Adaptive Disentangled Transformer for Sequential Recommendation

Yipeng Zhang
Department of Computer Science and Technology, Tsinghua University
zhang-yp22@mails.tsinghua.edu.cn

Xin Wang*
Department of Computer Science and Technology, BNRist, Tsinghua University
xin_wang@tsinghua.edu.cn

Hong Chen
Department of Computer Science and Technology, Tsinghua University
h-chen20@mails.tsinghua.edu.cn

Wenwu Zhu*
Department of Computer Science and Technology, BNRist, Tsinghua University
wwzhu@tsinghua.edu.cn

ABSTRACT

Sequential recommendation aims at mining time-aware user interests through modeling sequential behaviors. Transformer, as an effective architecture designed to process sequential input data, has shown its superiority in capturing sequential relations for recommendation. Nevertheless, existing Transformer architectures lack explicit regularization for layer-wise disentanglement, which fails to take advantage of disentangled representation in recommendation and leads to suboptimal performance. In this paper, we study the problem of layer-wise disentanglement for Transformer architectures and propose the Adaptive Disentangled Transformer (ADT) framework, which is able to adaptively determine the optimal degree of disentanglement of attention heads within different layers. Concretely, we propose to encourage disentanglement by requiring the independence constraint via mutual information estimation over attention heads and employing auxiliary objectives to prevent the information from collapsing into useless noise. We further propose a progressive scheduler to adaptively adjust the weights controlling the degree of disentanglement via an evolutionary process. Extensive experiments on various real-world datasets demonstrate the effectiveness of our proposed ADT framework.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Transformer, Disentangle, Sequential Recommendation

ACM Reference Format:
Yipeng Zhang, Xin Wang, Hong Chen, and Wenwu Zhu. 2023. Adaptive Disentangled Transformer for Sequential Recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599253

*Corresponding Authors.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08.
https://doi.org/10.1145/3580305.3599253

1 INTRODUCTION

Sequential recommender systems, which aim to accurately recommend the next item to a target user through modeling the sequential patterns of user interests, play an important role in facilitating our online experiences [23, 28, 48]. The major task is to find an appropriate way to represent the sequential user behaviors such as click, rate, favorite, etc. Transformer [50], as one of the most widely adopted architectures, has shown its power in representing sequential input data, thus capable of capturing dynamic relations for sequential recommendation [48].

On the one hand, users nowadays have diverse and constantly-changing interests in terms of various aspects in the course of time, thus accurately capturing the information regarding these aspects can help to boost the recommendation performance significantly. On the other hand, obtaining the information capturing various aspects is treated as one major advantage of disentangled representation learning, which is known to be very important for accurately making recommendations [54]. However, it is reported that existing Transformer architectures fail to guarantee the assumption that the widely adopted multi-head attention within different layers can capture the disentangled information related to different aspects of sequence data [10, 35, 45], and lack of explicit regularization for layer-wise disentanglement, which may not take the advantage of disentangled representation in recommendation and may lead to suboptimal performance.

To deal with the issue, in this paper, we investigate the discovery of the optimal degree of disentanglement for Transformer architectures that best serve sequential recommendation, where the attention heads in different layers of the Transformer are disentangled to capture various aspects of user behaviors in an adaptive manner. Nevertheless, learning to discover such a disentangled Transformer architecture adaptively for a given recommendation dataset is largely unexplored in the literature, posing the following two challenges:

(1) Transformer architectures or backbones may have different numbers of layers with different numbers of attention heads designed with various functions. Therefore, it is challenging to determine which part of the encoder and decoder layer in the backbone should be disentangled, and how we can encourage the disentanglement of attention heads to capture various aspects of user interests given a Transformer architecture.

(2) Different datasets may indicate different recommendation tasks imposing user interests with various aspects, and the capabilities of disentanglement differ among different layers, which requires an adequate Transformer architecture to adaptively adjust the degree of disentanglement given a recommendation task. Thus, it remains another challenge to adaptively discover the optimal degree of disentanglement for attention heads in different layers to achieve the best recommendation performance on various datasets.

To tackle these challenges, we propose the Adaptive Disentangled Transformer (ADT) framework, which simultaneously i) disentangles attention heads within different layers of a given Transformer architecture, and ii) determines the optimal degree of disentanglement for attention heads in different layers adaptively in order to achieve the best recommendation performance. Particularly, to disentangle the attention heads, our ADT framework requires the independence constraint via mutual information estimation over attention heads, utilizing an auxiliary classification objective with generated pseudo labels to encourage the disentanglement. To further prevent the learned disentangled information from collapsing into noisy signals, the proposed ADT framework employs another auxiliary objective with a decoder to reconstruct the input data. Moreover, we propose a progressive scheduler which is able to adaptively adjust the auxiliary weights controlling the degree of disentanglement of different layers via an evolutionary process. We also design a supernet to accelerate the searching progress for discovering the optimal weights for different auxiliary objectives. Extensive experiments including ablation studies are conducted to demonstrate that our proposed ADT framework can be applied to different Transformer architectures to significantly outperform state-of-the-art baselines, and verify that it is necessary for Transformer architectures to possess different degrees of disentanglement in different layers that can best serve the task of sequential recommendation. The code is available at https://github.com/defineZYP/ADT. Our main contributions are summarized as follows:

- We point out the importance of disentangling attention and investigate adaptive disentanglement of attention heads in Transformer architectures for the sequential recommendation, to the best of our knowledge, for the first time.
- We propose the Adaptive Disentangled Transformer (ADT) framework capable of simultaneously disentangling attention heads in a given Transformer, as well as determining the optimal degree of disentanglement for attention heads in different layers adaptively such that the best recommendation performance can be achieved.
- We conduct extensive experiments over various real-world datasets to show that the proposed ADT framework can be applied to different Transformer architectures to significantly outperform state-of-the-art baselines. Ablation studies also verify that Transformer may need different degrees of disentanglement for attention heads within different layers to best serve sequential recommendation.

2 RELATED WORK

In this section, we review related works on sequential recommendation, attention mechanism, as well as disentangled representation learning. We also briefly discuss relevant works on neural architecture search, which has close relations with the evolutionary procedure in the progress scheduler of our proposed ADT framework.

Sequential Recommendation. The purpose of the sequential recommendation is to use historical data from user interactions to predict the next item. Early work on sequential recommendation tasks is generally based on Markov chains. FPMC [44] combines Markov chains and matrix factorization. It generates a transition matrix for each user, allowing the model to capture both temporal information and long-term user preference information. However, FPMC only makes use of the first-order Markov chains. Fossil [19] extends this idea to high-order Markov chains to consider more items. However, as MCs based methods fail to model union-level sequential patterns and fail to allow skip behaviors, Caser [49] introduces convolutional neural networks into the sequential recommendation. Caser considers the embedding matrix of items as an image and learned transitions by using convolution operations. As better models [9, 25, 50] are applied to sequence data, some works adopt them such as RNNs [11, 22, 23, 30, 33, 37, 39] and Transformers [14, 28, 48] to model user behavior sequences. GRU4Rec [23] first applies Gated Recurrent Units to sequence recommendation. And unlike Markov chain and RNN-based methods, the multi-head self-attention mechanism is able to capture information on different aspects from all item-item pairs in the sequence. SASRec [28] and Bert4Rec [48] have achieved great success with the sequential recommendation.

Attention Mechanism and Transformer. Attention mechanism has shown amazing potential for many tasks, e.g., visual recognition [12, 57], image generation [36, 40], machine translation [50, 52] hyper-parameter optimization [56] and neural architecture search [16, 62]. The attention mechanism draws on the human attentional mindset, with the core goal of selecting the information that is more critical to the current task from among the many. And the multi-head attention mechanism allows the model to capture information from different representation sub-spaces at different positions. Several studies [3, 10, 27, 35, 45, 51, 60] have analyzed attention mechanism. Both [51] and [35] find that in the multi-head attention mechanism, there is redundancy in the heads. Bian et al. [3] find that there is a high similarity of attention patterns between different heads in vanilla Transformer and prune some attention heads but get a similar performance. Clark et al. [10] calculate the Jensen-Shannon divergence between attention distributions of different heads. And the result indicates that show that at the shallow layers, the distribution of representations learned by different attention heads differs widely, which means that it is indeed able to capture information on different aspects. However, as the layer becomes deeper, the distributions of representations learned by different attention heads gradually become consistent. These studies suggest that the mechanism of multi-head attention may contradict the original expectations.

