Automated Disentangled Sequential Recommendation with Large Language Models

XIN WANG, Department of Computer Science and Technology, BNRist, Tsinghua University, Beijing, China
HONG CHEN, ZIRUI PAN, YUWEI ZHOU, CHAOYU GUAN, and LIFENG SUN,
Department of Computer Science and Technology, Tsinghua University, Beijing, China
WENWU ZHU, Department of Computer Science and Technology, BNRist, Tsinghua University, Beijing, China

Sequential recommendation aims to recommend the next items that a target user may have interest in based on the user's sequence of past behaviors, which has become a hot research topic in both academia and industry. In the literature, sequential recommendation adopts a Sequence-to-Item or Sequence-to-Sequence training strategy, which supervises a sequential model with a user's next one or more behaviors as the labels and the sequence of the past behaviors as the input. However, existing powerful sequential recommendation approaches employ more and more complex deep structures such as Transformer in order to accurately capture the sequential patterns, which heavily rely on hand-crafted designs on key attention mechanism to achieve state-of-the-art performance, thus failing to automatically obtain the optimal design of attention representation architectures in various scenarios with different data. Other works on classic automated deep recommender systems only focus on traditional settings, ignoring the problem of sequential scenarios. In this article, we study the problem of automated sequential recommendation, which faces two main challenges: (1) How can we design a proper search space tailored for attention automation in sequential recommendation, and (2) How can we accurately search effective attention representation architectures considering multiple user interests reflected in the sequential behavior. To tackle these challenges, we propose an automated disentangled sequential recommendation (AutoDisenSeq) model. In particular, we employ neural architecture search (NAS) and design a search space tailored for automated attention representation in attentive intention-disentangled sequential recommendation with an expressive and efficient space complexity of $O(n^2)$ given n as the number of layers. We further propose a context-aware parameter sharing mechanism taking characteristics of each sub-architecture into account to enable accurate architecture performance estimations and great flexibility for disentanglement of latent intention representation. Moreover, we propose AutoDisenSeq-large language

This work was supported in part by the National Key Research and Development Program of China (Grant No. 2022ZD0115903), in part by the National Natural Science Foundation of China (Grant Nos. 62222209, 62250008, and 62102222), in part by the Beijing National Research Center for Information Science and Technology (Grant Nos. BNR2023RC01003 and BNR2023TD03006), and in part by the Beijing Key Lab of Networked Multimedia. All opinions, findings, and conclusions in this article are those of the authors and do not necessarily reflect the views of the funding agencies.
Authors' Contact Information: Xin Wang, Department of Computer Science and Technology, BNRist, Tsinghua University, Beijing, China; e-mail: xin_wang@tsinghua.edu.cn; Hong Chen, Department of Computer Science and Technology, Tsinghua University, Beijing, China; e-mail: h-chen20@mails.tsinghua.edu.cn; Zirui Pan, Department of Computer Science and Technology, Tsinghua University, Beijing, China; e-mail: panzr20@mails.tsinghua.edu.cn; Yuwei Zhou, Department of Computer Science and Technology, Tsinghua University, Beijing, China; e-mail: zhou-yw21@mails.tsinghua.edu.cn; Chaoyu Guan, Department of Computer Science and Technology, Tsinghua University, Beijing, China; e-mail: guancy19@mails.tsinghua.edu.cn; Lifeng Sun, Department of Computer Science and Technology, Tsinghua University, Beijing, China; e-mail: sunlf@tsinghua.edu.cn; Wenwu Zhu (corresponding author), Department of Computer Science and Technology, BNRist, Tsinghua University, Beijing, China; e-mail: wwzhu@mail.tsinghua.edu.cn.

ACM 1558-2868/2025/1-ART29
https://doi.org/10.1145/3675164

model (LLM), which utilizes the textual understanding power of LLM as a guidance to refine the candidate list for recommendation from AutoDisenSeq. We conduct extensive experiments to show that our proposed AutoDisenSeq model and AutoDisenSeq-LLM model outperform existing baseline methods on four real-world datasets in both overall recommendation and cold-start recommendation scenarios.

CCS Concepts: • **Computing methodologies → Artificial intelligence**; **Machine learning**; • **Information systems → Collaborative filtering**;

Additional Key Words and Phrases: Recommender systems, sequential modeling, disentangled representation learning, automated machine learning

ACM Reference format:
Xin Wang, Hong Chen, Zirui Pan, Yuwei Zhou, Chaoyu Guan, Lifeng Sun, and Wenwu Zhu. 2025. Automated Disentangled Sequential Recommendation with Large Language Models. *ACM Trans. Inf. Syst.* 43, 2, Article 29 (January 2025), 29 pages.
https://doi.org/10.1145/3675164

1 Introduction

Sequential recommendation aims to recommend the next (sequence of) items to a target user based on the sequence of his/her historical behaviors in web services and mobile applications as input, such as rate, watch, comment, favorite, click, and so on. Motivated by the power of deep learning in modeling sequential data, recent research works [51, 59, 117, 136] have gained huge success on sequential recommendation with deep sequential models such as self-attention networks (aka. Transformers) [28, 123]. Nevertheless, these works heavily rely on manual design to obtain self-attention representations for sequential patterns capture, where various functional structures, including **convolutional neural networks (CNN)**, **recurrent neural networks (RNN)**, Graph Attention Network, and Graph Convolutional Network are manually stacked or combined to calculate *Key*, *Query*, and *Value* from the input data [28, 79, 123, 124]. These hand-crafted designs of attention representations cost numerous trial-and-error human labors and tend to be sub-optimal due to human bias, thus making it difficult to fit in real-world sequential recommendation scenarios.

To solve this problem, we propose for the first time to study automated sequential recommendation through **neural architecture search (NAS)** on the attention mechanism. However, investigating automated sequential recommendation poses the following challenges:

— *How to design a proper search space tailored for attention automation in sequential recommender systems?* Recent sequential models utilize various self-attention blocks to capture the sequential patterns of interests hidden in user behaviors. A proper attention search space may significantly increase the possibility of obtaining the optimal self-attention representations for downstream sequential recommendation.
— *How to accurately search effective attention representations considering multiple user interests reflected in the sequential behavior?* Users tend to have diverse and multiple intentions in real-world scenarios. It is crucial for the search strategy to discover the most appropriate attention representations so that the latent deep representations can cover diverse and multiple user intentions.

To tackle these challenges, we propose **automated disentangled sequential recommendation (AutoDisenSeq)** model, which is able to automatically discover powerful attention representations for sequential recommendation while keeping intention disentangled in latent space. Our proposed self-attention automation can further obtain the adaptive extractions of multiple user interests in different scenarios through intention disentanglement, where a given sequence of behaviors is encoded into multiple representations, each of which characterizes a unique user intention.

Particularly, we design a search space tailored for automated attention representation in attentive intention-disentangled sequential recommendation with an expressive and efficient space complexity of $O(n^2)$ given n as the number of layers. We further propose a context-aware parameter sharing mechanism taking characteristics of each sub-architecture into consideration to obtain accurate architecture performance estimations and great flexibility for disentanglement of latent intention representations. Moreover, we discover that the ranking list of candidate items returned from AutoDisenSeq may not always have the optimal order. Therefore, we propose a **large language model (LLM)** variant, AutoDisenSeq-LLM, through employing the textual understanding power of LLM as a guidance to reorder the candidate list for recommendation from AutoDisenSeq. As such, the strength of LLM in understanding item characteristics via textual descriptions such as title of a movie, content and price of a video game and so on can help to boost recommendation performance. We conduct extensive experiments to show that both of our proposed AutoDisenSeq and AutoDisenSeq-LLM models are able to outperform existing baseline methods on several real-world datasets in overall and cold-start recommendation scenarios, validating the advantages of automating the attention representation and utilizing LLM in sequential recommendation.

To summarize, our main contributions are listed as follows:

— We propose an AutoDisenSeq model, which is able to discover the optimal self-attention representations capable of capturing sequential patterns in sequential recommendation through NAS.
— We propose a tailored search space supporting the joint search for attention representations and other functional blocks to discover optimal deep architectures for sequential modeling with a space complexity of $O(n^2)$ given n as the number of layers.
— We propose a context-aware parameter sharing mechanism for automated attention representation in sequential recommendation, which takes special characteristics of each architecture into account to provide reliable architecture evaluations for parameter sharing in our search space, providing great flexibility to intention disentanglement in latent space.
— We propose AutoDisenSeq-LLM, an LLM guided variant that further refines the ranking of candidate items via utilizing the textual understanding power of LLM for better recommendation.
— Extensive experiments empirically demonstrate the superiority of our proposed AutoDisenSeq and AutoDisenSeq-LLM against state-of-the-art baseline approaches in both overall recommendation and cold-start recommendation scenarios.

The remainder of this article is organized as follows. We review related works in Section 2, followed by our detailed discussions on our proposed AutoDisenSeq and AutoDisenSeq-LLM in Section 4. We then describe our experimental settings as well as empirical results in Section 5. At last, we summarize the whole article and point out future works deserving further investigation in Section 6.

2 Related Work

In this section, we review related works on sequential recommendation, disentangled representation, self-attention representation design, NAS, and LLM for recommendation.

2.1 Sequential Recommendation

Traditional **recommendation systems (RS)** typically employ collaborative filtering [27, 56, 107, 109, 113], particularly the matrix factorization-based [66, 112] approaches for mining user preferences hidden in their historical behaviors. Furthermore, the emergence of deep learning has significantly propelled research in this area [22, 50, 76, 77, 125, 151]. However, these pioneering works often overlook the sequential patterns inherent in user interactions within RS. To address this gap, some researchers have developed methods to model user sequential behaviors using first-order or higher-order Markov chains [46, 47, 49, 108, 119, 126]. Motivated by the expressive power of sequential models with deep architectures, recent sequential recommendations [11, 16, 51, 52, 57, 59, 71, 74, 84, 85, 117, 119, 130, 137, 143, 148, 151] make greater achievements by resorting to advanced deep models including RNN, CNN, and the state-of-the-art self-attention models, Transformer [28, 91, 123]. Recent transformer-based sequential models including SASRec [60], which stacks self-attention blocks to model user intentions and **Bidirectional Encoder Representations from Transformers (BERT)** 4Rec [117], which adopts the Cloze objective and utilizes the left and right context to predict the masked item. However, existing transformer-based sequential recommendation models usually rely on hand-crafted designs of attention representations, which are fixed and sub-optimal to various recommendation scenarios.

2.2 Disentangled Representation Learning

Disentangled representations capture distinct explanatory factors inherent in observations across different dimensions of the vectorized representation [3, 67, 127]. There have been various approaches proposed to enforce disentanglement in learned representations. For example, some methods reframe variational autoencoders [63] from an information-theoretic perspective and derive various regularization terms that minimize the mutual information among different parts of the representations [5, 14, 53, 61]. Others investigate disentangling factors behind an observation from the perspective of mixture data samples [4, 15, 29, 31, 58]. The concept of disentanglement finds application across diverse data types, including images [32, 54, 65, 68, 92, 154], texts [48], multimedia [12, 13, 132], and graphs [44, 72, 73, 89, 146, 149]. Recently, several recommendation algorithms based on representation learning [10, 70, 83, 90, 91, 128, 129, 131] have been proposed to disentangle and preserve the multiple intentions of target users via dynamic routing, prototype-based clustering, overlapped community detection, and so on. Our work differs from the literature in that we adopt an automated attention architecture to disentangle the user intentions, which are dynamic to different recommendation domains.

2.3 Self-Attention Representation

The attention mechanism was initially introduced to enable models to selectively focus on relevant segments of source sentences during translation of subsequent words [1, 87]. Subsequently, attention evolved beyond textual alignment in translations and found widespread application across various research domains [18, 26, 97, 116, 123, 140, 142] with manually designed information flows of Key, Query, and Value representations.

Designing powerful attention involves two primary steps: (1) derivation of Key, Query, and Value representations, and (2) attention calculation. The latter step has already received a lot of research interests [24, 64, 114]. Bahdanau et al. and Luong et al. [1, 87] are among the first to propose the concept of attention. Their encoder-decoder attention mechanisms help previous Long Short-Term Memory to align the target sentence and source sentence during translation. However, these early attention mechanisms limited decoder attention to the last encoder layer, neglecting information from other layers. The encoder–decoder attention is then improved and applied to many other research areas [20, 30, 69, 86, 96, 98, 145]. Lee et al. [69] and Lu et al. [86] extended attention to multi-modal fusion with cross-modal co-attention. Nguyen and Okatani [98] densely stacked co-attention layers to model encoder–decoder relations. Yu et al. [145] further explore various representation design approaches. Despite the strengths of attention mechanisms, crafting attention representations requires careful design and domain expertise, often leading to sub-optimal results due to inherent human biases.

