Model Training and Prediction.

4.5.1 Training. In the training stage, for each task k, the prediction score y_k is calculated by applying the sigmoid frunction on the sum of the utility logit u_k from the Multi-task Learning Layers and the bias logit y_b from the Bias Deep Neural Network. The multiple tasks are both classification problems, so we use the cross entropy loss function for each task. The total loss L is the weighted sum of losses in multiple objectives:

$$\begin{aligned} y_k &= \sigma(u_k + y_b) \\ L_k &= -\frac{1}{N}\sum_{i\in 1}^{N}(y_i log y_k + (1-y_i) log(1-y_k)) \\ L &= \sum_{k\in 1}^{N} \lambda_k L_k \end{aligned} \tag{5}$$

where σ is the sigmoid function, N is the size of training set, $y_i \in \{0, 1\}$ is the ground truth label, and λ_k is the weight of losses for each task.

4.5.2 Prediction. In the prediction stage in online serving system, for each task k, the utility score y_k is simply calculated by applying the sigmoid function on the utility score u_k from the Multi-task Learning Layers. The final online ranking score $\hat{y}$ for the target item is the weighted utility scores for each task:

$$\begin{aligned} \hat{y}_k &= \sigma(u_k) \\ \hat{y} &= \frac{\sum_{k\in 1}^{N} w_k \hat{y}_k}{\sum_{k\in 1}^{N} w_k} \end{aligned} \tag{6}$$

where w_k is the weight of ranking score for task k. We use offline grid search and online A/B testing [38] to choose the best values of w_k.

Table 1: Performance of different methods for Click and Order prediction. "$*$" indicates the statistically significant improvements (i.e., p-value < 0.01) over the best baseline.

Model	Dense features	Click Prediction				Order Prediction			
		AUC	RelaImpr	Precision@4	MRR@4	AUC	RelaImpr	Precision@4	MRR@4
DNN(Base)	No	0.6132	0.00%	0.2494	0.3776	0.6093	0.00%	0.1364	0.2465
DIN	No	0.6322	16.78%	0.2592	0.3960	0.6297	18.66%	0.1399	0.2535
DIEN	No	*0.6429*	26.24%	0.2645	0.4069	*0.6370*	25.34%	0.1410	0.2545
DMT$_{trans}$	No	0.6458	28.80%	0.2683	0.4172	0.6397	27.81%	0.1447	0.2598
DMT	No	**0.6624***	**43.46%***	**0.2793***	**0.4426***	**0.6697***	**55.26%***	**0.1509***	**0.2909***
GBDT	Yes	0.6590	-6.64%	0.2786	0.4405	0.6537	-7.63 %	0.1473	0.2765
DNN(Base)	Yes	0.6703	0.00%	0.2833	0.4494	0.6664	0.00%	0.1500	0.2796
DIN	Yes	0.6715	0.70%	0.2841	0.4502	0.6669	0.30%	0.1502	0.2807
DIEN	Yes	*0.6729*	1.53%	0.2847	0.4515	*0.6674*	0.60%	0.1523	0.2824
DMT$_{trans}$	Yes	0.6741	2.23%	0.2862	0.4543	0.6695	1.86%	0.1549	0.2847
DMT	Yes	**0.6855***	**8.93%***	**0.2896***	**0.4627***	**0.6866***	**12.14%***	**0.1561***	**0.3075***

Table 2: Statistics of the Datset

Type	Total Samples	Impressions	Clicks	Orders
Train	667,907,650	622,596,211	43,876,602	1,434,837
Test	105,444,671	98,732,799	6,477,409	234,463

5 EXPERIMENTAL SETTINGS

5.1 Dataset

We conduct our research on JD RecSys Dataset, a large scale industrial dataset that is collected from the logs in the Recommender System in JD. One week's samples is used for training and samples of the following day is used for testing. The statistics of the dataset is shown in Table 2.

5.2 Baselines

- **GBDT**. GBDT, which is a widely used model for industrial recommender systems, was used as the last generation recommender systems in JD for more than five years.
- **DNN**. DNN follows the Embedding&MLP architecture, and it is one of the most successful deep learning based model for industrial Recommender Systems.
- **DIN**. DIN exploits attention mechanism to learn the representation of users' interest from historical behaviors with respect to a certain ad.
- **DIEN**. DIEN uses two layers of GRU to model user's behavior sequences, captures the evolution of user's interest, and achieves state-of-the-art performance for CTR prediction.
- **DMT$_{trans}$**. It is a variant of DMT where the Bias Deep Neural Network and MMoE are not used.