Disentangled Representation Learning. Disentangled representation learning aims to learn different aspects of the data, capturing the interpretable representation behind different latent factors [2, 53]. Variational autoencoder (VAE) [29] is one of the representative works of disentangled representation learning which

captures the information of different latent factors by variational inference and an encoder-decoder based architecture. β–VAE[24] balances the representational ability and the disentanglement ability of the model based on VAE. Due to the diversity of purchase interests of users, multi-interests methods [5, 8] and disentangled representation learning have also been applied to recommendation tasks [34, 54, 55, 58, 61] with great success. However, to the best of our knowledge, there is still a lack of research to explicitly introduce layer-wise disentangled representation into the Transformer to adaptively make sequential recommendations.

Neural Architecture Search. Neural architecture search (NAS) is a method to automatically search for neural network architectures without excessive expert knowledge. Existing NAS methods consist of three main components: search space, search strategy, and evaluation method. Depending on the search strategy, NAS can be divided into three categories, reinforcement learning based methods [1, 15, 63, 64], evolutionary algorithm based methods [41, 46] and gradient based methods [7, 31]. However, many of these methods require a large amount of time to search for excellent architecture. For this reason, ENAS [38] proposes a method called supernet to share parameters between the same options in the search space to accelerate the search process. However, parameter sharing leads to the coupling between different candidate architectures, making the NAS method unable to accurately evaluate the performance of these architectures and thus affect the search result. One-shot method [4, 17] is therefore proposed, which decouples the search process into the training stage and search stage to alleviate the problems.

3 METHODOLOGY

In this section, we present the proposed Adaptive Disentangled Transformer. We first give the problem formulation and introduce the general approach to using Transformer for sequential recommendation in the preliminaries. Then, we give a brief overview of the main parts of our method and describe the auxiliary objectives in detail in section 3.2. And we detail our adaptive training strategy in section 3.3.

3.1 Preliminaries

In sequential recommendation, we have the user set $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$, the item set $\mathcal{V} = \{v_1, v_2, ..., v_{|\mathcal{V}|}\}$, and the interactions between the users and items. For each user $u \in \mathcal{U}$, we can organize behaviors of that user into a sequence in chronological order as $S^u = [v_1^u, v_2^u, \ldots, v_T^u]$, where $v_t^u \in \mathcal{V}$ is the item that the user u clicked at time step t and T is the maximum length of the sequence. The purpose of sequential recommendation is to infer the item that the user will most likely interact with at time step $T+1$ based on the past T historical items in the sequence. Next, we will introduce a general approach to the use of Transformer for sequential recommendation.

First, for each item $v_t^u \in S^u$, we need to embed them from the index into the representation space. Transformer has an item embedding matrix $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where each item can be represented as a d dimensional vector z_t^u. Then to model the position information of the sequence, Transformer usually has a positional embedding matrix $\mathbf{P} \in \mathbb{R}^{T \times d}$ that maps the item at time step t in the sequence to p_t. To represent both item attributes and the temporal information, a function $f(\cdot)$ is used to fuse the item embedding and the position embedding as follows,

$$h_t^u = f(z_t^u, p_t), t = 1, 2, ..., T, \tag{1}$$

and then the item sequence of the user u is represented as a hidden representation $h^0 = [h_1^u, h_2^u, ..., h_T^u]$. The form of $f(\cdot)$ is not consistent for different Transformers. The easiest way is to add the two embeddings or concatenate the embeddings and then put them into the following modules. Also, there are other methods during the embedding process. For a unified description, we set $h^0 = \text{Embed}(S^u) \in \mathbb{R}^{T \times d}$.

For sequential recommendation, the Transformer can be seen as a stack of L Transformer encoder layer. And for the l-th encoder layer, the representation h^{l-1} will first be input into the attention module. The most commonly used multi-head attention mechanism can be represented as:

$$\text{Attn}_j(h^{l-1}) = \text{Attention}_j(\mathbf{Q}_j^l, \mathbf{K}_j^l, \mathbf{V}_j^l) = \text{softmax}(\frac{\mathbf{Q}_j^l {\mathbf{K}_j^l}^T}{\sqrt{d}})\mathbf{V}_j^l, \tag{2}$$

$$\mathbf{Q}_j^l = h^{l-1} W_{qj}^l,\ \mathbf{K}_j^l = h^{l-1} W_{kj}^l,\ \mathbf{V}_j^l = h^{l-1} W_{vj}^l, j = 1, 2, \cdots, n \tag{3}$$

where $W_{qj}^l, W_{kj}^l, W_{vj}^l \in \mathbb{R}^{d \times \Delta d}$ are the matrices for queries, keys, and values in the attention module of the j-th head of the l-th encoder layer, $d = \Delta d \times n$ and n is the number of heads. The representations captured by different heads, $[Attn(h^{l-1})_1, \cdots, Attn(h^{l-1})_n]$, represent different aspects of the purchase intentions of that user, which to some extent can be regarded as a kind of disentangled information. And after that, we concatenate the output of all heads as $\text{Attn}(h^{l-1}) = [\text{Attn}_j(h^{l-1})]_{j=1}^n$.

And for computational stability and to accelerate convergence, the skip connection and the LayerNormalization(LN) are used as

$$h_{tmp}^l = h^{l-1} + \text{Attn}(\text{LN}((h^{l-1})). \tag{4}$$

After the attention module, a feedforward network is introduced to model the nonlinear relations as

$$h^l = h_{tmp}^l + \text{FFN}(\text{LN}(h_{tmp}^l)). \tag{5}$$

And after L encoder layers, we get h^L which contains the purchase preference information of user u. A classic way to use this information to make predictions is to use the matrix factorization as $r = h_T^L \mathbf{Z}^T$ with negative sampling and the binary cross-entropy loss, where h_T^L is the last embedding in the embedding sequence h^L, r is a matrix about the relevance of the user u and the items. A high score means a high possibility that the user may interact with. However, for different Transformer backbone, the prediction method and the training objective is different, and we denote $\mathcal{L}_{task}$ as the original objective of the selected Transformer backbone. Using $\mathcal{L}_{task}$ to train the whole Transformer backbone until convergence will result in the final Transformer-based recommender model.

Note that in the traditional Transformer-based recommender models, the representations learned from different heads, $[Attn(h^{l-1})_1$,

$Attn(h^{l-1})_2, \cdots, Attn(h^{l-1})_n]$, which are expected to capture different aspects of the user's preferences, sharing the similar idea with disentanglement. However, they may fail to achieve disentanglement without explicit regularization. Additionally, since previous works [10, 35, 45] indicate that the disentanglement characteristic of Transformer is layer-specific, i.e., only shallow layers need disentangled representation while the deep layers need task-specific representation. Additionally, the layers that need to be disentangled may vary with the tasks and datasets. Deciding in which layer we should add the disentangled regularization and how to discover the optimal degree of disentanglement is another challenging problem. To solve these two problems, we present our proposed Adaptive Disentangled Transformer in the next subsection.

3.2 Adaptive Disentangled Transformer

The overall framework of our Adaptive Disentangled Transformer is shown in Figure 1. To encourage the disentanglement of the attention heads in different layers, we proposed two auxiliary objectives which control the ability to disentangle the intentions and the capability to capture the user's comprehensive diverse interests, respectively. To discover the optimal layer-wise disentanglement, we design an adaptive training strategy for Transformer to automatically adjust the weights of these two auxiliary objectives based on the backbone architecture and the dataset. Next, we respectively describe the two auxiliary objectives and the adaptive training strategy.

Objective 1: The Independence Auxiliary Objective.

We apply this objective to the encoder to guarantee the independence between different heads and thus make the representation more disentangled. The structure of the Transformer-based encoder is consistent with the backbone. They usually have at least one embedding module and an attention module shown in section 3.1. In our method, the behavior of the encoder is the same as the backbone Transformer and the general process is shown in Eqs. (1) to (5).

To distinguish the representations of each head, we reshape $\text{Attn}(h^{l-1})$ to $A^l = [a^l_{1,1}, a^l_{1,2}, ..., a^l_{t,k}, ..., a^l_{T,n}] \in \mathbb{R}^{n \cdot T \times \Delta d}$, where $a^l_{t,k} \in \mathbb{R}^{\Delta d}$ is the hidden representation of head k and at the time step t.