Unlike the second step mentioned above, the first step, i.e., designing Key, Query, and Value of attention, has not been largely explored for a long time, although it is as important as the second.

The milestone is self-attention proposed by Vaswani et al. [123]. It helps achieve improvements in many research fields [24, 124]. It abandons the traditional recurrent structure and only relies on self-attention to extract relational features from data. It constrains the source layer of Key, Query, and Value to be the same and just merely uses separate linear transformations to deal with the difference. However, this constraint design increases the optimization difficulty because one layer needs to act as Key, Query, and Value at the same time [25]. At the same time, the interaction of information among different layers is also ignored. The power of linear transformation is limited to discriminating Query and Key, leading to sub-optimal results.

In our work, we focus on automating self-attention representation for sequential recommendation. Unlike prior approaches reliant on human expertise, we propose an automatic framework utilizing NAS to design attention representations, capturing sequential patterns and exploiting semantic meanings stored in different layers.

We remark that our automation of attention representation is orthogonal to other research topics in attention design area, such as computation efficiency [19, 64, 120] and power exploitation [24, 94, 106], which can simply be employed to make our framework more efficient and powerful.

2.4 NAS

NAS aims to design optimal architectures tailored to specific tasks and has gained increasing popularity in recent years [81, 95, 101, 155]. A variety of search algorithms including reinforcement learning [101, 155], **evolutionary algorithm (EA)** [42, 115, 156], gradient-based algorithm [81, 95], and Bayesian Optimization have been developed to tackle the NAS problem across various domains such as Computer Vision [80, 95, 153], **Natural Language Processing (NLP)** [115, 134], Graph Representation Learning [7, 8, 37, 39, 102, 105, 138, 141, 150, 152], and so on. Moreover, the search efficiency can be significantly boosted by introducing parameter-sharing [101], one-shot formulation with supernet [42], and soft-relaxation of search space [81]. However, these techniques speed up the training process at the cost of eliminating the difference in operations [21], so it becomes difficult to model the relations among different operation choices, especially when the operation choices are highly relevant with their *contexts*. In this study, we propose automating attention representation design through NAS with context-aware parameter sharing to enhance performance in sequential recommendation tasks.

Our work is also related to the search space design of NAS, which receives increasing research focus with the success of NAS. Zoph and Le [155] propose the first search space for CNN and RNN, which is optimized for the full model architecture. Later, the cell-based search space was proposed [156] and further advanced by many other works [6, 9, 35, 81, 95, 101, 139] for its higher efficiency and generality. Though the search space design over other architectures is also popular [37, 80, 111, 115, 152], few of them consider attention when designing search space.

Recent works [37, 115, 134, 144, 152] utilize self-attention or encoder–decoder attention in their space design to search for Transformer or Graph Neural Network-styled architectures, which merely consider attention as a candidate primitive operation. The input design for attention is still fixed to certain layers, leading to sub-optimal solutions. There are also some works [88, 133] focusing on the calculation of attention given Key, Query, and Value, which can be seen as automating attention computation (the second step for designing attention as discussed in Section 2.3), thus being orthogonal to our automation of attention representation in this article.

2.5 LLM for Recommendation

LLM for recommendation refers to the application of LLM on RS. Given the demonstrated efficacy of LLMs in NLP, researchers are exploring avenues to extend their utility. One of the promising domains that LLM can be adapted to is recommendation. RS, reliant on deep learning methods

[22, 50, 76, 77, 125, 151], face challenges in generalization, explanation, and reasoning despite their ability to model user–item interactions. Consequently, recent efforts have focused on integrating LLMs into recommendation, categorized into three branches [33]: pretraining-based, finetuning-based, and prompting-based approaches. Pretraining-based methods emulate common NLP tasks such as masked language modeling and next token prediction to pretrain recommendation models. For instance, PTUM [135] employs Masked Behavior Prediction and Next K Behavior Prediction tasks, M6 [23] utilizes text-infilling and auto-regressive language generation objectives, and P5 [38] employs multi-mask modeling and amalgamates datasets from diverse recommendation tasks. Finetuning-based methods seek to transfer LLM knowledge to the recommendation domain. For example, RecLLM [36] fully finetunes LaMDA [121] as a conversational recommender system for YouTube video recommendations, while TallRec [2] refines the LLaMA-7B model [122] in a parameter-efficient manner based on LoRA [55]. Prompting-based methods leverage the advantages of prompting to align LLMs with the recommendation domain. For example, Liu et al. [82] teach ChatGPT how to act as recommender systems by in-context learning, Guo et al. [41] encode mutual information for cross-domain recommendations into soft prompts for prompt tuning, and Zhang et al. [147] design recommendation-oriented templates to generate instructions for recommendation tasks through instruction tuning. In this work, considering the generalization ability of LLM to various domains, we utilize ChatGPT to collaborate with our AutoDisenSeq model, where the pretrained knowledge of LLM and specific knowledge of the AutoDisenSeq are combined to improve the recommendation performance, especially for the cold-start recommendation scenario.

3 Notations and Preliminary

Sequential recommendation require a sequence of user–item interactions as data for model learning. In this work, the training data is denoted as $\{s^{(n)}\}_{n=1}^{N}$, e.g., the sequence of user clicks on items. Here N is the number of users and $s^{(n)} = [s_1^n, s_2^n, s_3^n, ..., s_m^n]$ is the ordered sequence of items clicked by user n, where m is the number of user clicks, and $s_m^n \in \{1, 2, \cdots, T\}$ means that the latest item clicks made by user n. We focus on the candidate generation phase of modern recommender systems, whose task is to predict the next item that a user is likely to click on among all possible options based on the observed sequence. For this task, we need to build a sequence encoder $\varphi_\theta(\cdot)$ and an item embedding table $\boldsymbol{H} \in \mathbb{R}^{T\times D}$. The encoder takes a sequence $s^{(n)}$ representing user's recent clicks as input and outputs the representation of the sequence $\varphi_\theta(s^{(n)}) \in \mathbb{R}^{K\times D}$. In this article, the representation consisting of K D-dimensional vectors is expected to be disentangled, preserving user's intention across K different latent categories. Through utilizing disentangled representation, the model estimates the probability that user n will click item i by measuring the similarity between $\varphi_\theta(s^{(n)})$ and $\boldsymbol{H}_i$ in the vector space.

For most of the current RS, it is not only necessary to capture users' long-term preferences across different sessions as the conventional recommendation does, but also extremely important to simultaneously model users' short-term interest in a session (or a short sequence). To satisfy the above requirements, attention mechanism becomes the mainstream solution. Essentially, the idea behind is that the sequential output depends on some "relevant" part of the input that the model should focus on consecutively. Another benefit is that attention-based approaches are relatively easy to interpret. The widely adopted attention is defined as

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

where Q represents the queries, K represents the keys, and V represents the values (each row represents an item). Intuitively, the attention layer calculates a weighted sum of all values, where the weight between query i and value j relates to the interaction between them. The scale factor $\sqrt{d}$ is to avoid overly large values of the inner product, especially in the case of high dimensionality.

For the self-attention method, it uses the same objects as queries, keys, and values. The self-attention operation takes the embedding E as input, converts it to three matrices through linear projections, and feeds them into an attention layer

$$Self\text{-}Attention(E) = Attention\left(EW^Q, EW^K, EW^V\right), \tag{2}$$

where the projection matrices $W^Q, W^K, W^V \in \mathbb{R}^{d\times d}$ can make the model more flexible. For instance, the model can learn asymmetric interactions (i.e., <query i, key j > and <query j, key i > can be different).

We remark that the self-attention mechanism serves as an extremely crucial component in architectures such as Transformer which has been playing an important role in modeling sequential user behaviors. Therefore, an optimal self-attention representation architecture is able to significantly boost sequential recommendation performance.

4 AutoDisenSeq with LLM

In this section, we first introduce our proposed *Intention Disentangled Sequence Encoding Strategy* which achieves a disentangled sequential recommendation with latent self-supervision, then we in detail discuss *Automated Attention Representation Search* which uses NAS to discover the optimal attention representation architecture for sequential recommendation, and finally we utilize the rich general knowledge and power in text understanding of LLM to improve the recommendation prediction accuracy. Last but not least, via incorporating all these elements, we obtain our proposed LLM-guided AutoDisenSeq model, i.e., AutoDisenSeq-LLM.

4.1 Intention Disentangled Sequence Encoding

4.1.1 Sequence Encoder. A sequence encoder takes a sequence, i.e., user's recent clicked items, as input, and outputs an embedding representing the whole sequence. In the recommendation scenario, Transformer encoders are the state-of-the-art encoders. Therefore, we adopt a recent variant of Transformer, that is, the SASRec Encoder [60]. Specifically, SASRec builds a sequential recommendation model by stacking *self-attention* layers and uses a set of trainable positional embeddings to encode the order of items in a sequence. Beyond that, SASRec also reuses the item embedding table so that the target would share the same encoding space with the input sequence. However, we note that changing SASRec from single-head to multi-head does not induce a great improvement. This may be due to that the multiple vector representations outputted by the multi-head version of SASRec fail to achieve semantic disentanglement and hence do not surpass the single vector representation, which already captures most of the information.

4.1.2 Disentangle the Latent Intentions. Based on the SASRec sequence encoder, we further propose to add a disentanglement layer, which follows a single-head SASRec encoder, to derive the multiple intentions buried in the encoding as well as utilizing the expressive power of SASRec. Let $\{z_1^{(n)}, z_2^{(n)}, \cdots, z_m^{(n)}\}$, where $z_i^{(n)}$ be the output of a single-head SASRec encoder as i position, i.e.,

$$\left\{z_1^{(n)}, z_2^{(n)}, \cdots, z_m^{(n)}\right\} = SASRec\left(\left\{s_1^{(n)}, s_2^{(n)}, \cdots, s_m^{(n)}\right\}\right). \tag{3}$$

The disentanglement layer starts by clustering encoding $\mathbf{z}$ to a set of hyper prototypical intention representations under K latent categories, that is $\{c_1, c_2, \cdots, c_K\}$, where $c_i \in \mathbb{R}^D$, $i = 1, 2, \cdots, K$.

$$q_i^{(k)} = \frac{\exp\left[\frac{1}{\sqrt{D}}\text{LayerNorm}_1\left(z_i^{(n)}\right) \cdot \text{LayerNorm}_2\left(c_k\right)\right]}{\sum_{s=1}^{K} \exp\left[\frac{1}{\sqrt{D}}\text{LayerNorm}_1\left(z_i^{(n)}\right) \cdot \text{LayerNorm}_2\left(c_s\right)\right]}, \tag{4}$$

where LayerNorm represents a layer-normalization layer, which is distinguished through subscripts. We use cosine instead of dot product to reduce mode collapse, that is where most candidates are assigned to the same category due to the findings of previous work [90].

The clustering procedure derives $q_i^{(k)}$, which naturally measures the similarity between the intention at position i, i.e., $z_i^{(n)}$ and latent category k. We further calculate the relation between $z_i^{(n)}$ and the user's future intention

$$q_i = \frac{\exp\left[\frac{1}{\sqrt{D}} z_i^{(n)\prime} \cdot \text{LayerNorm}_3\left(z_M^{(n)}\right)\right]}{\sum_{s=1}^{m} \exp\left[\frac{1}{\sqrt{D}} z_s^{(n)\prime} \cdot \text{LayerNorm}_3\left(z_M^{(n)}\right)\right]}, \tag{5}$$

where $z_M^{(n)} = \boldsymbol{\alpha}_m + z_m^{(n)} + \mathbf{b}'$ and $z_s^{(n)\prime} = \text{LayerNorm}_4\left(\boldsymbol{\alpha}_s + z_s^{(n)}\right) + \mathit{RELU}\Big(\mathbf{W}^T\big(\text{LayerNorm}_4\big(\boldsymbol{\alpha}_s + z_s^{(n)}\big)\big) + \mathbf{b}\Big)$. Here $\boldsymbol{\alpha}, \mathbf{b}, \mathbf{b}' \in \mathbb{R}^D$ and $\mathbf{W} \in \mathbb{R}^{D \times D}$ are all trainable parameters, and $\boldsymbol{\alpha}_s$ serves as a positional embedding for position s. Since the latest action usually plays the most important role in prediction, we use $z_M^{(n)}$ to stand for user's probable future intention and calculate the similarity between each $z_i^{(n)}$ and $z_M^{(n)}$ by clustering, adding $\mathbf{b}, \mathbf{b}'$ and $\mathbf{W}$ to simulate bias and transformation.