5.3 Evaluation Metrics

To evaluate the effectiveness of the methods, we compare them on two tasks: click prediction and order prediction, which aim to predict whether the user will click or order the target item respectively. For offline A/B Testing, we use four widely used metrics AUC [41], RelaImpr [41], Precision@K and MRR@K [6]. For online A/B Testing, we use three core metrics in e-commerce sites: CTR, CVR and GMV [41].

6 EXPERIMENTAL RESULTS

6.1 Comparison with Baselines

Table 1 shows the experimental results of different methods for the click prediction and order prediction tasks. All experiments are repeated 5 times and the averaged results are reported. Firstly, to investigate the effectiveness of our method on model users' behavior sequences, we don't use the dense features and demonstrate the results in the upper part of the table. Secondly, to investigate whether the existing dense features for our previous GBDT model can help improve the model's performance, we further incorporate them in the models, and demonstrate the results in the lower part of the table. From this table, we can find that: (1) DIN performs better than DNN model by utilizing the attention mechanism, and DIEN can achieve better performance than DIN by further modeling the sequential information in user's historical sequence. (2) Our method DMT achieves superior performance compared with GBDT and deep network based models including DNN, DIN and DIEN. When the dense features are used, DMT achieves 0.0126 absolute AUC gain and 7.4% RelaImpr over DIEN for click prediction, and 0.0192 absolute AUC gain and 11.54% RelaImpr over DIEN for order prediction. This is a significant improvement for industrial applications where 0.1% absolute AUC gain is remarkable [24, 41]. (3) The improvements of DIEN, DIEN, DMT over DNN(Base) become smaller when the dense features are used. The reason is that the the dense features, which have been designed and improved for more than five years, already contain about 200 dense features to model the information in users' behavior sequences.

6.2 Effectiveness of Components in DMT

To investigate the effectiveness of components in DMT, we conduct extensive ablation study.

6.2.1 Deep Multifaceted Transformers. Firstly, we investigate how different Positional embedding methods (described in 4.2.2) influence the performance of the Deep Multifaceted Transformers,

and demonstrate the results in Table 3, where we use neither Multi-task learning nor Bias Deep Neural Network. From this table, we can find that: The Learned Positional Embedding method achieves the best performance compared with other methods. So we use this method in the experiments.

Table 3: Performance of position embedding in DMT.

Model	behaviors	position embedding	Click AUC	Order AUC
DMT	click+cart+order	no	0.6435	0.6279
DMT	click+cart+order	pos_sincos	0.6396	0.6344
DMT	click+cart+order	pos_learn	**0.6458**	**0.6397**

Secondly, we investigate whether modeling multiple types of behavior sequences is beneficial. From the top three lines results in Table 4, we find that simultaneously modeling modeling cart sequence can achieve better performance than singly modeling the click sequence. However, if we further adding the order sequence, the performance for click prediction and order prediction doesn't increase further. We empirically find that: For a product that have a long repurchase period (*i.e.,* computer), a user will tend to click but not purchase the product again in short time after buying it. For a product that have a short repurchase period (*i.e.,* milk), the user may both click and purchase it again in short time. The order sequence may lead to the conflict between click prediction and order prediction and disturb the information in the click or cart sequence.

6.2.2 Multi-task Learning. We study how different Multi-task learning methods(*i.e.,* Shared-bottom [38] and MMoE) will influence the performance of the model, and demonstrate the results in Table 4. From this table, we can conclude that: (1) Modeling multiple types of behaviors will bring better performance than singly modeling the click sequence. (2) Shared-bottom can improve the prediction performance. (3) MMoE achieves the best performance for both click prediction and order prediction. This demonstrates that MMoE can better model the relationships and conflicts between multiple objectives and users' diverse behavior sequences.

To demonstrate how MMoE helps multi-objectives optimization, in Figure 3, for each task, we plot the average weights of experts,

Table 4: Performance of multi-task learning in DMT.