We expect each head to capture different latent factors of the user's purchase intentions, and they are independent of each other. However, since there is no label of intention in the training data, we do not know exactly which kind of information is captured by each head. To solve the problem, we can enhance the relationship between the latent factor k and the learned embedding $a^l_{t,k}$ by maximizing the mutual information between them. Although we do not know the exact meaning of latent factors, each latent factor is independent of the other. Therefore if the representation learned by each head k can be made to depend on a latent factor k, it is possible to make the representation learned by different heads independent.

According to [26, 59], the maximization of mutual information can be converted into the following form. Given that the representation $a^l_{t,k}$ is expected to belong to latent factor k, the regularizer $P^l(k|a^l_{t,k})$ estimates the probability that $a^l_{t,k}$ belongs to the k-th latent factor as

$$P^l(k|a^l_{t,k}) = \text{softmax}(\mathbf{W}^l a^l_{t,k} + b^l), \tag{6}$$

where $\mathbf{W}^l \in \mathbb{R}^{n \times \Delta d}$. For all $a^l_{t,k} \in A^l$, we calculate $P^l(k|a^l_{t,k})$ and obtain a vector $P^l(A^l)$. And thus we can use a classification task to optimize this regularizer. First, we generate an auxiliary label $y \in \{1, 2, ..., n\}$ for every representation learned by different heads according to the position of the head, i.e., for the representation output by the i-th head, we give the auxiliary label $y = i$. And we get an auxiliary label vector $Y^l_f = \{y_{1,1}, y_{1,2}, ..., y_{n,T}\}$ where each $y_{i,t} = i$ means the auxiliary label of head i at the time step t. This regularization is an approximation of mutual information and when the loss of the auxiliary objective becomes small, the representation learned by different heads would be more independent. And the auxiliary objective of the l-th layer can be formulated as:

$$\mathcal{L}^l_{ind} = \text{CrossEntropy}(P^l(A^l), Y^l_f). \tag{7}$$

Objective 2: The Reconstruction Auxiliary Objective.

After disentangling the representation of different heads of the Transformer, we need to ensure that the disentangled representations learned by the different heads are meaningful and contain the user's comprehensive and diverse interests. Inspired by VAE[29], which makes the representation contain rich input information through reconstructing the input data, we introduce the reconstruction objective.

To achieve this, a decoder is used to decode the learned representation. A decoder, as opposed to an encoder, is a process of converting a representation to a specific sequence. Thus, each of its layers corresponds to the layer of the encoder. We reconstruct the input of layer l of encoder h^{l-1} and the output of layer $L-l+1$ of decoder h^{L-l+1}_{dec}, to guarantee the representations learned by the l-th layer of encoder still contain rich information.

The structure of the decoder is similar to [50] and consists of two attention modules and shares the embedding module with the encoder so the input of the first layer of the decoder can be embedded as $h^0_{dec} = \text{Embed}(S^u_{dec})$, where S^u_{dec} is the input sequence of the decoder. For S^u_{dec}, since the decoder is originally used for generation tasks, its final output contains the information of the sequence S^u_{dec} and the information of the item at the next time step. Considering the need for reconstruction, we want the information output by the decoder at the last layer to be similar to the information obtained by the encoder at the first layer. Therefore we take out the first $T-1$ time steps of the input sequence of the encoder as S^u_{dec} and pad it to length T. The token we use for padding is related to the backbone Transformer. For example, in the SASRec backbone, we use token 0 for padding since there is no special need for it. But in the Bert4Rec backbone, we use the token [MASK] for padding since Bert4Rec treats sequential recommendation as a Cloze objective problem and uses the token [MASK] to replace the item that needs to be predicted.

To adapt to the encoder of the selected backbone, the decoder attention modules have the same structure as the encoder. The first attention module computes the self-attention of the input of the decoder as:

$$h^l_{tmp} = h^{l-1}_{dec} + \text{Attn}(\text{LN}(h^{l-1}_{dec})), \tag{8}$$

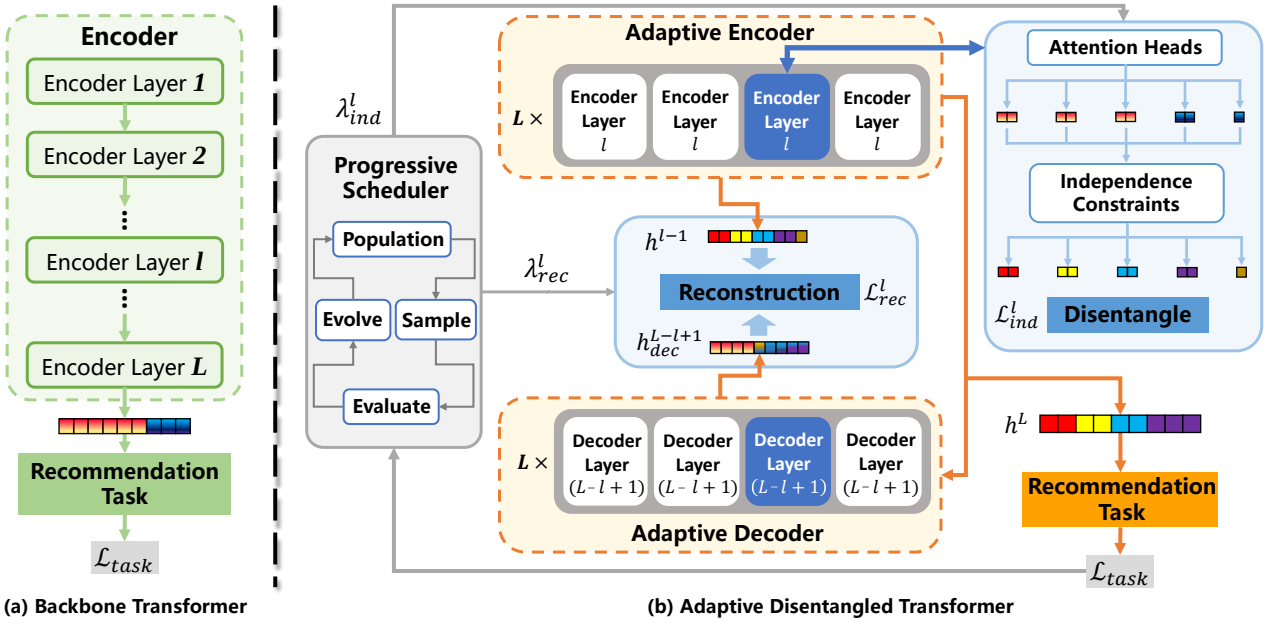


Figure 1: Existing Backbone Transformer V.S. The Proposed Adaptive Disentangled Transformer Framework. (a) An existing backbone Transformer for sequential recommendation. (b) Our proposed Adaptive Disentangled Transformer with the same encoder as the backbone Transformer, which consists of four key parts. 1. Transformer-based encoder: to encode the sequence of items, where the multi-head attention mechanism captures the information of different aspects of the items in the behavior sequence. In this module, we add the first auxiliary independence objective $\mathcal{L}^l_{ind}$ of each layer l to enhance the independence of the learned representations between different heads. 2. Transformer-based decoder: to decode the learned representation, and in this module, we add another reconstruction auxiliary objective $\mathcal{L}^l_{rec}$ of each layer l to make the disentangled representation contain rich information about the input sequence, so that we can capture the user's comprehensive and diverse interests. 3. Downstream recommendation task processing module: which is utilized to conduct the final prediction. 4. Weight scheduler: the core component of the adaptive training strategy, for adaptively adjusting the weights of the auxiliary objectives according to different datasets and backbones.

where h^{l-1}_{dec} is the input of the l-th layer of decoder.