Finally, given q_i and $q_i^{(k)}$, we can aggregate the intention under each latent category throughout the positions to derive the kth disentangled sequence encoding:

$$\varphi_\theta^{(k)} = \text{LayerNorm}_5\left(\boldsymbol{\beta}_k + \sum_{i=1}^{t} q_i^{(k)} \cdot q_i \cdot z_i^{(n)}\right), \tag{6}$$

where $k = 1, 2, \cdots, K$ and $\boldsymbol{\beta}_k \in \mathbb{R}^D$ represents bias for latent category k. Note that two different sets of $\{\boldsymbol{\beta}_i\}_{i=1}^K$ are used for encoding input sequence and label sequence, respectively.

4.1.3 Seq2Seq Training Strategy. In this section, we discuss our Seq2Seq training strategy which optimizes **Sequence-to-Item (S2I)** loss and **Sequence-to-Sequence (S2S)** loss simultaneously. Each element in a mini-batch $\mathbb{B}$ is a set of sampled sequence from the training set $\{(n, m) : 1 \leq n \leq N\}$. We divide each training sample into an earlier sequence and its corresponding future sequence. We denote the earlier sequence as $s_{1:i}^{(n)} = \left[s_1^{(n)}, s_2^{(n)}, ..., s_i^{(n)}\right]$ and the future sequence as $s_{i+1:m}^{(n)} = \left[s_{i+1}^{(n)}, s_{i+2}^{(n)}, ..., s_m^{(n)}\right]$. From Section 4.1.2, we derive an intention disentangled sequence encoder $\varphi_\theta(\cdot)$. The input of the encoder is the earlier sequence $s_{1:i}^{(n)}$. The output of the encoder is K vectors in D-dimensional space which are denoted as $\varphi_\theta^{(k)}\left(s_{1:i}^{(n)}\right)$, representing a user's intention for K different latent item categories. In addition, if the input sequence $s_{1:i}^{(n)}$ does not contain any items under the kth latent category, we assume that the value of $\varphi_\theta^{(k)}\left(s_{1:i}^{(n)}\right)$ is a temporary noise vector. In general, $\varphi_\theta^{(k)}\left(s_{1:i}^{(n)}\right)$ can also be regarded as a prototype item, i.e., a pseudo item that summarizes the clicked items under the kth latent category.

S2I Loss. The *S2I* loss is widely used in sequential recommendation tasks. Its input is the user's historical behavior sequence, and its label is the user's next behavior. In other words, the *S2I* training strategy generally takes the interaction sequence before the ith moment as input and takes the $i + 1$th moment as supervised label information to train the model. It is useful when training an encoder and aligning the vector space of sequences and items in a relatively short period of time. The objective function of *S2I* loss is defined as follows:

$$\mathcal{L}_{S2I}(\theta, \mathbb{B}) = \sum_{(n,m)\in\mathbb{B}} \mathcal{L}_{S2I}(\theta, n, m) \tag{7}$$

$$= - \sum_{(n,m)\in\mathbb{B}} \ln p_\theta\left(h_{i+1}^{(n)} \middle| \varphi_\theta^{(k)}\left(s_{1:i}^{(n)}\right)\right) \tag{8}$$

$$= - \sum_{(n,m)\in\mathbb{B}} \ln \frac{max_{k\in\{1,2,\ldots,K\}} \exp\left(\frac{1}{\sqrt{D}} h_{i+1}^{(n)} \cdot \varphi_\theta^{(k)}\left(s_{1:i}^{(n)}\right)\right)}{\sum_{(n',m')\in\mathbb{B}} \sum_{k'=1}^{K} \exp\left(\frac{1}{\sqrt{D}} h_{i'+1}^{(n')} \cdot \varphi_\theta^{(k')}\left(s_{1:i}^{(n)}\right)\right)}, \tag{9}$$

where $h_{i+1}^{(n)} \in \mathbb{R}^D$ is the representation of item $s_{i+1}^{(n)}$. The item embedding table of row $s_{i+1}^{(n)}$ can be denoted as $H \in \mathbb{R}^{M\times D}$.

S2S Loss. We argue that only considering the *S2I* loss is short-sighted, which may run the risk of resulting in a lack of diversity in recommendations. As such, the *S2S* employed by our Seq2Seq training strategy uses the user's historical sequence to predict the user's possible future sequence, which benefits in more robustness and diversity. The objective function of *S2S* loss is defined as follows:

$$\mathcal{L}_{S2S}(\theta, \mathbb{B}) = \sum_{(n,m)\in\mathbb{B}} \sum_{k=1}^{K} \mathcal{L}_{S2S}(\theta, n, m, k) \cdot \mathbf{1}[\mathcal{L}_{S2S}(\theta, n, m, k) \le \tau] \tag{10}$$

$$= - \sum_{(n,m)\in\mathbb{B}} \sum_{k'} \ln p_\theta\left(\varphi_\theta^{(k')}\left(s_{m:i+1}^{(n)}\right) \middle| \varphi_\theta^{(k')}\left(s_{1:i}^{(n)}\right)\right) \tag{11}$$

$$= - \sum_{(n,m)\in\mathbb{B}} \sum_{k'} \ln \frac{\exp\left(\frac{1}{\sqrt{D}} \varphi_\theta^{(k')}\left(s_{m:i+1}^{(n)}\right) \cdot \varphi_\theta^{(k')}\left(s_{1:i}^{(n)}\right)\right)}{\sum_{(n',m')\in\mathbb{B}} \sum_{t=1}^{K} \exp\left(\frac{1}{\sqrt{D}} \varphi_\theta^{(t)}\left(s_{m':i'+1}^{(n')}\right) \cdot \varphi_\theta^{(k')}\left(s_{1:i}^{(n)}\right)\right)}, \tag{12}$$

where we scale the dot product scores by a factor of $\frac{1}{\sqrt{D}}$ because the last layer of the encoder is a layer-normalization one, and the scaling factor helps convergence. Here, the softmax is normalized over the samples that appear in the current mini-batch $\mathbb{B}$ instead of all samples in the training set to reduce computational complexity. We use the reversed sequence $s_{m:i+1}^{(n)}$ instead of the original $s_{i+1:m}^{(n)}$, because our encoder weights the items in the sequence according to their temporal order and we want the items closer to position i to receive higher weights. Moreover, only a selected subset of $S_{\mathbb{B},K} = \{\mathcal{L}_{S2S}(\theta, n, m, k) : (n, m) \in \mathbb{B}, 1 \le k \le K\}$ should be included for training. For instance, if the input sequence $s_{1:i}^{(n)}$ shares intention categories $\{k : k \in K'\}$ with label sequence $s_{i+1:m}^{(n)}$ rather than the whole set $\{k : 1 \le k \le K\}$, then we should only use $k \in K'$, i.e., $\{\mathcal{L}_{S2S}(\theta, n, m, k) : (n, m) \in \mathbb{B}, k \in K'\}$. In our method, the threshold τ is the $\lceil\lambda \cdot |\mathbb{B}| \cdot K\rceil$th smallest value in $S_{\mathbb{B},K}$, where $\lambda \in [0, 1]$ is a hyper-parameter. This is based on the assumption that the smaller the loss, the higher the probability that the input and the label sequence share the same intention under the corresponding latent category. In particular, λ is incorporated as a mask to filter $\mathcal{L}_{S2S}(\theta, n, m, k)$ with a large *S2S* loss, indicating that the input and the label sequence do not share the same intention under latent category k. As such, this item is regarded as a noise and should not be included in training. λ lies in $[0, 1]$, which means we only keep the smallest λ percent of the whole $\mathcal{L}_{S2S}$ set.

However, the *S2S* loss still suffers from two problems. First, it is more difficult to construct future sequences of behaviors than predict a single item, so it may not converge; second, the user's future behavior may contain multiple intentions and not every intention can be predicted from the earlier behavior. To sum up, *S2S* strategy should not completely replace the *S2I* strategy, hence, we derive the following *Seq2Seq* training loss which takes the advantages of both *S2S* and *S2I* strategies into account

$$\mathcal{L}_{Seq2Seq}(\theta, \mathbb{B}) = \mathcal{L}_{S2I}(\theta, \mathbb{B}) + \mathcal{L}_{S2S}(\theta, \mathbb{B}), \tag{13}$$

where θ represents the model parameters. We need to note that our designed loss functions (see the numerators in Equations (9) and (12)) encourage the model to preserve different information in $\varphi_\theta^{(k)}$ than the other $k-1$ parts. Therefore, the disentanglement across different latent categories is ensured.

Based on the derived intention disentangled sequence encoder and the end-to-end training strategy, we have already completed the whole recommendation task, where we can recommend new items to users by the similarity between items and the input sequence in the vector space to sort the candidates and make predictions. However, the self-attention layers in the sequence encoder, i.e., SASRec, are fixed, making it difficult to find the optimum architecture and limiting its capabilities to effectively adapt to various environments.

4.2 Automated Attention Representation Search

In this section, we propose an automated framework that automates the self-attention representation in our sequence encoder. In order to achieve the optimum performance, it is necessary to search for the attention representation along with the other components of the deep neural network to achieve the global optimum. Therefore, our search space contains both the original search space of NAS and the attention representation.

4.2.1 Attention Representation Search Space. The search space defines the scope of the search, which typically contains the type and number of layers as well as the type of connections and the parameters within. A modern neural network can be considered as a set of layers with optional connections between any two of the layers. More specifically, the layers can be treated as information aggregator, adding all the information it receives and processing it before passing it to the next layer. Therefore, we can simplify all the original layers to be the same type, i.e., *addition layer*.

Since the hyper-parameters depend on the type of operation, the result of parameterization for search space is not fixed-length but conditional. We formulate the source layer and operation selection processes as the following equation:

$$S^Q = XW^Q, S^K = XW^K, S^V = XW^V, \tag{14}$$

$$Q = O^Q(S^Q), K = O^K(S^K), V = O^V(S^V), \tag{15}$$

where S^Q, S^K, and S^V are the input source layers, O^Q, O^K, and O^V are operation choices selected from the original operation library, and K, Q, V are the outputs. We automate the design of attention representations by introducing such a novel aggregation layer called *attention layer* and calculate attention as follows:

$$Q = O^Q(S^Q), K = O^K(S^K), V = O^V(S^V), \tag{16}$$

$$Attend_{Q->K} = Sim(Q, K), \tag{17}$$

$$Out = Attend_{Q->K}V. \tag{18}$$

In conclusion, the architectures can be described as a bunch of choices. For each layer, we search for the layer type and determine its source layers and operations.

4.2.2 Search Space Complexity Analysis. Assume that the number of hidden layers is W, and the size of the primitive operation pool is b, then the total number of architectures included in the defined search space above is $\prod_{k=1}^{W}\left(k^2b^2 + k^3b^3\right) \in O\left(n!^4b^{3W}\right)$. However, this search space is still difficult for searching since it contains many isomorphic architectures. Therefore, we propose two constraints to further reduce the complexity without harming expressivity.

Constraint 1. We first constrain the architecture to be a chain by forcing each layer to have at least one connection from its last layer. For the *addition layer* and the *attention layer*, input connection and *query* connection are bind to the last layer, respectively. Moreover, we also restrict the choices of these operations to be nonzero. These constraints shape a chain neural network, where the information mainly flow along the skeleton. This chain-like macro search space is widely adopted in previous NAS works [40, 43].

Constraint 2. For the attention layer, we further bind the *key* and *value* connection to have the same source layer. This is due to the presumption that *key* and *value* serve as *memory* in attention blocks, thus should have similar semantic meanings [93, 100].

With the constraints mentioned above, the complexity of the search space is reduced to

$$\prod_{k=1}^{W}\left(b(b-1)k + (b-1)^3k\right) \in O\left(n!^2b^{3n}\right), \tag{19}$$

which greatly eases the pressure for NAS while still retaining the high expressive power.