Model	behaviors	Multi-task	Click AUC	Order AUC
DMT	click	No	0.6443	0.6393
DMT	click+cart	No	0.6460	0.6410
DMT	click+cart+order	No	0.6458	0.6397
DMT	click	Shared-bottom	0.6567	0.6568
DMT	click+cart	Shared-bottom	0.6593	0.6622
DMT	click+cart+order	Shared-bottom	0.6563	0.6622
DMT	click	MMoE	0.6577	0.6603
DMT	click+cart	MMoE	0.6595	0.6628
DMT	click+cart+order	MMoE	**0.6608**	**0.6642**

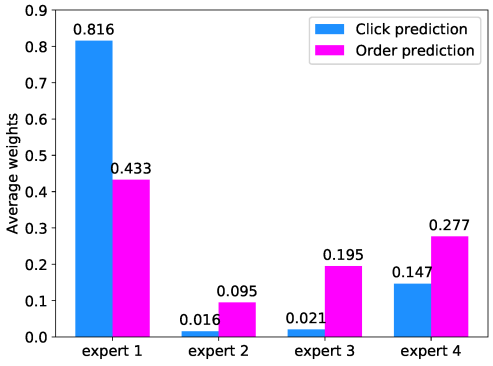

Figure 3: Expert Utilization for Multiple Tasks in MMoE.

Table 5: Performance of Bias Deep Neural Network in DMT.

Model	Bias features	Multi-task	Click AUC	Order AUC
DMT	No	No	0.6742	0.6688
DMT	No	MMoE	0.6846	0.6857
DMT	position_index	MMoE	0.6828	0.6862
DMT	position_page	MMoE	0.6840	0.6865
DMT	neighboring	MMoE	**0.6855**	**0.6866**

which are the outputs of the gating networks. We can find that: (1) The Click prediction task is mainly influenced by the expert 1 and expert 4; while the Order prediction task is mostly influenced by the expert 1, expert 3 and expert 4. (2) Each expert has different influence on the two tasks. This demonstrates that MMoE can effectively modularize input information (*i.e.,* both diverse interest vectors from multiple types of behaviors and dense features) into experts using the gating networks, and capture the relation and conflict of multiple tasks.

6.2.3 Bias Deep Neural Network. Table 5 shows how different types of bias features will influence the performance of DMT. From this table we can find that: Modeling the neighboring bias will bring better performance than modeling the position bias features, where the position is defined by the index or page number in the screen.

6.3 Online A/B Testing

We conduct online A/B testing for one month and examine online metrics such as CTR, CVR and GMV. The results are demonstrated in Table 6, where the bias features used is the neighboring bias. From this table, we can find that: (1) Modeling the bias information in the implicit feedback brings 0.6%, 2.3% and 1.7% gains in CTR, CVR and GMV respectively. This demonstrates the effectiveness of the Bias Deep Neural Network. (2) DMT outperforms state-of-the-art method DIEN by 4.5%, 4.6% and 8.0% in CTR, CVR and GMV respectively. (3) Compared with the GBDT, the last generation model in our recommender systems, DMT improves the CTR, CVR and GMV by 18.8%, 19.2% and 17.9% respectively, which are the largest improvements in JD Recommender Systems over past three

years. DMT has been deployed online successfully and serves the main traffic in JD Recommender Systems.

Table 6: Performance in online A/B Testing in JD Recommender Systems. "$*$" indicates the statistically significant improvements (i.e., p-value < 0.01) over the baseline.

Model	BDNN	CTR	CVR	GMV
GBDT(base)	No	+0.00%	+0.00%	+0.00%
DIEN	No	+14.3%	+14.6%	+11.9%
DMT	No	+18.2%	+16.9%	+16.2%
DMT	Yes	**+18.8%***	**+19.2%***	**+17.9%***

7 CONCLUSION

In this paper, we propose DMT, which exploits Deep Multifaceted Transformers to model users' diverse behavior sequences, utilizes Multi-gate Mixture-of-Experts to jointly optimize multi-objectives in e-commerce, and uses the Bias Deep Neural Networks to reduce the select bias in implicit feedback. We conduct extensive experiments and demonstrate the effectiveness of DMT for multi-objective ranking in large-scale e-commerce Recommender Systems. Online A/B testing in JD Recommeder Systems further demonstrates that DMT can achieve substantial improvements in business.

REFERENCES

[1] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019. A General Framework for Counterfactual Learning-to-Rank. In *SIGIR'19*. 5–14.

[2] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent cross: Making use of context in recurrent recommender systems. In *WSDM'18*. 46–54.

[3] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior sequence transformer for e-commerce recommendation in Alibaba. In *DLP-KDD'19*. 1–4.

[4] Weijian Chen, Yulong Gu, Zhaochun Ren, Xiangnan He, Hongtao Xie, Tong Guo, Dawei Yin, and Yongdong Zhang. 2019. Semi-supervised user profiling with heterogeneous graph attention networks. In *IJCAI'19*. 2116–2122.