And the second attention module computes the attention between the output of the encoder and the decoder as:

$$h^l_{tmp} = \text{LN}(h^l_{tmp}), \tag{9}$$

$$h^l_{tmp} = h^l_{tmp} + \text{Attention}(h^l_{tmp}W^l_{q2}, h^L W^l_{k2}, h^L W^l_{v2}), \tag{10}$$

$$h^{l+1}_{dec} = h^l_{tmp} + \text{FFN}(\text{LN}(h^l_{tmp})), \tag{11}$$

where $W^l_{q2}, W^l_{k2}, W^l_{v2} \in \mathbb{R}^{d\times d}$ are the matrices of the second attention module, h^L is the output of encoder, and FFN is a feed-forward network. And the auxiliary objective of the l-th layer can be formulated as:

$$\mathcal{L}^l_{rec} = \text{MSELoss}(h^{L-l+1}_{dec}, h^{l-1}). \tag{12}$$

This reconstruction auxiliary objective makes the learned disentangled representation contains rich input information, i.e., the user's sequential behaviors which can reflect his or her comprehensive and diverse interests. Additionally, this regularization can be regarded as an auxiliary task for recommendation, which can improve the representation generalization ability [6][32], suitable for the recommendation task where the data is often sparse.

Training Objective. With the two proposed auxiliary objectives, our final training objective is as follows:

$$\mathcal{L} = \mathcal{L}_{task} + \sum_{l=1}^{L} \lambda^l_{ind}\mathcal{L}^l_{ind} + \sum_{l=1}^{L} \lambda^l_{rec}\mathcal{L}^l_{rec}, \tag{13}$$

where λ^l_{ind} and λ^l_{rec} control how disentangled the representation is and how much input information we want to keep in each layer. Since there are totally $2L$ weight hyper-parameters that are hard to decide, we propose the adaptive training strategy which automatically obtains the optimal value of these parameters and trains the whole Transformer model.

3.3 The Adaptive Training Strategy

We introduce the adaptive training strategy inspired by the neural architecture search (NAS), which trains the model and simultaneously obtains the best weights of each layer. To further accelerate convergence, we propose a supernet that shares the model parameters and only updates part of the specific weights after each training epoch.

For each auxiliary objective, we set the search space of its weight to a continuous interval and divide it into M sub-intervals regarding the order of magnitude to represent the importance of the auxiliary

objective in a certain level as $[\lambda_0, \lambda_1), [\lambda_1, \lambda_2), \ldots, [\lambda_{M-1}, \lambda_M]$. For each sub-interval, we share the parameters in the supernet. And as we have two auxiliary objectives, for each layer in the supernet, there are $M \times M$ identical modules that cover the entire search space like a grid.

To make it easier to search for auxiliary weights with different ranges of values, we use a vector $\mathbf{c} = [c^1_{rec}, c^1_{ind}, ..., c^L_{rec}, c^L_{ind}]$ to represent the sampled sub-network candidate, where $c^l_{rec}, c^l_{ind} \in [0, 1]$ denote the weight of the auxiliary weights of the l-th layer and L is the number of the layers. For $c^l_{rec} \in [\frac{m}{M}, \frac{m+1}{M})$, where $m \in \{0, 1, 2, ..., M-1\}$, we use linear interpolation method to find its true auxiliary weight as:

$$s^l_{rec} = (c^l_{rec} - \frac{m}{M}) \cdot M, \tag{14}$$

$$\lambda^l_{rec} = s^l_{rec} \cdot \lambda_{m+1} + (1 - s^l_{rec}) \cdot \lambda_m, \tag{15}$$

and we do the same for $c^l_{ind} \in [\frac{m}{M}, \frac{m+1}{M})$ to calculate the real auxiliary weight as:

$$s^l_{ind} = (c^l_{ind} - \frac{m}{M}) \cdot M, \tag{16}$$

$$\lambda^l_{ind} = s^l_{ind} \cdot \lambda_{m+1} + (1 - s^l_{ind}) \cdot \lambda_m, \tag{17}$$

In the forward phase of the supernet, we use bi-linear interpolation to find which modules of the grid to use for the computation. We let $h^l_{i,j}$ represent the hidden representation of the output of the module shared by the reconstruction auxiliary objective, whose weight lies in the i-th interval, and the independence auxiliary objective whose weight lies in the j-th interval in the l-th layer. The hidden representation of the l-th layer of the supernet h^l_{super} with $c^l_{rec} \in [\frac{m_{rec}}{M}, \frac{m_{rec}+1}{M}), c^l_{ind} \in [\frac{m_{ind}}{M}, \frac{m_{ind}+1}{M})$ can be formulated as follows:

$$s^l_{rec} = (c^l_{rec} - \frac{m_{rec}}{M}) \cdot M, {s^l_{rec}}' = 1 - s^l_{rec}, \tag{18}$$

$$s^l_{ind} = (c^l_{ind} - \frac{m_{ind}}{M}) \cdot M, {s^l_{ind}}' = 1 - s^l_{ind}, \tag{19}$$

$$\begin{aligned} h^l_{super} = {} & {s^l_{rec}}' \cdot {s^l_{ind}}' \cdot h^l_{m_{ind}, m_{rec}} + {s^l_{rec}}' \cdot s^l_{ind} \cdot h^l_{m_{ind}+1, m_{rec}} \\ & + s^l_{rec} \cdot {s^l_{ind}}' \cdot h^l_{m_{ind}, m_{rec}+1} + s^l_{rec} \cdot s^l_{ind} \cdot h^l_{m_{ind}+1, m_{rec}+1} \end{aligned} \tag{20}$$

This is like a single path method [17]. Although we need to generate $M \times M$ modules, only a constant number of modules need to be extracted in each calculation, so there is no out-of-memory problem. And we use the differential evolutionary algorithm[47] to find the best candidate $\mathbf{c}$.

According to [17], we divide the search process into two phases, the warmup phase and the search phase. The pseudo-code for this process is shown in appendix in section A.2. Similar to the usual evolutionary algorithms, the differential evolutionary algorithm optimizes the candidates by crossover and mutation. The additional required arguments are listed in the inputs of the pseudo-code with description. These arguments affect the size of the search space and the search speed.

During the warmup phase, for all shared parameters to learn information about the data, we randomly generate a candidate at the beginning of each epoch. At the end of the search process, we obtain the best candidate $\mathbf{c}$ and thus we can get the optimal auxiliary weights according to Eqs. (14) to (17). After that, we retrain or finetune the model and evaluate the performance of the model.

Till now, we complete the disentangled Transformer for recommendation, which makes the representation output by the multi-head attention disentangled with two auxiliary objectives. Additionally, to fit the layer-specific disentanglement characteristics of the Transformer, we propose the adaptive training strategy to automatically find the optimal auxiliary weights for each layer. Our proposed method can be applied to various recommendation models that are based on Transformer and generally improve the performance of the backbone model.

4 EXPERIMENTS

In this section, we empirically evaluate the performance of the proposed Adaptive Disentangled Transformer and analyze how it works. Next, we will describe the backbone models, the evaluation metrics, and the datasets we adopt.

Backbones. To better illustrate the adaptability of our method to different situations, we adopt three different Transformer-based sequential recommendation backbones and evaluate how much improvement these models can achieve through the proposed adaptive disentanglement.

- **SASRec[28]:** This backbone makes full use of the self-attention mechanism and is one of the pioneers in the use of Transformers for sequential recommendation tasks.
- **Bert4Rec[48]:** This backbone adopts the Cloze objective to the sequential recommendation and predicts the masked item by jointly using the left and the right context.
- **STOSA[14]:** This backbone treats the embedding of items as a stochastic Gaussian distribution to fully find the similarity between different items. And they proposed a module using the Wasserstein distance to measure the relationship between items.

The Evaluation Protocols. To evaluate the effectiveness of the method, we chose Hit Ratio (HR), and Normalized Discounted Cumulative Gain (NDCG) as the evaluation metrics, the same as the original papers of the aforementioned backbones. For a fair comparison, we keep the other evaluation details the same as the original paper. However, in the original paper of SASRec and Bert4Rec, they use popularity sampling to sample 100 negative items to calculate these metrics, which will be easily biased as later works indicate[42]. Therefore, on these two backbones, we add the Area Under Curve (AUC) metric which is consistent with different sampling methods. And for STOSA backbone, we add the Mean Reciprocal Rank (MRR) as an evaluation metric to more comprehensively show the superiority of our proposed method.

Datasets. We compare the methods on publicly available datasets from real-world applications.

- **Amazon**[1]**:** The Amazon Dataset records user reviews of Amazon.com products and is a classic dataset for recommendation systems. In our experiments, we adopt the "Beauty ", "Home and Kitchen", "Tools and Home Improvement", "Toys and Games" and "Office" sub-datasets.