4.2.3 Search Algorithm. The search for the optimum architecture generally can be formulated as a bi-level optimization problem

$$\begin{aligned} a^* &= \arg\min_{a\in\mathbb{A}} L_{val}\big(a, w^*(a)\big), \\ s.t.\ \ w^*(a) &= \arg\min_{w\in W(a)} L_{train}(a, w), \end{aligned} \tag{20}$$

where $\mathbb{A}$ stands for the search space, a is a candidate architecture in $\mathbb{A}$, $W(a)$ is the parameter space given the fixed architecture a, $w^*(a)$ is the best parameter in $W(a)$, and a^* is the best architecture of the task and the output of NAS. Solving the bi-level optimization problem in Equation (20) directly can be resource intensive because the architecture needs to be fully trained from scratch for the internal optimization. To make the search process more efficient, the bi-level optimization can be split into two separate optimization problems

$$\begin{aligned} a^* &= \arg\min_{a\in\mathbb{A}} L_{val}\big(a, w^*(a)\big), \\ s.t.\ \ w^*(a) &= \arg\min_{w\in W} \mathbb{E}_{a\sim\Gamma(\mathbb{A})} L_{train}(a, w), \end{aligned} \tag{21}$$

where $\mathbb{E}_{a\sim\Gamma(\mathbb{A})}$ is the prior architecture distribution of $a \in \mathbb{A}$, and W is the trainable weights of the supernet, which contains all the architectures in the search space and shares the set of parameters with them. The process reduces the time cost since optimizing the supernet is quite faster than training all the architectures from scratch. After deriving the parameters of the supernet, we perform genetic algorithm in a single path one-shot manner [43] to search for the optimum architecture

$$a^* = \arg\min_{a \in \Gamma(\mathbb{A})} L_{val}(a, w^*(a)), \tag{22}$$

where we randomly sample architectures $a \in \Gamma(\mathbb{A})$ and perform *crossover*, *mutation* as in genetic search to find the best architecture on the validation set. We note that a inherits its weight from the weight of the supernet w^* as $w^*(a)$, and the search process only requires inference, thus being effective.

4.2.4 Context-Aware Parameter Sharing Technique. In our search space, the parameters of the connections depend heavily on its contexts, which are the type of layers that the connections are connected with. Thus, directly sharing the parameters of the same operations at the same places without considering the contexts as in previous works fails to model the specific characteristics of each architecture.

Therefore, in our model, we only share parameters within the same contexts. In our proposed automated neural network, there are two types of layers, namely the original *addition layer* and the added *attention layer*. Hence, naturally there are four types of connections, with the random combination of *addition layer* and *attention layer*. In addition, when the target layer is an *attention layer*, the connection is needed to be distinguished among *Query*, *Key*, and *Value*, respectively. This raises the total amounts of contexts to eight, and for each of these contexts, we assign independent parameters for a single connection in the supernet. Although this technique increases the total parameters by eight times, we only sample a single sub-architecture to optimize at each step using Monte-Carlo, keeping the total number of parameters same as before.

4.2.5 Automated Disentangled Seq2Seq Training Strategy. We include the intention disentangled sequence encoder into the automated framework and propose the whole end-to-end learning objective. Via putting all the objectives together, the whole model achieves high performance in recommendation while maintaining an automatic training procedure free of human labor. Specifically, we optimize the following loss to train our model:

$$Loss = \mathcal{L}^{(val)}_{Seq2Seq}\big(a, \arg\min_{\theta} \mathbb{E}_{a' \sim \Gamma(A)} \mathcal{L}^{(train)}_{Seq2Seq}(a', \theta)\big), a \in A. \tag{23}$$

The meaning of the symbols is the same as in Sections 4.2.3 and 4.1.3. As such, the learning objective of our proposed AutoDisenSeq model is a joint optimization of both model parameters, i.e., intention disentangled Seq2Seq loss and model architectures, i.e., automated representation search. The overall training process is shown in Figure 1, and the detailed optimization procedure is illustrated in Algorithm 1. In the inner loop, we use the gradient descent to update the parameters and after the parameters are optimized, we use EA to search the best architecture through the validation loss.

4.3 LLM-Guided Sequential Recommendation

While the proposed AutoDisenSeq model with Automated *Seq2Seq* training strategy is effective, it can only obtain existing behavioral knowledge without accessing general knowledge of the candidate items, e.g., AutoDisenSeq only has knowledge of the training dataset but does not know some general intentions of users. Therefore, we proposed AutoDisenSeq-LLM, an LLM-guided variant that utilizes LLM to incorporate the general knowledge may further improve the recommendation performance.

Specifically, a pretrained LLM (ChatGPT, in this article) is asked to determine whether to modify the prediction results of the AutoDisenSeq recommendation model, given the item descriptions, i.e., title of a movie, content and price of a video game, and so on. Since the abilities of the base

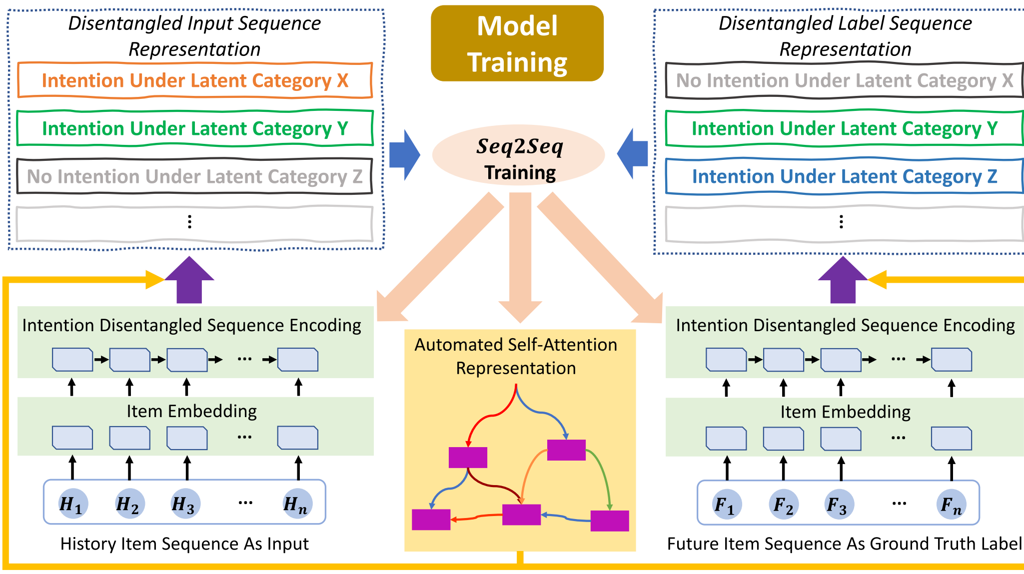

Fig. 1. The overall training procedure of the proposed AutoDisenSeq model. The history item sequence of the target users is first mapped into the item embedding space via the learnable item embedding function. Then, the Transformer-based intention disentangled sequence encoder, whose self-attention representation architecture is tailored to reach optimal performance for the current recommendation task, will be employed to obtain disentangled latent representation for the whole history sequence of the target user. As we can observe from the figure, the disentangled sequence representation may contain different blocks of dimensions indicating different latent categories (e.g., *Category X, Y, and Z*). A similar procedure will be used for ground truth item sequence to obtain the disentangled latent sequence representation. To jointly discover the optimal attention representation architecture as well as the model parameters, the *Seq2Seq* training strategy simultaneously optimizing the Sequence-to-Sequence (*S2S*) loss and Sequence-to-Item (*S2I*) loss will be employed as the objective function, which explicitly considers various categories for model training.

recommendation model are relatively strong, LLM only plays the role of denoising and refining. We discover that the Top-5 items from the recommended results are of high confidence, thus, we use LLM to only finetune the results, rearranging the recommendation list except for the Top-5 items. Figure 3 illustrates an example of the interaction process in which LLM replies with the sorted recommendation list based on item descriptions. In the carefully designed prompt, we organize the information, including commands and descriptions in an easy-to-understand manner, in order that LLM can comprehend and respond accurately. The overall recommendation procedure of our proposed AutoDisenSeq-LLM model is shown in Figure 2.

5 Empirical Experiments

We conduct empirical experiments to validate the performance of our approach by comparing our proposed AutoDisenSeq and AutoDisenSeq-LLM models with several state-of-the-art methods on four real-world datasets. The results show that both AutoDisenSeq and AutoDisenSeq-LLM are powerful sequential recommendation models with automatic attention representation mechanism, beating all other baselines in terms of recommendation accuracy. The tasks and datasets are introduced in Section 5.1. The descriptions of comparative approaches are presented in Section 5.2. The implementation details are described in Section 5.3. Experimental results and analyses are shown in Section 5.4.

Algorithm 1: Automated Disentangled Sequential Recommendation (AutoDisenSeq)

Input: Click sequence $s^{(n)}$ for each user n, dividing into earlier sequence $\left\{s_1^{(n)}, ..., s_i^{(n)}\right\}$ and future sequence $\left\{s_{i+1}^{(n)}, ..., s_m^{(n)}\right\}$ as input and label, respectively.

Output: The disentangled representation for user n, $\varphi_\theta\left(s^{(n)}\right)$.

function $\textsc{IntentionDisentangledSequenceEncoder}(s^{(n)}, a')$
  $z^{(n)} \leftarrow \mathrm{SASEncoder}\left(s^{(n)}, a'\right)$
  for $i = 1, 2, \cdots, m$ **do**
    for $k = 1, 2, \cdots, K$ **do**
      $q_i^{(k)} \leftarrow \mathrm{IntentionClustering}\left(z_i^{(n)}, c_k\right)$ (Equation (4))
    end for
  end for
  for $i = 1, 2, \cdots, m$ **do**
    $q_i \leftarrow \mathrm{FutureIntentionRelation}\left(z_i^{(n)}, z^{(n)}\right)$ (Equation (5))
  end for
  for $k = 1, 2, \cdots, K$ **do**
    $\varphi_\theta^{(k)} \leftarrow \mathrm{IntentionAggregating}\left(q^{(k)}, q, z^{(n)}\right)$ (Equation (6))
  end for
  return $\varphi_\theta = \left[\varphi_\theta^{(1)}, \varphi_\theta^{(2)}, \cdots, \varphi_\theta^{(K)}\right]$
end function
while not converged **do**
  for every mini-batch **do**
    $\varphi_\theta\left(s_{1:i}^{(n)}\right) \leftarrow \textsc{IntentionDisentangledSequenceEncoder}(s_{1:i}^{(n)}, a')$
    $\varphi_\theta\left(s_{m:i+1}^{(n)}\right) \leftarrow \textsc{IntentionDisentangledSequenceEncoder}(s_{m:i+1}^{(n)}, a')$
    $\mathcal{L}_{S2I} \leftarrow Loss_{S2I}\left(h_{i+1}^{(n)}, \varphi_\theta\left(s_{1:i}^{(n)}\right)\right)$
    $\mathcal{L}_{S2S} \leftarrow Loss_{S2S}\left(\varphi_\theta(s_{1:i}^{(n)}), \varphi_\theta\left(s_{m:i+1}^{(n)}\right)\right)$
    $\mathcal{L}_{Seq2Seq} \leftarrow \mathcal{L}_{S2I} + \mathcal{L}_{S2S}$
    $\theta \leftarrow \arg\min_{\theta'} \mathbb{E}_{a' \sim \Gamma(A)} \mathcal{L}_{Seq2Seq}^{(train)}\left(a', \theta'\right)$
  end for
end while
$a \leftarrow \arg\min_{a \in A} \mathcal{L}_{Seq2Seq}^{(val)}\left(a, \mathbb{E}_{a' \sim \Gamma(A)} \mathcal{L}_{Seq2Seq}^{(train)}(a', \theta)\right)$

5.1 Datasets and Evaluation Metrics

5.1.1 Datasets Description. We conducted experiments on four datasets that were collected from three real-world platforms with different domains and densities. The statistics of these datasets after pre-processing are summarized in Table 1.

- **MovieLens-1m (ML-1m)**: The ML-1m dataset [45] consists of 1 million ratings from 6,000 users on approximately 4,000 movies. It includes user demographics, movie details, and timestamps. In this work, we use the titles and genres of the movies as corresponding attributes for the LLM.