[5] 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. In *DLRS'16*. 7–10.

[6] Aleksandr Chuklin, Pavel Serdyukov, and Maarten De Rijke. 2013. Click model-based information retrieval metrics. In *SIGIR'13*. 493–502.

[7] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys'16*. 191–198.

[8] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. In *IJCAI'19*. 2301–2307.

[9] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[10] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, and Dawei Yin. 2020. Hierarchical User Profiling for E-commerce Recommender Systems. In *WSDM'20*. 223–231.

[11] Yulong Gu, Jiaxing Song, Weidong Liu, and Lixin Zou. 2016. HLGPS: a home location global positioning system in location-based social networks. In *ICDM'16*. 901–906.

[12] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *IJCAI'17*. 1725–1731.

[13] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *ADKDD'14*. 1–9.

[14] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *WSDM'17*. 781–789.

[15] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM'18*. 197–206.

[16] Chenyi Lei, Shouling Ji, and Zhao Li. 2019. TiSSA: A Time Slice Self-Attention Approach for Modeling Sequential User Behaviors. In *WWW'19*. 2964–2970.

[17] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-interest network with dynamic routing for recommendation at Tmall. In *CIKM'19*.

[18] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM'20*. 322–330.

[19] Xiao Lin, Hongjie Chen, Changhua Pei, Fei Sun, Xuanji Xiao, Hanxiao Sun, Yongfeng Zhang, Wenwu Ou, and Peng Jiang. 2019. A pareto-efficient algorithm for multiple objective optimization in e-commerce recommendation. In *RecSys'19*. 20–28.

[20] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.

[21] Yiding Liu, Yulong Gu, Zhuoye Ding, Junchao Gao, Ziyi Guo, Yongjun Bao, and Weipeng Yan. 2020. Decoupled Graph Convolution Network for Inferring Substitutable and Complementary Items. In *CIKM'20*.

[22] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Why I like it: multi-task learning for recommendation and explanation. In *RecSys'18*. 4–12.

[23] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *KDD'18*. 1930–1939.

[24] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *SIGIR'18*. 1137–1140.

[25] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *KDD'15*. 785–794.

[26] Wentao Ouyang, Xiuwu Zhang, Li Li, Heng Zou, Xin Xing, Zhaojie Liu, and Yanlong Du. 2019. Deep spatio-temporal neural networks for click-through rate prediction. In *KDD'19*. 2078–2086.

[27] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on long sequential user behavior modeling for click-through rate prediction. In *KDD'19*. 2671–2679.

[28] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *KDD'16*. 255–262.

[29] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *CIKM'19*. 1161–1170.

[30] 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'17*. 5998–6008.

[31] Huizhao Wang, Guanfeng Liu, Yan Zhao, Bolong Zheng, Pengpeng Zhao, and Kai Zheng. 2019. DMFP: A Dynamic Multi-faceted Fine-Grained Preference Model for Recommendation. In *ICDM'19*. 608–617.

[32] 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'18*. 839–848.

[33] Shanfeng Wang, Maoguo Gong, Haoliang Li, and Junwei Yang. 2016. Multi-objective optimization for long tail recommendation. *Knowledge-Based Systems* (2016), 145–155.

[34] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to rank with selection bias in personal search. In *SIGIR'16*. 115–124.

[35] Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly, Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, et al. 2016. Ranking relevance in yahoo search. In *KDD'16*. 323–332.

[36] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *KDD'18*. 974–983.

[37] Jiaxuan You, Yichen Wang, Aditya Pal, Pong Eksombatchai, Chuck Rosenburg, and Jure Leskovec. 2019. Hierarchical temporal convolutional networks for dynamic recommender systems. In *WWW'19*. 2236–2246.

[38] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *RecSys'19*. 43–51.

[39] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. Atrank: An attention-based user behavior modeling framework for recommendation. In *AAAI'18*.

[40] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI'19*. 5941–5948.

[41] 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'18*. 1059–1068.

[42] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement Learning to Optimize Long-term User Engagement in Recommender Systems. In *KDD'19*. 2810–2818.

[43] Lixin Zou, Long Xia, Yulong Gu, Xiangyu Zhao, Weidong Liu, Jimmy Huang, and Dawei Yin. 2020. Neural Interactive Collaborative Filtering. In *SIGIR'20*. 749–758.