[1] http://jmcauley.ucsd.edu/data/amazon/

Table 1: The Overall Performance Comparison Table with SASRec and Bert4Rec backbones. The best and second-best results are bold and underlined. where the "-ADT" suffix means that our method is used.

Dataset	Metric	POP	BPR-MF	NCF	FPMC	GRU4Rec	Caser	SASRec	Bert4Rec	**SASRec-ADT**	**Bert4Rec-ADT**
Beauty	HR@5	0.0392	0.1209	0.1305	0.1387	0.1315	0.1625	0.1934	0.2343	0.2193	**0.2523**
	NDCG@5	0.0230	0.0814	0.0855	0.0902	0.0812	0.1050	0.1436	0.1711	0.1598	**0.1827**
	AUC	0.5201	0.5434	0.5467	0.5534	0.5867	0.6041	0.6634	0.6679	**0.6809**	0.6752
Steam	HR@5	0.0805	0.1177	0.1203	0.1517	0.2171	0.1766	0.2559	0.2282	**0.2785**	0.2458
	NDCG@5	0.0477	0.0744	0.0717	0.0945	0.1370	0.1131	0.1727	0.1509	**0.1917**	0.1619
	AUC	0.5111	0.5311	0.5763	0.5835	0.6133	0.6808	0.7325	0.7204	**0.7408**	0.7321
ML-1m	HR@5	0.0715	0.2866	0.1932	0.4297	0.4673	0.5353	0.5435	0.5902	0.5725	**0.6015**
	NDCG@5	0.0416	0.1903	0.1146	0.2885	0.3196	0.3832	0.3980	0.4515	0.4233	**0.4580**
	AUC	0.5251	0.7411	0.7349	0.7556	0.8311	0.8469	0.8725	0.8805	0.8801	**0.8809**
ML-20M	HR@5	0.0805	0.2128	0.1358	0.3601	0.4657	0.3804	0.5727	0.5439	**0.5841**	0.5599
	NDCG@5	0.0511	0.1332	0.0771	0.2239	0.3090	0.2538	0.4208	0.4018	**0.4311**	0.4150
	AUC	0.5329	0.7213	0.7009	0.7211	0.7780	0.8393	0.8884	0.8863	**0.9048**	0.8916

Table 2: The Overall Performance Comparison Table with the STOSA backbone. The best and second-best results are bold and underlined. where the "-ADT" suffix means that our method is used. And "OOM" means the out of memory error.

Dataset	Metric	LightGCN	TransRec	Caser	SASRec	Bert4Rec	DSSRec	ComiRec	DT4SR	ICLRec	STOSA	**STOSA-ADT**
Home	HR@5	0.0095	0.0063	OOM	0.0127	0.0105	0.0123	0.0092	0.0129	0.0153	0.0133	**0.0184**
	NDCG@5	0.0060	0.0040	OOM	0.0087	0.0067	0.0085	0.0058	0.0082	0.0101	0.0093	**0.0136**
	MRR	0.0071	0.0052	OOM	0.0094	0.0092	0.0086	0.0079	0.0093	0.0102	0.0100	**0.0140**
Beauty	HR@5	0.0300	0.0321	0.0309	0.0416	0.0396	0.0436	0.0351	0.0449	0.0500	0.0504	**0.0533**
	NDCG@5	0.0174	0.0204	0.0214	0.0274	0.0257	0.0308	0.0219	0.0296	0.0326	0.0351	**0.0379**
	MRR	0.0203	0.0236	0.0231	0.0291	0.0294	0.0314	0.0265	0.0323	0.0322	0.0360	**0.0392**
Tools	HR@5	0.0231	0.0210	0.0129	0.0284	0.0189	0.0283	0.0283	0.0289	0.0326	0.0312	**0.0341**
	NDCG@5	0.0152	0.0134	0.0091	0.0194	0.0123	0.0202	0.0204	0.0196	0.0218	0.0217	**0.0241**
	MRR	0.0170	0.0152	0.0106	0.0207	0.0160	0.0211	0.0212	0.0206	0.0230	0.0226	**0.0249**
Toys	HR@5	0.0266	0.0222	0.0240	0.0551	0.0300	0.0565	0.0366	0.0550	0.0598	0.0577	**0.0602**
	NDCG@5	0.0173	0.0143	0.0210	0.0377	0.0206	0.0387	0.0233	0.0360	0.0414	0.0412	**0.0434**
	MRR	0.0200	0.0166	0.0221	0.0385	0.0244	0.0392	0.0272	0.0387	0.0415	0.0415	**0.0438**
Office	HR@5	0.0226	0.0343	0.0302	0.0656	0.0485	0.0599	0.0438	0.0630	0.0653	0.0677	**0.0687**
	NDCG@5	0.0157	0.0219	0.0186	0.0428	0.0309	0.0395	0.0304	0.0421	0.0452	0.0461	**0.0486**
	MRR	0.0181	0.0263	0.0268	0.0457	0.0408	0.0407	0.0376	0.0475	0.0495	0.0502	**0.0521**

- **Steam**[2]**:** This dataset contains a large number of user reviews crawled from the Steam gaming platform.
- **MovieLens**[3]**:** This dataset is one of the most famous datasets for recommendation systems. It contains multiple user reviews for multiple movies. In our experiment, we choose ML-1M and ML-20M which are widely used by most of the recommendation system researches.

In the experiments, in order to be as consistent as possible with the treatment of the original paper of the backbone Transformer, we performed popularity sampling on the experimental data of SASRec backbone and Bert4Rec backbone, while experiments on STOSA backbone do not perform any sampling method.

For the popularity sampling method, we follow the previous work and match each ground truth item in the test set with 100 negative items that do not interact with the user. We use the popularity of the item as the sampling weight and randomly sample these 100 negative samples. For the non-sampling method, we take all items that have not interacted with the current user as negative samples.

[2]https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam
[3]https://grouplens.org/datasets/movielens/

4.1 Main Results

We report the performances of different methods in Table 1 and in Table 2. And we present the basic information of the baselines used in A.1. Note that besides the aforementioned three backbones, we also compare with some other baselines which the original papers compare with, whose details are described in the appendix. According to the results, we have some observations as follows.

The sequential model such as GRU4Rec and Caser is superior to the non-sequential model such as BPR-MF and LightGCN. This indicates that the non-sequential model only takes into account the user behavior information and ignores the temporal information, which does not make full use of the data. In the non-sequential models, we find that LightGCN can achieve the most promising results, which means that the introduction of graph data can capture the interaction behavior of the users. And Transformer-based methods such as SASRec and Bert4Rec not only take advantage of the temporal information but also capture the different intents of the user behavior, thus providing better performances.

For all the backbones, our method is able to bring improvement and outperforms all the other baselines. The relative improvement of NDCG@5 and HR@5 metrics of our method ranges from 1.48%

to 46.24%, which illustrates the superiority of our method. We believe that these improvements come from the following aspects: 1. Compared with the non-Transformer model, thanks to the multi-head attention mechanism, the Transformer is indeed able to extract different intentions of users to purchase different items and thus improve the performance of the model. 2. Compared to the transformer-based model, our method automatically selects the weights of the auxiliary objectives using an adaptive approach that takes full account of the factors including different models, different datasets, and different optimization objectives.

For relatively small datasets such as Beauty and ML-1M in Table 1, Bert4Rec outperforms SASRec in NDCG@5 and HR@5 metrics, but both have similar AUC metrics. On the other hand, for larger datasets, Bert4Rec is not as good as SASRec. It means that the recommendation capability of SASRec is not weaker or even stronger than Bert4Rec, which is also shown in the results in Table 2. This is most likely due to the introduction of the Cloze objective task in Bert4Rec, which leads to inconsistency in model training and testing and thus affects the performance. Nevertheless, our method can consistently bring performance improvement for both of the two backbones. This shows that our method can be applied to different architectures, while the proposed auxiliary objectives can improve the generalization ability of the model, thus alleviating this inconsistency.