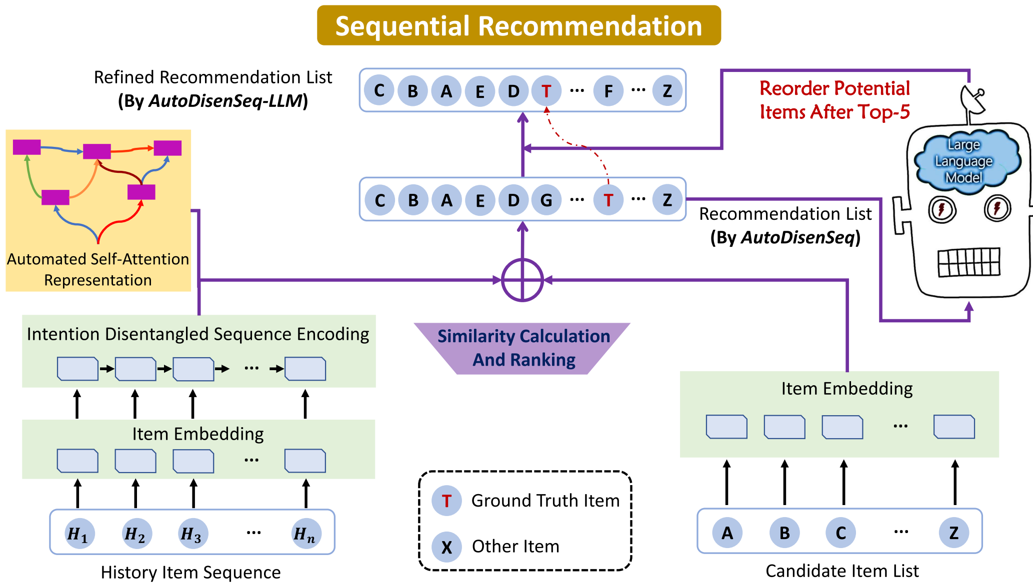

Fig. 2. The overall recommendation procedure of our proposed AutoDisenSeq with LLM (AutoDisenSeq-LLM). During the recommendation phase, the history item sequence of the target user will first be mapped to the item embedding space and then be passed through the sequence encoder with automated representation architecture to form the latent sequence representation. The candidate items will also be mapped to the item embedding space before calculating similarity with the history latent sequence representation. The candidate items will be sorted in descending order based on the calculated similarity (i.e., recommendation list by AutoDisenSeq). We empirically discover that the Top-5 items in the recommendation list by AutoDisenSeq are of high confidence. Therefore, a pretrained LLM (i.e., ChatGPT) is employed to finetune the results via reordering the potentially promising items except for the Top-5 items in the recommendation list returned from AutoDisenSeq to obtain the refined recommendation list from AutoDisenSeq-LLM. For instance, the ground truth item **T** is first ranked far after item **F** in the recommendation list by AutoDisenSeq and will be reordered to the six place in the refined recommendation by AutoDisenSeq-LLM after the finetuning process by the LLM. This example demonstrates the necessity and efficacy of AutoDisenSeq-LLM through utilizing the text understanding power of LLM to further improve the recommendation accuracy.

Table 1. Dataset Statistics

Dataset	Users	Items	Actions	Density (%)
ML-1m	6,038	3,308	841,827	4.214
Steam	120,269	22,884	2,852,115	0.104
Beauty	12,060	35,578	165,925	0.038
Game	7,973	19,671	141,354	0.090

— Steam: The Steam dataset [60] is crawled from a large online video game distribution platform called *Steam*. It contains over 2 million users and approximately 15,000 games. In this work, we utilize the title, price, and category of the video game as the attributes for the LLM.
— Amazon Beauty and Amazon Game: The Amazon Review Dataset [99] is a classic dataset for RS, which is divided into subsets, such as Books, Electronics, Movies, TV, CDs, and so on. It contains 142.8 million product reviews (ratings, text, help polls), metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also purchased

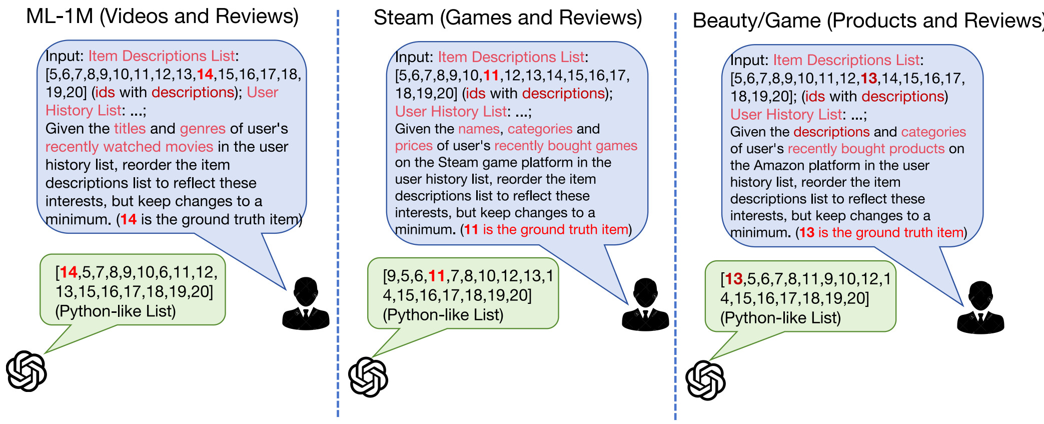

Fig. 3. Showcases of interactions with LLM (ChatGPT) on ML-1m, Steam, Amazon Beauty, and Game datasets for experiments (from left to right). We carefully design prompts so that the commands and descriptions fed into the LLM are easy to understand by ChatGPT. Take the Amazon dataset as an example, and we design the prompt as: **Input:** Item Descriptions List: [5,6,7,8,9,10,11,12,**13**,14,15,16,17, 18,19,20]; (ids with descriptions) User History List:...; Given the descriptions and categories of user's recently bought products on the Amazon platform in the user history list, reorder the item descriptions list to reflect these interests, but keep changes to a minimum. (**13** is the ground truth item) Then, the LLM will reply with the refined recommendation list in the form of a Python-like list as follows: [**13**,5,6,7,8,11,9,10,12,14,15,16,17,18,19,20]. ML-1m, MovieLens-1m.

charts) on Amazon.com from May 1996 to July 2014. In our experiments, we select two subcategories: (1) *Beauty* and (2) *Game*, and use the granular categories and brands of the products as their attributes for the LLM.

For all datasets, we group the interaction records by the user and sorted them by interaction timestamp in ascending order. Following [110, 118], we only adopt the 5-core dataset, filtering out unwanted items and inactive users with less than five interaction records.

5.1.2 Evaluation Metrics. In our experiments, we adopt widely used metrics in related work to evaluate the performance of our model, including **top-k hit ratio (HR@k)** and **top-k normalized discounted cumulative gain (NDCG@k)**. In our experiment on overall recommendation accuracy, we report results for HR@10 and NDCG@10.

We follow the previous work and apply the leave-one-out method for evaluation. Specifically, for each user interaction sequence, the last item was used as test data, some items before the last one as validation data, and the remaining as training data. Since the item set is massive, taking all items as candidates for testing is time-consuming. Therefore, we follow BERT4Rec's advice and pair each authentic item in the test set with 100 negative items randomly sampled according to their popularity, which is a common practice in the literature. The recommendation task then becomes to identify which item among these 101 items is the ground-truth next item for each user. We calculate all metrics based on the ranking of the items and report the average score of the overall test users. Higher values in all these metrics reflect better recommendation performance.

5.2 Comparative Approaches

Baselines. The following five baseline methods are used for comparisons.

— ICLRec [17]: This model learns user's latent intents via contrastive learning and leverages them effectively for sequential recommendation.

—STOSA [34]: This model treats the embedding of items as a stochastic Gaussian distribution, and they propose a module using Wasserstein distance to calculate the similarity between different items.
—BERT4Rec [117]: This model adopts the Cloze objective to the sequential recommendation and predicts the masked item by jointly using the left and the right context.
—SASRec [60]: This model effectively utilizes the self-attention mechanism and achieves good performance in sequence recommendation.
—DSSRec [91]: This model adopts a SASRec encoder and combines disentangled representation learning and self-supervised learning to balance the weights of multiple interests.

Our Proposed Models. We compare our proposed AutoDisenSeq and AutoDisenSeq-LLM models with the five baselines.

—AutoDisenSeq: Our proposed Automated Disentangled Sequence Representation model, which learns user's latent sequence representation in a disentangled manner with optimal self-attention representation architecture for sequential recommendation.
—AutoDisenSeq-LLM: Our proposed AutoDisenSeq with LLM. This model utilizes the power of LLM in understanding textual information to further improve the recommendation accuracy via finetuning the results returned by AutoDisenSeq.

5.3 Implementation

We follow the experimental setup described in BERT4Rec [117], a widely used state-of-the-art recommendation model based on BERT. We implement our model in PyTorch and initialize the parameters using default initialization. We use the Adam [62] optimizer for mini-batch gradient descent. We set the learning rate to 0.001 and batch size to 256 and cap the maximum sequence length to 50 for all four datasets. We follow the structure of a single-head SASRec [60] encoder and automate the choices for inner layers' type and connections. Then, we tune the other hyper-parameters using ASHA [75] implemented by Ray Tune [78]. Specifically, we run the ray tune with 500 samples under each setting, with the hyper-parameter search space specified as follows: $D \in \{64, 128, 256\}$, *number of hidden layers* $\in \{2, 3, 4, 5, 6\}$, $\lambda \in \{0.1, 0.2, \cdots, 0.6\}$.

5.4 Experimental Results

5.4.1 Recommendation Accuracy. Table 2 presents the overall recommendation accuracy of all the methods on the four datasets. From the results, we observe that our proposed model, i.e., AutoDisenSeq-LLM, consistently outperforms all the baselines in terms of HR@10 and NDCG@10, respectively. This demonstrates the benefits of the proposed disentangled Seq2Seq recommendation, which learns user's latent intentions and builds the future sequence as a whole and automated attention representation, which automatically searches for the best architecture and LLM denoising.

5.4.2 Ablation Study.

Model Components. We perform ablation study of multiple variants of the proposed method on dataset ML-1m and list the results in Table 3. Variant (1) of our method only optimizes traditional *S2I* loss, removing all the other parts including disentangled *Seq2Seq* training, automated attention representation search, and LLM denoizing. Variant (2) of our method adds disentangled *Seq2Seq* training based on variant (1), and variant (3) additionally searches the attention representation using NAS. We observe a performance drop in variant (2) compared to variant (3), and variant (1) performs even worse. As LLM excludes the Top-5 items when denoizing, it only affects HR@10 and NDCG@10, and both improve in variant (4), i.e., the proposed AutoDisenSeq-LLM model. The results demonstrate the integrity of the whole method and the effectiveness of each component,

Table 2. The Overall Recommendation Accuracy Comparison between Our Proposed Model and the Baselines

Dataset	Metric	ICLRec	STOSA	BERT4Rec	SASRec	DSSRec	AutoDisen Seq	AutoDisen Seq-LLM
ML-1m	HR@10	0.1207	0.3170	0.6749	0.6570	0.6800	0.7040	**0.7125**
	NDCG@10	0.0489	0.1365	0.4508	0.4258	0.4563	0.4733	**0.4770**
Beauty	HR@10	0.1024	0.0779	0.2796	0.3520	0.3663	0.3726	**0.3995**
	NDCG@10	0.0461	0.0346	0.1705	0.2296	0.2517	0.2498	**0.2601**
Game	HR@10	0.1233	0.1406	0.3443	0.6251	0.6351	0.6390	**0.6634**
	NDCG@10	0.0496	0.0591	0.2232	0.3989	0.4168	0.4191	**0.4277**
Steam	HR@10	0.0783	0.1424	0.4651	0.5514	0.6212	0.6474	**0.6605**
	NDCG@10	0.0360	0.0625	0.2809	0.2560	0.4057	0.4122	**0.4176**

Bold font denotes the winner.

Table 3. Ablation Study of Multiple Variants of Our Method on Dataset ML-1m

Variants of Our Method	**Evaluation Metrics**				
	HR@1	HR@5	NDCG@5	HR@10	NDCG@10
(1) Without Disen-Seq2Seq, auto, and LLM	0.2263	0.5298	0.3843	0.6570	0.4258
(2) Without auto and LLM	0.2547	0.5656	0.4191	0.6800	0.4563
(3) AutoDisenSeq (without LLM)	**0.2655**	**0.5871**	**0.4352**	0.7040	0.4733
(4) AutoDisenSeq-LLM	-	-	-	**0.7125**	**0.4770**

Bold values denote the best performance over each metric.

validating their important roles in the model's final performances. Similar results hold on other datasets as well.

Including vs. Excluding the Top-5 Candidate Items During LLM Reordering. On the one hand, we discover the Top-5 items from the recommended candidates are of high confidence due to the effectiveness of the AutoDisenSeq model. On the other hand, the prediction results of LLM may have a certain degree of randomness and allowing it to rearrange the Top-5 candidate items may run the risk of removing the hit (right) items from the Top-5 positions, thus deteriorating the recommendation accuracy. Therefore, our proposed AutoDisenSeq-LLM model does not include the items in the Top-5 positions of the candidate list for recommendation when conducting reordering. We also perform experiments to validate our choice of excluding the Top-5 recommended items for LLM reordering. We compare AutoDisenSeq (without LLM reordering), AutoDisenSeq-LLM (LLM reordering excluding Top-5 candidates), AutoDisenSeq-LLM-Top5 (LLM reordering including Top-5 candidates) over all the four datasets, ML-1m, Beauty, Game, and Steam, whose results are shown in Table 4. From the results, we can observe a significant performance drop when we include the Top-5 candidate items for LLM reordering under most experimental settings, which is even inferior to AutoDisenSeq model that does not involve any LLM reordering. Therefore, we believe it is necessary to employ LLM for finetuning the results via reordering the candidate list excluding Top-5 items.

5.4.3 Visualizations of the Optimal Architectures. In the previous sequential recommendation models, we have to manually design the architecture of the sequence encoder, which is a time-consuming and laborious process, and the outcome probably would not be the optimum structure.

Table 4. Ablation Study of Whether to Include the Top-5 Candidate Items during LLM Reordering

Dataset	Variants of Our Method	Evaluation Metrics				
		HR@1	HR@5	NDCG@5	HR@10	NDCG@10
ML-1m	AutoDisenSeq-LLM-Top5	0.2501	0.5512	0.4091	0.6638	0.4457
	AutoDisenSeq	**0.2655**	**0.5871**	**0.4352**	0.7040	0.4733
	AutoDisenSeq-LLM	-	-	-	**0.7125**	**0.4770**
Beauty	AutoDisenSeq-LLM-Top5	0.1498	0.2666	0.2108	0.3404	0.2345
	AutoDisenSeq	**0.1533**	**0.2871**	**0.2224**	0.3726	0.2498
	AutoDisenSeq-LLM	-	-	-	**0.3995**	**0.2601**
Game	AutoDisenSeq-LLM-Top5	**0.2441**	0.4680	0.3620	0.6037	0.4093
	AutoDisenSeq	0.2324	**0.5096**	**0.3773**	0.6390	0.4191
	AutoDisenSeq-LLM	-	-	-	**0.6634**	**0.4277**
Steam	AutoDisenSeq-LLM-Top5	0.2189	0.4672	0.3465	0.6197	0.3957
	AutoDisenSeq	**0.2267**	**0.4884**	**0.3609**	0.6474	0.4122
	AutoDisenSeq-LLM	-	-	-	**0.6605**	**0.4176**

AutoDisenSeq-LLM-Top5 denotes the variant including Top-5 candidate items for LLM reordering. Bold values denote the best performance for each dataset.

We visualize the final searched architecture for the four datasets in Figure 4, where ML-1m and Steam have 5 hidden layers, and the rest 2 have 10 hidden layers. We can observe that the final structure is complicated, especially on dataset Beauty and Game. For example, *Layer 9* in the architecture for dataset Beauty has *value* and *key* connection from all the way back to *Layer 2*. Moreover, the choice of *addition layer* and *attention layer* also plays an important role in the final structure. It is very difficult to obtain this structure manually, let alone a deep neural network with many more hidden layers, which validates the necessity and effectiveness of our proposed automated attention representation search.

5.4.4 Cold-Start Setting. Cold-start problem is frequently encountered in the RS. In our experiment, we limit the length of input sequence, i.e., user's recent interaction items, in the training and inference stage to simulate the scarce historical interaction information under cold-start setting. Specifically, the maximum length of input sequence is set to 5, compared to 50 in normal recommendation. The results are depicted in Figure 5. We observe that AutoDisenSeq-LLM still outperforms other baselines (SASRec and DSSRec, two relatively strong ones among all baselines) significantly in general, which indicates the strong practicality of our proposed model in real recommendation scenarios. The comparisons between AutoDisenSeq and AutoDisenSeq-LLM also demonstrate that the improvement of AutoDisenSeq-LLM under cold-start setting is largely brought by the incorporation of LLM. We analyze this may be the reason that LLM has rich prior knowledge through pretraining on large-scale data. The cold-start problem mainly arises due to the lack of prior information on user preferences, which may be appropriately mitigated by the extensive prior knowledge inherently carried by LLM.

5.4.5 Time Consuming Comparison. In this section, we compare the difference in training time between our method and other baselines in Table 5. Firstly, since our method introduces more

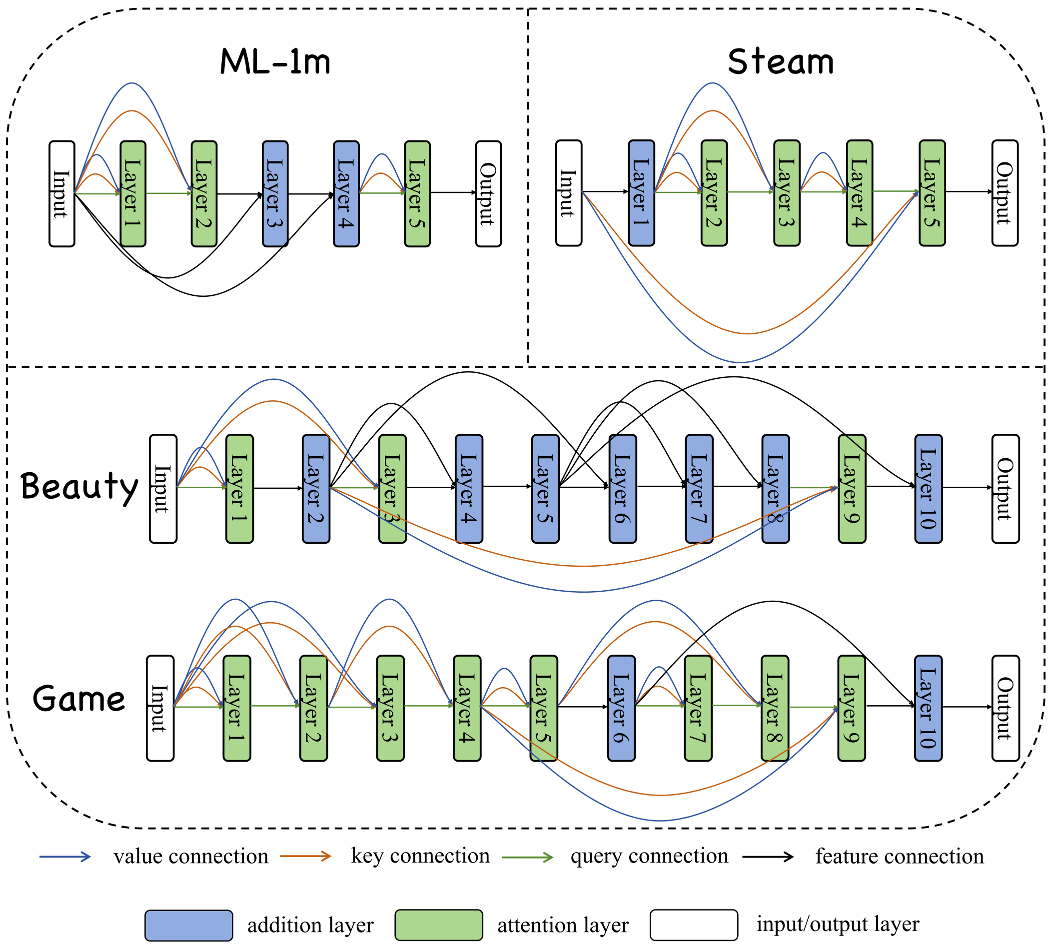

Fig. 4. Visualizations of the searched self-attention representation architectures on ML-1m, Steam, Amazon Beauty, and Game datasets (the best view in color). Different colors of arrows denote different types of connections (vale connection, key connection, query connection, and feature connection), and different colors of blocks indicate different types of layers (addition layer and attention layer). We observe that ML-1m and Steam require the optimal number of layers of 5, while Amazon Beauty and Game require the optimal layer number to be 10. Besides, it is obvious that different datasets indeed have different optimal attention representation architectures, validating the necessity of our proposed AutoDisenSeq model on searching the optimal architectures for different datasets.

Table 5. Comparison on Time Consumption (Minute) of Different Methods on the Four Datasets

Method	**Dataset**			
	ML-1m	Beauty	Steam	Game
SASRec	80	240	1,500	115
DSSRec	600	550	4,500	250
AutoDisenSeq-LLM	900	600	6,000	300

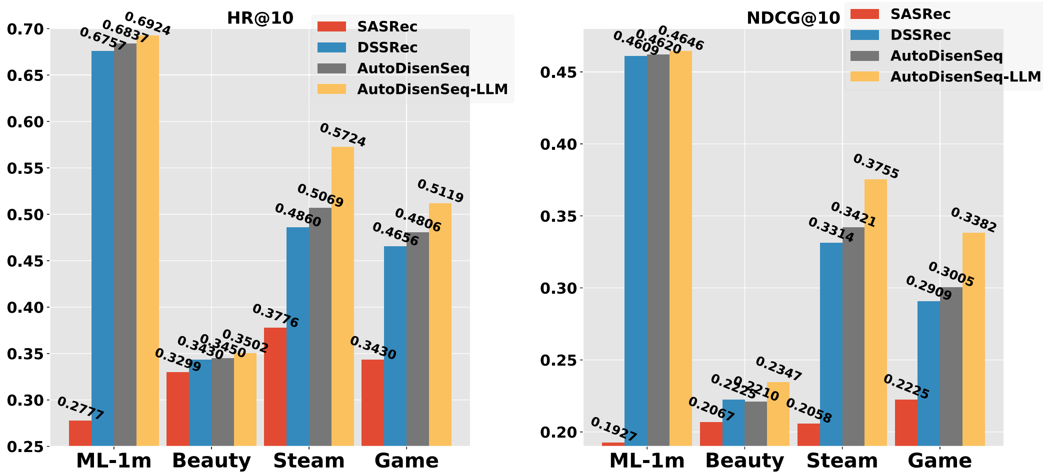

Fig. 5. Recommendation accuracy under *cold-start* setting in terms of HR@10 and NDCG@10 on the four datasets. We can observe that our proposed AutoDisenSeq-LLM is able to significantly outperform state-of-the-art Transformer-based sequential recommendation approaches in the challenging cold-start scenario over all four datasets. We also remark that the improvement of AutoDisenSeq-LLM over baselines mainly comes from the incorporation of LLM, which obtains rich prior knowledge through pretraining on large-scale data.

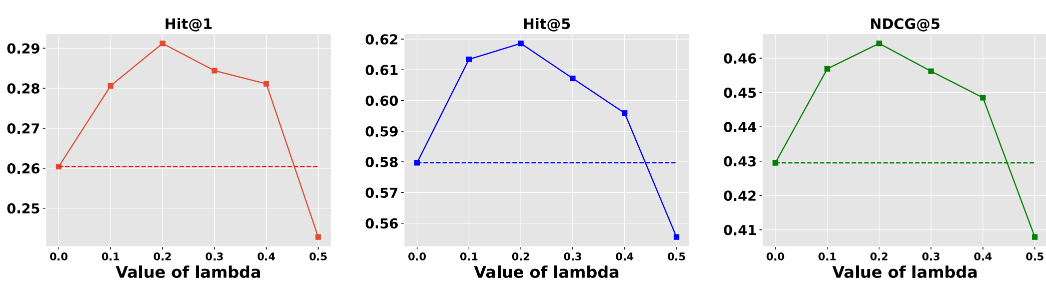

Fig. 6. Analysis of the threshold parameter λ which determines whether the input and the label sequence share same intention under this latent category. In particular, $\lambda = 0$ is equivalent to not using *S2S* training.

complicated modules and uses NAS to automatically search for the optimum structure, it is natural that our method will cost more time in training, which is necessary as a tradeoff for performance and it saves the time of manually searching for the best parameters as well. From the results, we can also observe that the extra time consumed compared to SASRec mainly comes from the disentangled part in Seq2Seq training since the time gap between AutoDisenSeq-LLM and DSSRec is much shorter than that between DSSRec and SASRec. This demonstrates the relative efficiency of our proposed automated attention representation search, especially the necessity of the two constraints which reduce the time complexity greatly.