We note that the improvement brought by our method is different on different datasets. Taking the experimental results of the STOSA backbone as an example, for the MRR evaluation metric, the relative improvement of our method is 40% on the Home dataset. However, on the Office dataset, it is 3.78%. We believe this is because the Home dataset is sparser and it is more difficult for the recommendation system to capture the commonality of behaviors among users. But our method enables the learned representations to extract this information through auxiliary objectives that allow a deeper understanding of the sequence data. Similar phenomena can be found in our experiments on the SASRec backbone and Bert4Rec backbone. In particular, for Bert4Rec, there is almost no improvement on the ML-1M dataset. This indicates that on a simple dataset like ML-1M, the improvement on the multi-head self-attention mechanism can no longer continue to dig deeper into the information of user behaviors. And to get further improvement, it may be necessary to introduce additional user behavior by other methods, such as explicitly introducing graph data, etc.

4.2 Ablation Studies

We further conduct ablation studies to demonstrate the effectiveness of different components as follows.

- We illustrate that the improvement brought by our proposed ADT framework is not due to the increase in the number of parameters.
- We show the necessity of adaptively finding the optimal degree of disentanglement through exploring the effect of the proposed auxiliary objectives.

Number of model parameters. Since our method adds a decoder, the parameters of the backbone model are increased. To show that the improvement brought by our method is not due to the increase of the parameters, we double the layers of the backbone model to eliminate the effect of the number of parameters. And the results can be found in Table 3.

Table 3: Performance Comparison Table with Double Layers. The "-Double" suffix means that we double the number of layers of that backbone model.

Dataset	Metric	SASRec	SASRec-Double	SASRec-ADT
Beauty	HR@5	0.1934	0.1791	0.2193
	NDCG@5	0.1436	0.1306	0.1598
	AUC	0.6634	0.6071	0.6809
ML-1M	HR@5	0.5435	0.5347	0.5725
	NDCG@5	0.3980	0.3785	0.4233
	AUC	0.8725	0.8608	0.8801

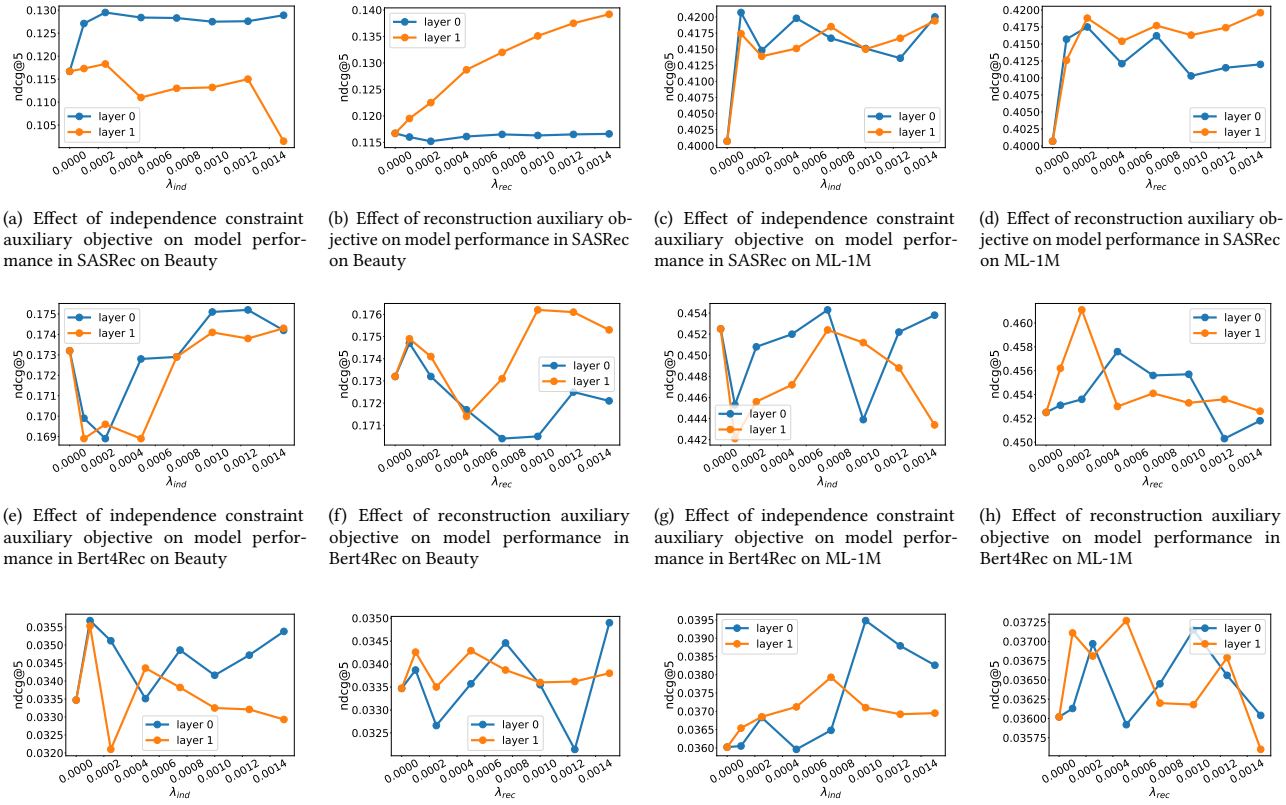
Dataset	Metric	Bert4Rec	Bert4Rec-Double	Bert4Rec-ADT
Beauty	HR@5	0.2343	0.2199	0.2523
	NDCG@5	0.1711	0.1586	0.1827
	AUC	0.6679	0.6382	0.6752
ML-1M	HR@5	0.5902	0.5940	0.6015
	NDCG@5	0.4515	0.4567	0.4580
	AUC	0.8805	0.8775	0.8809

Dataset	Metric	STOSA	STOSA-Double	STOSA-ADT
Home	HR@5	0.0133	0.0168	0.0184
	NDCG@5	0.0093	0.0121	0.0136
	MRR	0.0100	0.0124	0.0140
Beauty	HR@5	0.0504	0.0511	0.0533
	NDCG@5	0.0351	0.0355	0.0379
	MRR	0.0360	0.0368	0.0392
Tools	HR@5	0.0312	0.0331	0.0341
	NDCG@5	0.0217	0.0230	0.0241
	MRR	0.0226	0.0238	0.0249
Toys	HR@5	0.0577	0.2658	0.0602
	NDCG@5	0.0412	0.0174	0.0434
	MRR	0.0415	0.0192	0.0438

From the results, we observe that for most of the datasets, the performance of the model decreases after doubling the number of layers of the backbone model. For the results of the STOSA backbone, we note that the model fails to achieve a good performance after doubling the number of layers on the Toys dataset. One plausible reason is that the Toys dataset is a relatively simple dataset and numerous parameters lead to overfitting on this dataset. But for the Home dataset, the performance is improved because this dataset is more complex. Our method can handle both situations and get better results, by considering the relationship between different models and different datasets adaptively.

Auxiliary Objectives. We conduct experiments on Beauty, ML-1M, and Office datasets to illustrate the effect of our auxiliary objectives. We fixed the structure of the model that obtained optimal results in the previous experiments, i.e., the number of layers, the number of heads, hidden size, etc. For each layer of the model, we give only one of the auxiliary objectives a weight and set the weights of all auxiliary objectives in the other layers to 0. The results are shown in Figure 2.

We find that for different datasets, different model architectures, and different layers, the performance of the model changes with



(a) Effect of independence constraint auxiliary objective on model performance in SASRec on Beauty

(b) Effect of reconstruction auxiliary objective on model performance in SASRec on Beauty

(c) Effect of independence constraint auxiliary objective on model performance in SASRec on ML-1M

(d) Effect of reconstruction auxiliary objective on model performance in SASRec on ML-1M

(e) Effect of independence constraint auxiliary objective on model performance in Bert4Rec on Beauty

(f) Effect of reconstruction auxiliary objective on model performance in Bert4Rec on Beauty

(g) Effect of independence constraint auxiliary objective on model performance in Bert4Rec on ML-1M

(h) Effect of reconstruction auxiliary objective on model performance in Bert4Rec on ML-1M

(i) Effect of independence constraint auxiliary objective on model performance in STOSA on Beauty

(j) Effect of reconstruction auxiliary objective on model performance in STOSA on Beauty

(k) Effect of independence constraint auxiliary objective on model performance in STOSA on Office

(l) Effect of reconstruction auxiliary objective on model performance in STOSA on Office

Figure 2: Effect of auxiliary objectives on different layers of the model

these two auxiliary weights. And this change is beneficial in most cases when the values of these auxiliary weights are in a suitable range. Besides, the value where the best performance appears varies from different models and different datasets. This means that these two auxiliary objectives we propose really make sense and that how to adaptively find the optimal weights for different layers is significant for the backbone Transformer model to get more positive information. And we also note that for the independence constraint auxiliary objective, the best model performance always occurs at the shallow layer, and overall, it is better to apply the objective to the shallow layer than to the deep layer. This is consistent with previous works [10, 35, 45] where shallow layers need more disentangled representations. And for the reconstruction auxiliary objective, the best model performance always occurs at the deep layer except in Figure 2(j), where the reconstruction can be regarded as an auxiliary task to improve the generalization ability of the whole model instead of only the shallow part.