5.4.6 Sensitivity Analysis. In Section 4.1.3, a hyper-parameter λ is introduced to control the confidence threshold that the input sequence and label sequence share the same intent under a specific latent category k. This parameter is critical to our *S2S* training, since $\lambda = 0$ implies that our model is using *S2I* training only, while large λ value tends to allow more *S2S* samples.

We conduct experiments on dataset ML-1m with different values of λ and report the results in Figure 6. We observe a performance drop when setting the threshold too strict, i.e., $\lambda = 0$, or too loose, i.e., $\lambda = 0.5$. A strict threshold will exclude too many *S2S* samples, reducing the effectiveness of *S2S* training, while a loose threshold will include some irrelevant *S2S* samples, deteriorating the performance as well. Similar results hold on other three datasets, although the optimum value of λ may vary.

6 Conclusion and Future Work

In this article, we first propose an AutoDisenSeq model, which is able to discover the optimal self-attention representations capable of capturing sequential patterns in sequential recommendation through NAS while keeping intention disentangled in latent space. The proposed AutoDisenSeq model is able to provide us with accurate architecture performance estimations and great flexibility for disentanglement of latent intention representation. Furthermore, we propose AutoDisenSeq-LLM, an LLM-guided model for sequential recommendation, where the domain-specific knowledge learned by automated disentangled Transformer-based sequential recommendation model and the general knowledge carried in the LLM are combined to provide more precise recommendation for the users. Extensive experiments on four real-world datasets are conducted to show that our proposed AutoDisenSeq and AutoDisenSeq-LLM models both beat existing baseline methods in overall recommendation and cold-start recommendation scenarios. To conclude, we note that the proposed automated disentangled sequential modeling mechanism for the Transformer-based architecture can help to adapt to various scenarios and datasets, and the LLM as an auxiliary enhancer can further improve the recommendation accuracy with its prior knowledge.

This work can be regarded as a prior work in collaborating the domain-specific model and the LLM in recommendation. Future works such as finetuning the LLM and the automated disentangled sequence latent representation together in the training stage will be interesting and promising direction that deserves further investigations.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*. 1–11.

[2] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. arXiv:2305.00447. Retrieved from https://arxiv.org/abs/2305.00447

[3] 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.

[4] Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. 2018. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 2095–2102.

[5] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. 2018. Understanding disentangling in $beta$-VAE. arXiv:1804.03599. Retrieved from https://arxiv.org/abs/1804.03599

[6] Han Cai, Ligeng Zhu, and Song Han. 2019. ProxylessNAS: Direct neural architecture search on target task and hardware. In *Proceedings of the International Conference on Learning Representations*.

[7] Jie Cai, Xin Wang, Chaoyu Guan, Yateng Tang, Jin Xu, Bin Zhong, and Wenwu Zhu. 2022. Multimodal continual graph learning with neural architecture search. In *Proceedings of the ACM Web Conference 2022*. 1292–1300.

[8] Jie Cai, Xin Wang, Haoyang Li, Ziwei Zhang, and Wenwu Zhu. 2024. Multimodal graph neural architecture search under distribution shifts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8227–8235.

[9] Jianlong Chang, Yiwen Guo, gaofeng Meng, Shiming Xiang, Chunhong Pan. 2019. DATA: Differentiable ArchiTecture approximation. In *Proceedings of the 33rd Conference on in Neural Information Processing Systems (NeurIPS '19)*.

[10] Hong Chen, Yudong Chen, Xin Wang, Ruobing Xie, Rui Wang, Feng Xia, and Wenwu Zhu. 2021. Curriculum disentangled recommendation with noisy multi-feedback. In *Proceedings of the 35th International Conference on Neural Information Processing Systems.* 26924–26936.

[11] Hong Chen, Bin Huang, Xin Wang, Yuwei Zhou, and Wenwu Zhu. 2023a. Global-local GraphFormer: Towards better understanding of user intentions in sequential recommendation. In *Proceedings of the 5th ACM International Conference on Multimedia in Asia*. 1–7.

[12] Hong Chen, Yipeng Zhang, Xin Wang, Xuguang Duan, Yuwei Zhou, and Wenwu Zhu. 2024. DisenDreamer: Subject-driven text-to-image generation with sample-aware disentangled tuning. *IEEE Transactions on Circuits and Systems for Video Technology* (2024), Vol. 14. DOI: https://doi.org/10.1109/TCSVT.2024.3369757

[13] Hong Chen, Yipeng Zhang, Simin Wu, Xin Wang, Xuguang Duan, Yuwei Zhou, and Wenwu Zhu. 2023b. Disenbooth: Identity-preserving disentangled tuning for subject-driven text-to-image generation. In *Proceedings of the 12th International Conference on Learning Representations*. 1–12.

[14] Tian Qi Chen, Xuechen Li, Roger B. Grosse, and David K. Duvenaud. 2018a. Isolating sources of disentanglement in variational autoencoders. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems.* 2610–2620.

[15] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems.* 1–10.

[16] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018b. Sequential recommendation with user memory networks. In *Proceedings of WSDM 2018*. 108–116.

[17] 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.

[18] Zhongxia Chen, Xiting Wang, Xing Xie, Tong Wu, Guoqing Bu, Yining Wang, and Enhong Chen. 2019. Co-attentive multi-task learning for explainable recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 2137–2143.

[19] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. arXiv:1904.10509. Retrieved from http://arxiv.org/abs/1904.10509

[20] Heeyoul Choi, Kyunghyun Cho, and Yoshua Bengio. 2018. Fine-grained attention mechanism for neural machine translation. *Neurocomputing* 284 (2018), 171–176.

[21] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. 2019. FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search. arXiv:1907.01845. Retrieved from https://arxiv.org/abs/1907.01845

[22] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems.* ACM, 191–198.

[23] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. M6-rec: Generative pretrained language models are open-ended recommender systems. arXiv:2205.08084. Retrieved from https://arxiv.org/abs/2205.08084

[24] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*. 2978–2988.

[25] Michal Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. 2017. Frustratingly short attention spans in neural language modeling. In *Proceedings of the 5th International Conference on Learning Representations*.

[26] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*. 933–941.

[27] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM TOIS* 22, 1 (2004), 143–177.

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805. Retrieved from https://arxiv.org/abs/1810.04805

[29] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. 2016. Deep unsupervised clustering with Gaussian mixture variational autoencoders. arXiv:1611.02648. Retrieved from https://arxiv.org/abs/1611.02648

[30] Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

[31] Emilien Dupont. 2018. Learning disentangled joint continuous and discrete representations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems.* 710–720.

[32] S. M. Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S. Morcos, Marta Garnelo, Avraham Ruderman, Andrei A. Rusu, Ivo Danihelka, Karol Gregor, David P. Reichert, Lars Buesing, Theophane Weber, Oriol Vinyals, Dan Rosenbaum, Neil Rabinowitz, Helen King, Chloe Hillier, Matt Botvinick, Daan Wierstra, Koray Kavukcuoglu, and Demis Hassabis. 2018. Neural scene representation and rendering. *Science* 360, 6394 (2018), 1204–1210.

[33] Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Jiliang Tang, and Qing Li. 2023. Recommender systems in the era of large language models (LLMs). arXiv:2307.02046. Retrieved from https://arxiv.org/abs/2307.02046

[34] 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.

[35] Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. 2019. Densely connected search space for more flexible neural architecture search. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10628–10637.

[36] Luke Friedman, Sameer Ahuja, David Allen, Terry Tan, Hakim Sidahmed, Changbo Long, Jun Xie, Gabriel Schubiner, Ajay Patel, Harsh Lara, Brian Chu, Zexi Chen, and Manoj Tiwari. 2023. Leveraging large language models in conversational recommender systems. arXiv:2305.07961. Retrieved from https://arxiv.org/abs/2305.07961

[37] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2020. Graph neural architecture search. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*.

[38] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*. 299–315.

[39] Chaoyu Guan, Xin Wang, Hong Chen, Ziwei Zhang, and Wenwu Zhu. 2022. Large-scale graph neural architecture search. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 7968–7981.

[40] Chaoyu Guan, Xin Wang, and Wenwu Zhu. 2021. Autoattend: Automated attention representation search. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 3864–3874.

[41] Lei Guo, Chunxiao Wang, Xinhua Wang, Lei Zhu, and Hongzhi Yin. 2023. Automated prompting for non-overlapping cross-domain sequential recommendation. arXiv:2304.04218. Retrieved from https://arxiv.org/abs/2304.04218

[42] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. 2020a. Single path one-shot neural architecture search with uniform sampling. In *Proceedings of the 16th European Conference on Computer Vision*.

[43] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. 2020b. Single path one-shot neural architecture search with uniform sampling. In *Proceedings of the 16th European Conference on Computer Vision (ECCV '20)*. Springer, 544–560.

[44] Haoyang Li, Xin Wang, Zeyang Zhang, Haibo Chen, Ziwei Zhang, Wenwu Zhu. 2024. Disentangled graph self-supervised learning for out-of-distribution generalization. In *Proceedings of the 41st International Conference on Machine Learning*. PMLR. 1–13.

[45] F. Maxwell Harper and Joseph A. Konstan. 2015. The Movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TIIS)* 5, 4 (2015), 1–19.

[46] Ruining He, Chen Fang, Zhaowen Wang, and Julian McAuley. 2016. Vista: A visually, socially, and temporally-aware model for artistic recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 309–316.

[47] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017a. Translation-based recommendation. In *Proceedings of ACM RecSys 2017*. 161–169.

[48] Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017b. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 388–397.

[49] Ruining He and Julian McAuley. 2016. Fusing similarity models with Markov chains for sparse sequential recommendation. In *Proceedings of the 16th International Conference on Data Mining (ICDM)*. IEEE, 191–200.

[50] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017c. Neural collaborative filtering. In *Proceedings of WWW 2017*. 173–182.

[51] 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.

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

[53] 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 *Proceedings of the International Conference on Learning Representations*, Vol. 3. 1–11.

[54] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F. Fei-Fei, and Juan Carlos Niebles. 2018. Learning to decompose and disentangle representations for video prediction. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 517–526.

[55] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. arXiv:2106.09685. Retrieved from https://arxiv.org/abs/2106.09685
[56] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*. IEEE, 263–272.
[57] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 505–514.
[58] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. 2017. Variational deep embedding: an unsupervised and generative approach to clustering. In *Proceedings of IJCAI 2017*. 1965–1972.
[59] Wang-Cheng Kang and Julian McAuley. 2018a. Self-attentive sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 197–206.
[60] Wang-Cheng Kang and Julian McAuley. 2018b. Self-attentive sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.
[61] Hyunjik Kim and Andriy Mnih. 2018. Disentangling by factorising. In *Proceedings of the 35th International Conference on Machine Learning*. 2654–2663.
[62] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations*. 1–11.
[63] Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. arXiv:1312.6114. Retrieved from https://arxiv.org/abs/1312.6114
[64] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *Proceedings of the International Conference on Learning Representations*. 1–11.
[65] Nikos Komodakis and Spyros Gidaris. 2018. Unsupervised representation learning by predicting image rotations. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 1–14.
[66] Yehuda Koren, Robert Bell, Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[67] Xin Wang, Hong Chen, Si'ao Tang, Zihao Wu, and Wenwu Zhu. 2024. Disentangled Representation Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1–20.
[68] Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. 2018. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 8606–8616.
[69] Yoonhyung Lee, Seunghyun Yoon, and Kyomin Jung. 2020. Multimodal speech emotion recognition using cross attention with aligned audio and text. In *Proceedings of the 21st Annual Conference of the International Speech Communication Association*. 2717–2721.
[70] 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 *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2615–2623.
[71] Haoyang Li, Xin Wang, Ziwei Zhang, Jianxin Ma, Peng Cui, and Wenwu Zhu. 2021a. Intention-aware sequential recommendation with structured intent transition. *IEEE Transactions on Knowledge and Data Engineering* 34, 11 (2021), 5403–5414.
[72] Haoyang Li, Xin Wang, Ziwei Zhang, Zehuan Yuan, Hang Li, and Wenwu Zhu. 2021b. Disentangled contrastive learning on graphs. In *Proceedings of the 35th Conference on Neural Information Processing Systems*. 21872–21884.
[73] Haoyang Li, Ziwei Zhang, Xin Wang, and Wenwu Zhu. 2022. Disentangled graph contrastive learning with independence promotion. *IEEE Transactions on Knowledge and Data Engineering* 35, 8 (2022), 7856–7869.
[74] 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.
[75] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. 2018. Massively parallel hyperparameter tuning. arXiv:1810.05934. Retrieved from https://arxiv.org/abs/1810.05934
[76] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 305–314.
[77] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of 2018 World Wide Web Conference*. 689–698.
[78] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. 2018. Tune: A research platform for distributed model selection and training. arXiv:1807.05118. Retrieved from https://arxiv.org/abs/1807.05118

[79] Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of the 5th International Conference on Learning Representations*.

[80] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L. Yuille, and Li Fei-Fei. 2019a. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 82–92.

[81] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019b. DARTS: Differentiable architecture search. In *Proceedings of the 7th International Conference on Learning Representations* (2019). 1–11.

[82] Junling Liu, Chao Liu, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is ChatGPT a good recommender? A preliminary study. arXiv:2304.10149. Retrieved from https://arxiv.org/abs/2304.10149

[83] Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu. 2019c. Is a single vector enough? Exploring node polysemy for network embedding. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[84] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-aware sequential recommendation. In *Proceedings of the 16th International Conference on Data Mining (ICDM)*. IEEE, 1053–1058.

[85] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1831–1839.

[86] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Proceedings of the Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*. 289–297.

[87] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1412–1421.

[88] Benteng Ma, Jing Zhang, Yong Xia, and Dacheng Tao. 2020a. Auto learning attention. In *Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems*. 1488–1500.

[89] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019a. Disentangled graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*. 4212–4221.

[90] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019d. Learning disentangled representations for recommendation. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 5712–5723.

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

[92] Liqian Ma, Qianru Sun, Stamatios Georgoulis, Luc Van Gool, Bernt Schiele, and Mario Fritz. 2018. Disentangled person image generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 99–108.

[93] Qianli Ma, Liuhong Yu, Shuai Tian, Enhuan Chen, and Wing W. Y. Ng. 2019b. Global-local mutual attention model for text classification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27, 12 (2019), 2127–2139.

[94] Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. 2019c. A tensorized transformer for language modeling. In *Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*. 2232–2242.

[95] Jieru Mei, Yingwei Li, Xiaochen Lian, Xiaojie Jin, Linjie Yang, Alan Yuille, and Jianchao Yang. 2020. AtomNAS: Fine-grained end-to-end neural architecture search. In *Proceedings of the International Conference on Learning Representations*. 1–13.

[96] Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one?. In *Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*. 1–11.

[97] Federico Monti, Michael Bronstein, and Xavier Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*. 3700–3710.

[98] Duy-Kien Nguyen and Takayuki Okatani. 2018. Improved fusion of visual and language representations by dense symmetric co-attention for visual question answering. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition*. 6087–6096.

[99] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 188–197.

[100] Guocheng Niu, Hengru Xu, Bolei He, Xinyan Xiao, Hua Wu, and Sheng Gao. 2019. Enhancing local feature extraction with global representation for neural text classification. In *Proceedings of the 2019 Conference on Empirical Methods*

in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 496–506.

[101] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. 2018. Efficient neural architecture search via parameter sharing. *Proceedings of the 35th International Conference on Machine Learning* (2018).

[102] Yijian Qin, Xin Wang, Peng Cui, and Wenwu Zhu. 2021. Gqnas: Graph q network for neural architecture search. In *Proceedings of the 2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1288–1293.

[103] Yijian Qin, Xin Wang, Ziwei Zhang, Hong Chen, and Wenwu Zhu. 2024. Multi-task graph neural architecture search with task-aware collaboration and curriculum. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*.

[104] Yijian Qin, Xin Wang, Ziwei Zhang, Pengtao Xie, and Wenwu Zhu. 2022a. Graph neural architecture search under distribution shifts. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 18083–18095.

[105] Yijian Qin, Ziwei Zhang, Xin Wang, Zeyang Zhang, and Wenwu Zhu. 2022b. Nas-bench-graph: Benchmarking graph neural architecture search. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*. 54–69.

[106] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *Proceedings of the International Conference on Learning Representations*. 1–14.

[107] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. 452–461.

[108] 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.

[109] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*. ACM, 175–186.

[110] C. Freudenthaler S. Rendle and L. Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. 811–820.

[111] Tonmoy Saikia, Yassine Marrakchi, Arber Zela, Frank Hutter, and Thomas Brox. 2019. AutoDispNet: Improving disparity estimation with AutoML. In *Proceedings of the IEEE International Conference on Computer Vision*. 1812–1823.

[112] Ruslan Salakhutdinov and Andriy Mnih. 2011. Probabilistic matrix factorization. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 20. 1–8.

[113] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*. ACM, 285–295.

[114] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

[115] David R. So, Chen Liang, and Quoc V. Le. 2019. The evolved transformer. In *Proceedings of the 36th International Conference on Machine Learning*.

[116] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Proceedings of the Advances in Neural Information Processing Systems*.

[117] 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*.

[118] Y. Liu, V. S. Sheng, J. Xu, D. Wang, G. Liu, T. Zhang, P. Zhao, and X. Zhou. 2019. Feature-level deeper self-attention network for sequential recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 4320–4326.

[119] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 565–573.

[120] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020. Sparse sinkhorn attention. In *Proceedings of the 37th International Conference on Machine Learning*.

[121] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire

Cui, Marian Croak, Ed Chi, and Quoc Le. 2022. LaMDA: Language models for dialog applications. arXiv:2201.08239. Retrieved from https://arxiv.org/abs/2201.08239

[122] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and efficient foundation language models. arXiv:2302.13971. Retrieved from https://arxiv.org/abs/2302.13971

[123] 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 *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6000–6010.

[124] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proceedings of the 6th International Conference on Learning Representations*. 1–12.

[125] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015b. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1235–1244.

[126] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015a. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[127] Xin Wang, Hong Chen, Si'ao Tang, Zihao Wu, and Wenwu Zhu. 2022a. Disentangled representation learning. arXiv:2211.11695. Retrieved from https://arxiv.org/abs/2211.11695

[128] Xin Wang, Hong Chen, Yuwei Zhou, Jianxin Ma, and Wenwu Zhu. 2022b. Disentangled Representation Learning for Recommendation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 1 (2022), 408–424.

[129] Xin Wang, Hong Chen, and Wenwu Zhu. 2021. Multimodal disentangled representation for recommendation. In *Proceedings of the 2021 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1–6.

[130] Xin Wang, Steven CH Hoi, Chenghao Liu, and Martin Ester. 2017. Interactive social recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 357–366.

[131] Xin Wang, Zirui Pan, Yuwei Zhou, Hong Chen, Chendi Ge, and Wenwu Zhu. 2023a. Curriculum co-disentangled representation learning across multiple environments for social recommendation. In *Proceedings of the 40th International Conference on Machine Learning*. PMLR, 36174–36192.

[132] Xin Wang, Zihao Wu, Hong Chen, Xiaohan Lan, and Wenwu Zhu. 2023b. Mixup-augmented temporally debiased video grounding with content-location disentanglement. In *Proceedings of the 31st ACM International Conference on Multimedia*. 4450–4459.

[133] Xiaofang Wang, Xuehan Xiong, Maxim Neumann, A. J. Piergiovanni, Michael S. Ryoo, Anelia Angelova, Kris M. Kitani, and Wei Hua. 2020a. AttentionNAS: Spatiotemporal attention cell search for video classification. In *Proceedings of the 16th European Conference on Computer Vision*.

[134] Yujing Wang, Yaming Yang, Yiren Chen, Jing Bai, Ce Zhang, Guinan Su, Xiaoyu Kou, Yunhai Tong, Mao Yang, and Lidong Zhou. 2020b. TextNAS: A Neural Architecture Search Space Tailored for Text Representation. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence. The 32nd Innovative Applications of Artificial Intelligence Conference*.

[135] Chuhan Wu, Fangzhao Wu, Tao Qi, Jianxun Lian, Yongfeng Huang, and Xing Xie. 2020. PTUM: Pre-training user model from unlabeled user behaviors via self-supervision. arXiv:2010.01494. Retrieved from https://arxiv.org/abs/2010.01494

[136] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. 495–503.

[137] Zihao Wu, Xin Wang, Hong Chen, Kaidong Li, Yi Han, Lifeng Sun, and Wenwu Zhu. 2023. Diff4Rec: Sequential recommendation with curriculum-scheduled diffusion augmentation. In *Proceedings of the 31st ACM International Conference on Multimedia*. 9329–9335.

[138] Beini Xie, Heng Chang, Ziwei Zhang, Xin Wang, Daixin Wang, Zhiqiang Zhang, Rex Ying, and Wenwu Zhu. 2023. Adversarially robust neural architecture search for graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8143–8152.

[139] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. 2019. SNAS: Stochastic neural architecture search. In *Proceedings of the International Conference on Learning Representations*.

[140] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[141] Yang Yao, Xin Wang, Yijian Qin, Ziwei Zhang, Wenwu Zhu, and Hong Mei. 2024. Data-augmented curriculum graph neural architecture search under distribution shifts. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 15 (2024), 16433–16441.

[142] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[143] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[144] Zhou Yu, Yuhao Cui, Jun Yu, Meng Wang, Dacheng Tao, and Qi Tian. 2020. Deep multimodal neural architecture search. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20)*.

[145] Zhou Yu, Jun Yu, Yuhao Cui, Dacheng Tao, and Qi Tian. 2019. Deep modular co-attention networks for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[146] Zeyang Zhang, Xin Wang, Yijian Qin, Hong Chen, Ziwei Zhang, Xu Chu, Wenwu Zhu. 2024. Disentangled continual graph neural architecture search with invariant modular supernet. In *Proceedings of the International Conference on Machine Learning*. PMLR. 1–17.

[147] Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023b. Recommendation as instruction following: A large language model empowered recommendation approach. arXiv:2305.07001. Retrieved from https://arxiv.org/abs/2305.07001

[148] Yipeng Zhang, Xin Wang, Hong Chen, and Wenwu Zhu. 2023a. Adaptive disentangled transformer for sequential recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3434–3445.

[149] Zeyang Zhang, Xin Wang, Ziwei Zhang, Guangyao Shen, Shiqi Shen, and Wenwu Zhu. 2024. Unsupervised graph neural architecture search with disentangled self-supervision. *Proceedings of the 37th International Conference on Neural Information Processing Systems.*

[150] Zeyang Zhang, Ziwei Zhang, Xin Wang, Yijian Qin, Zhou Qin, and Wenwu Zhu. 2023c. Dynamic heterogeneous graph attention neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11307–11315.

[151] 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 *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI '18)*.

[152] Kaixiong Zhou, Qingquan Song, Xiao Huang, and Xia Hu. 2019. Auto-GNN: Neural architecture search of graph neural networks. arXiv:1909.03184. Retrieved from https://arxiv.org/abs/1909.03184

[153] Yuwei Zhou, Xin Wang, Hong Chen, Xuguang Duan, Chaoyu Guan, and Wenwu Zhu. 2022. Curriculum-nas: Curriculum weight-sharing neural architecture search. In *Proceedings of the 30th ACM International Conference on Multimedia*. 6792–6801.

[154] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Josh Tenenbaum, and Bill Freeman. 2018. Visual object networks: Image generation with disentangled 3D representations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems.* 118–129.

[155] Barret Zoph and Quoc V. Le. 2017. Neural architecture search with reinforcement learning. In *Proceedings of the 5th International Conference on Learning Representations*.

[156] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition.*

Received 18 February 2024; revised 23 April 2024; accepted 4 June 2024