To conclude, the experiments verify that our ADT framework is able to encourage the disentanglement of attention heads to capture various aspects of user interests, and adaptively discover the optimal degree of disentanglement for attention heads in different layers to achieve the best recommendation performance.

5 CONCLUSION

In this paper, we propose a novel Adaptive Disentangled Transformer (ADT) framework for the sequential recommendation, which is able to disentangle representations learned by different heads within different layers as well as adaptively determine the optimal degree of disentanglement for recommendation simultaneously. Extensive experiments show that our proposed framework can significantly improve the performance of various transformer-based recommendation models. We believe that it deserves further investigations to explore the adaptive disentangled Transformer in a more general setting, i.e., it will definitely be interesting to design an adaptive Transformer capable of handling more tasks including machine translation, image classification, object detection, etc.

ACKNOWLEDGMENTS

This work was supported in part by the National Key Research and Development Program of China No. 2020AAA0106300, National Natural Science Foundation of China (No. 62250008, 62222209, 62102222, 61936011), Beijing National Research Center for Information Science and Technology under Grant No. BNR2023RC01003, BNR2023TD03006, and Beijing Key Lab of Networked Multimedia.

REFERENCES

[1] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. 2017. Designing Neural Network Architectures using Reinforcement Learning. In *International Conference on Learning Representations*.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35, 8 (2013), 1798–1828.

[3] Yuchen Bian, Jiaji Huang, Xingyu Cai, Jiahong Yuan, and Kenneth Church. 2021. On attention redundancy: A comprehensive study. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 930–945.

[4] Andrew Brock, Theo Lim, JM Ritchie, and Nick Weston. 2018. SMASH: One-Shot Model Architecture Search through HyperNetworks. In *International Conference on Learning Representations*.

[5] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable multi-interest framework for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2942–2951.

[6] Hong Chen, Xin Wang, Yue Liu, Yuwei Zhou, Chaoyu Guan, and Wenwu Zhu. 2022. Module-Aware Optimization for Auxiliary Learning. *Advances in Neural Information Processing Systems*.

[7] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1294–1303.

[8] Yongjun Chen, Zhiwei Liu, Jia Li, Julian McAuley, and Caiming Xiong. 2022. Intent contrastive learning for sequential recommendation. In *Proceedings of the ACM Web Conference 2022*. 2172–2182.

[9] Kyunghyun Cho, Bart Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.

[10] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 276–286.

[11] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential user-based recurrent neural network recommendations. In *Proceedings of the eleventh ACM conference on recommender systems*. 152–160.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

[13] Ziwei Fan, Zhiwei Liu, Shen Wang, Lei Zheng, and Philip S Yu. 2021. Modeling sequences as distributions with uncertainty for sequential recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3019–3023.

[14] Ziwei Fan, Zhiwei Liu, Yu Wang, Alice Wang, Zahra Nazari, Lei Zheng, Hao Peng, and Philip S Yu. 2022. Sequential recommendation via stochastic self-attention. In *Proceedings of the ACM Web Conference 2022*. 2036–2047.

[15] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2019. Graphnas: Graph neural architecture search with reinforcement learning. *arXiv preprint arXiv:1904.09981* (2019).

[16] Chaoyu Guan, Xin Wang, and Wenwu Zhu. 2021. Autoattend: Automated attention representation search. In *International conference on machine learning*. PMLR, 3864–3874.

[17] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. 2020. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*. Springer, 544–560.

[18] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*. 161–169.

[19] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 191–200.

[20] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[21] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[22] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 843–852.

[23] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[24] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*.

[25] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[26] Linmei Hu, Siyong Xu, Chen Li, Cheng Yang, Chuan Shi, Nan Duan, Xing Xie, and Ming Zhou. 2020. Graph neural news recommendation with unsupervised preference disentanglement. In *Proceedings of the 58th annual meeting of the association for computational linguistics*. 4255–4264.

[27] Sarthak Jain and Byron C Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 3543–3556.

[28] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[29] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[30] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[31] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*.

[32] Shikun Liu, Andrew Davison, and Edward Johns. 2019. Self-supervised generalisation with meta auxiliary learning. *Advances in Neural Information Processing Systems* 32 (2019).

[33] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 825–833.

[34] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled self-supervision in sequential recommenders. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 483–491.

[35] Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems* 32 (2019).

[36] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In *International conference on machine learning*. PMLR, 4055–4064.

[37] Bo Peng, Zhiyun Ren, Srinivasan Parthasarathy, and Xia Ning. 2021. HAM: Hybrid Associations Models for Sequential Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 10 (2021), 4838–4853.

[38] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*. PMLR, 4095–4104.

[39] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *proceedings of the Eleventh ACM Conference on Recommender Systems*. 130–137.

[40] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *International Conference on Machine Learning*. PMLR, 8821–8831.

[41] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, Vol. 33. 4780–4789.

[42] Steffen Rendle. 2019. Evaluation metrics for item recommendation under sampling. *arXiv preprint arXiv:1912.02263* (2019).

[43] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461.

[44] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[45] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics* 8 (2021), 842–866.

[46] Michael S Ryoo, AJ Piergiovanni, Mingxing Tan, and Anelia Angelova. 2019. AssembleNet: Searching for Multi-Stream Neural Connectivity in Video Architectures. In *International Conference on Learning Representations*.

[47] RAINER STORN and KENNETH PRICE. 1997. Differential Evolution–A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization* 11 (1997), 341–359.

[48] 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 *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[49] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 565–573.
[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[51] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 5797–5808.
[52] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. 2019. Learning Deep Transformer Models for Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 1810–1822.
[53] Xin Wang, Hong Chen, Si'ao Tang, Zihao Wu, and Wenwu Zhu. 2022. Disentangled Representation Learning. arXiv:2211.11695 [cs.LG]
[54] Xin Wang, Hong Chen, Yuwei Zhou, Jianxin Ma, and Wenwu Zhu. 2022. Disentangled representation learning for recommendation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 1 (2022), 408–424.
[55] Xin Wang, Hong Chen, and Wenwu Zhu. 2021. Multimodal disentangled representation for recommendation. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1–6.
[56] Xin Wang, Shuyi Fan, Kun Kuang, and Wenwu Zhu. 2021. Explainable automated graph representation learning with hyperparameter importance. In *International Conference on Machine Learning*. PMLR, 10727–10737.
[57] Xin Wang, Yue Liu, Jiapei Fan, Weigao Wen, Hui Xue, and Wenwu Zhu. 2023. Continual Few-shot Learning with Transformer Adaptation and Knowledge Regularization. In *Proceedings of the ACM Web Conference 2023*. 1519–1527.
[58] Xin Wang, Zirui Pan, Yuwei Zhou, Hong Chen, Chendi Ge, and Wenwu Zhu. 2023. Curriculum Co-disentangled Representation Learning across Multiple Environments for Social Recommendation. In *International conference on machine learning*. PMLR.
[59] Cheng Yang, Maosong Sun, Xiaoyuan Yi, and Wenhao Li. 2018. Stylistic chinese poetry generation via unsupervised style disentanglement. In *Proceedings of the 2018 conference on empirical methods in natural language processing*. 3960–3969.
[60] Biao Zhang, Ivan Titov, and Rico Sennrich. 2019. Improving Deep Transformer with Depth-Scaled Initialization and Merged Attention. In *2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics (ACL), 898–909.
[61] Yin Zhang, Ziwei Zhu, Yun He, and James Caverlee. 2020. Content-collaborative disentanglement representation learning for enhanced recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 43–52.
[62] Zizhao Zhang, Xin Wang, Chaoyu Guan, Ziwei Zhang, Haoyang Li, and Wenwu Zhu. 2023. AutoGT: Automated Graph Transformer Architecture Search. In *The Eleventh International Conference on Learning Representations*.
[63] Barret Zoph and Quoc Le. 2017. Neural Architecture Search with Reinforcement Learning. (2017).
[64] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8697–8710.

A SUPPLEMENTARY MATERIAL

A.1 Comparison Methods

In this section, we give a brief description of the baselines for reference.

- **POP** is a simple method which ranks items according to their popularity.
- **BPR-MF** [43] learns personalized rankings from implicit feedback and uses matrix factorization as recommender.
- **NCF** [21] uses neural architecture instead of the inner product to model interactions between users and items.
- **FPMC** [44] combines matrix factorization and Markov chains to capture user preferences.
- **GRU4Rec** [23] uses Gated Recurrent Units to model the sequence information of user's behaviors.
- **Caser** [49] models high order Markov chains by introducing convolution operations.
- **LightGCN** [20] simplifies the architecture of GCN to better handle the recommendation tasks.
- **TransRec** [18] embeds items into the latent transition space.
- **DSSRec** [34] combines disentangled representation learning and self-supervised learning to balance the weights of multiple interests.
- **ComiRec** [5] introduces attention mechanism and user interests to make recommendations.
- **DT4SR** [13] uses distributions to represent items and sequences and develops two Transformers for modeling mean and covariance embeddings.
- **ICLRec** [8] uses contrastive learning to model different purchase interests.

A.2 Searching Algorithm

Algorithm 1 Searching Process of ADT

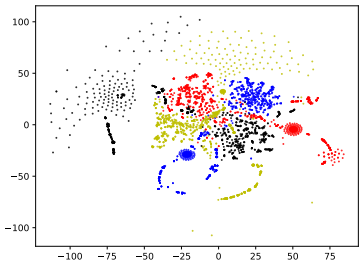
Input: The differential weight F, the crossover probability P_c, the mutation probability P_m, the number of population n_p, the number of crossover per iteration n_c, the number of mutation per iteration n_m, and the sequential dataset $\boldsymbol{D}$

Output: The best candidate c on the given dataset.

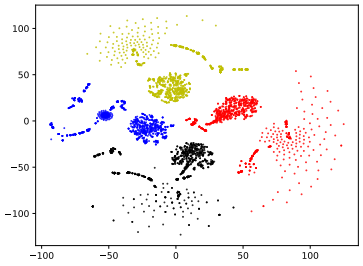
Warmup the supernet according to the forward form of Eqs. (18) to (20).
Randomly initialize the population $\mathbf{C} = \{\mathbf{c}_1.\mathbf{c}_2, ..., \mathbf{c}_n\}$ where each $\mathbf{c}_i$ is a d-dimensional vector representing a candidate sub-networks and initialize the set $\mathbf{G} = \mathbf{C}$ representing the optimal n candidates.
while Not reach the maximum number of search steps and not exceed the time threshold **do**
  clear $\mathbf{C}$
  for i = 1 to n_m **do**
    Randomly choose 3 candidates $\mathbf{c}_r, \mathbf{c}_p, \mathbf{c}_q$ from $\mathbf{G}$
    $\mathbf{v}_i = \mathbf{c}_r + F(\mathbf{c}_p - \mathbf{c}_q)$
    for j = 1 to d **do**
      Generate a random number P
      if $P \leq P_m$ **then**
        $\mathbf{u}_{i,j} = \mathbf{v}_{i,j}$
      end if
    end for
    Appending $\mathbf{u}_i$ into $\mathbf{C}$
  end for
  for i = 1 to n_c **do**
    Randomly choose 2 candidates $\mathbf{c}_p, \mathbf{c}_q$ from $\mathbf{G}$
    for j = 1 to d **do**
      Generate a random number P
      if $P \leq P_c$ **then**
        $\mathbf{u}_{i,j} = \mathbf{c}_{p,j}$
      else
        $\mathbf{u}_{i,j} = \mathbf{c}_{q,j}$
      end if
    end for
    Appending $\mathbf{u}_i$ into $\mathbf{C}$
  end for
  Randomly generate candidate vectors until $|\mathbf{C}| \geq n_p$
  Evaluate the performance on the validation set of $\mathbf{D}$ of each candidate in the recommendation task and update $\mathbf{G}$ based on the metric.
end while
return The candidate $\mathbf{c}$ with the best performance that have appeared in the search history.

A.3 Latent Space Analysis

We visualize the representations learned by different heads of SASRec and SASRec-ADT. We can find that the original backbone fails to disentangle the representations of different heads as expected but our method disentangles the different parts and forms distinct clusters which represent different aspects of user behaviors.



(a) t-SNE of the representation of different heads learned by SASRec



(b) TSNE of the representation of different heads learned by SASRec-ADT

Figure 3: t-SNE of the representations on ML-1M dataset

A.4 Implementation Details

For SASRec backbone and Bert4Rec backbone, we trained the model on a device with 8 NVIDIA GeForce GTX TITAN X. And for STOSA backbone, we trained the model on a device with 2 NVIDIA GeForce RTX 3090.

In the search process, for all backbones, we use the Adam optimizer with learning rate of $1e-3$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, ℓ_2 weight decay of $1e-4$. And the gradient is clipped when its norm exceeds 5. We set the number of population to 50, the number of candidates obtained from both crossover and mutation in each iteration to 20, the differential weight F to 0.5 , and the probability of both crossover and mutation to 0.1. During the retraining process, we choose different hyperparameters depending on the backbone.

- **SASRec:** We set the layer number $L = 2$, and consider the head number h from $\{1, 2, 4\}$, the hidden size d from $\{64, 128, 256\}$, the learning rate from $\{0.0001, 0.001\}$, the ℓ_2 weight decay from $\{1e-5, 1e-4, 1e-3\}$. We train the model using Adam and a batch size of 256. We use the same maximum sequence length in [28], $|S_u| = 200$ for MovieLens datasets and $|S_u| = 50$ for others.
- **Bert4Rec:** We set the layer number $L = 2$, the head number from $h = \{1, 2, 4\}$ and use the same maximum sequence length as in SASRec backbone. We follow the original paper of Bert4Rec [48] and set the hidden size of each head to 32. For the mask proportion ρ, we set $\rho = 0.6$ for Beauty, $\rho = 0.4$ for Steam, and $\rho = 0.2$ for MovieLens datasets as recommended. We train the model using Adam with learning rate of $1e-4$ for Steam and ML-20M, $1e-3$ for Beauty and ML-1M and ℓ_2 weight decay from $\{0, 1e-4, 1e-3\}$ for all of them.
- **STOSA:** We grid search the hyperparameter similar to the original paper of STOSA [14]. We consider the hidden size d from $\{32, 64\}$, the number of layers from $\{1, 2, 3\}$, the number of heads from $\{1, 2, 4\}$, the learning rate from $\{1e-3, 1e-4\}$, the ℓ_2 weight decay from $\{0, 1e-4, 1e-3\}$ and the dropout rate from $\{0.0, 0.1, 0.3, 0.5, 0.7\}$. For the maximum sequence length, we set it to 100 for all datasets.

A.5 Statistics of Datasets

The details of datasets statistics used in SASRec and Bert4Rec are presented in Table 4 and the statistics used in STOSA are presented in Table 5[4].

Table 4: Datasets Statistics for SASRec and Bert4Rec backbone

Dataset	#user	#items	#interactions	density	avg. interactions per user
Beauty	40226	54542	353962	0.02%	8.7993
Steam	334730	13047	3686172	0.08%	11.0124
ML-1M	6040	3416	999611	4.84%	165.4985
ML-20M	138493	26744	20000263	0.54%	144.4135

Table 5: Datasets Statistics for STOSA backbone

Dataset	#user	#items	#interactions	density	avg. interactions per user
Home	66519	28238	551582	0.03%	8.2936
Beauty	22363	12102	198502	0.07%	8.8764
Toys	19412	11925	167597	0.07%	8.6337
Tools	16638	10218	134476	0.08%	8.0825
Office	4905	2421	53258	0.45%	10.8579

[4]The statistics of Beauty dataset here is different from in Table 4 is because STOSA backbone uses 5-core settings.