Compressing Long Range User Behavior Sequence with Interest Experts

Anonymous submission

Abstract

State-of-the-art sequential recommendation models heavily rely on transformer's attention mechanism. However, the quadratic computational and memory complexities of self attention have limited its scalability for modeling users' long range behaviour sequences. To address this problem, in this paper we propose a novel efficient sequential recommendation framework PKM-ACT that leverages the novel product key technique for sparsely retrieving compressed user's interests from a vast pool of interests which significantly increases the capacity for modelling user interest with a negligible computational overhead. Moreover, in order to optimize the time complexity when modeling long range sequence, we propose a novel dispatcher module for aggregating and dispatching the dependencies among behviour sequences. The proposed dispatcher mechanism significantly reduces the quadratic complexity and makes the model feasible for adequately modelling extremely long sequences. To validate the effectiveness of our proposed PKM-ACT, we conduct extensive experiments on various public datasets and compare it with several strong sequential recommenders. Experimental results demonstrate that PKM-ACT consistently outperforms baselines by a significant margin and also highlight the computational efficiency of PKM-ACT when modelling long sequences. We will make our implementation code publicly available.

Introduction

Sequential recommender systems (SRs) have play a vital role in online content sharing platforms and e-commerce, such as Kuaishou (Chang et al. 2023; Si et al. 2024) and Taobao (Zhou et al. 2018, 2019; Pi et al. 2020), which capture user's actual preferences from the long-term action history (e.g. click, view and comment) to predict next action. In practical recommendation scenarios, hundreds of millions of active users leave tens of billions of user interaction logs each day. So accurately modeling the extensive length of user behavior sequence across the life cycle not only helps SRs learn better representations for revealing users' actual preferences, but also boosts the user experience and business effectiveness on such platforms.

Recently, many efforts (Kang and McAuley 2018; Liu et al. 2023; Sun et al. 2019; Zhang et al. 2019) have been devoted to leveraging transformer-based architectures (Vaswani et al. 2017), which is known as self-attention recommenders (SARs), for modeling users historical behavior sequences. The core of SARs adopts the key mechanism that captures contextual information from the entire user behaviour sequence by modeling pairwise item-to-item interactions between the inputs at every tokens. For example, BERT4Rec (Sun et al. 2019) leverages bidirectional encoding through multi-head self attention and captures the context from both directions in user history for enhanced sequential recommendation. SASRec (Kang and McAuley 2018) adopts self-attention mechanisms to sequential recommendation by balancing long-term and short-term user preferences. Despite their superior performance and parallel training efficiency, SARs have encountered critical efficiency issues when scaling up to long-term sequences (Pancha et al. 2022; Yue et al. 2024) because dot-product operation in the attention layer result quadratic scaling complexity $\mathcal{O}(N^2)$ with respect to the sequence length N. Aiming to solve this limitation, recent works adopt linear attention mechanism (Liu et al. 2023) to estimate original dot-product attention or leverage linear recurrent units (Orvieto et al. 2023; Yue et al. 2024) to address the dilemma of training and inference efficiency. However, these methods may sacrifice recommendation accuracy and stability when compared with SARs methods with regular attention.

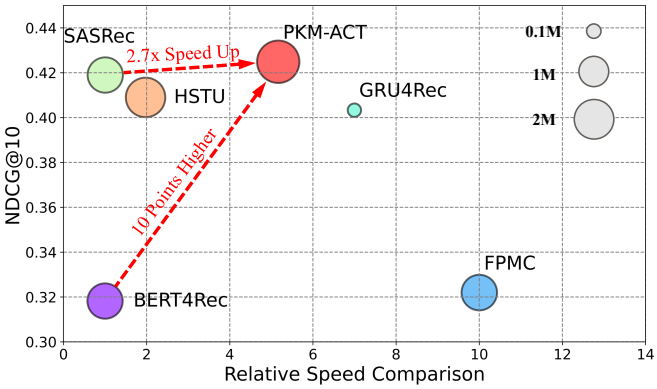

Figure 1: Trade-off between NDCG@10 (y-axis), inference speed (x-axis) and model parameter (cir-radius) on ML-1M.

In this paper, we propose an efficient **P**roduct **K**ey **M**emory-based interest retrieval framework with **AC**tion **T**ransformer (PKM-ACT) to address the issue of high com-

plexity of standard attention. Specifically, we first adopt product key retrieval (Lample et al. 2019) for sparsely retrieving compressed user's interests representations from an large number of interest experts pool which significantly increases the parameter capacity with a negligible computational overhead. In practical recommendation scenarios, the interest experts pool can be initialized at a scale of millions to effectively model the personalized interests of hundreds of millions of users. This large-scale interest pool allows for a more granular representation of user preferences and enable the SRs to capture a wide spectrum of interest patterns across users more effectively. Then, action transformer (ACT) interacts the compressed user's interests representation with the original user behavior sequence with a novel dispatcher mechanism to reduce complexity from quadratic to linear. Compared with traditional attention mechanism, action transformer take the compressed user interests with a fixed length as query input which allows ACT to perform attention operation linearly while also storing adequate contextual information from vast interest pool. We perform extensive experiments on multiple public datasets and compare PKM-ACT to a variety of strong SRs baseline models. As shown in Figure 1, PKM-ACT achieves competitive and even better performance, while acquiring significant gains of computational efficiency compared with traditional SARs methods.

We summarize the contributions of our work as follows:

- We introduce a novel product key memory retrieval mechanism (PKM) for sparsely retrieving compressed user's interests representations from a large interest experts pool with a negligible computational overhead.
- We propose an efficient action transformer (ACT) with a novel dispatcher mechanism to address the dilemma of training efficiency, inference efficiency and recommendation performance.
- Extensive experiments demonstrate the effectiveness and efficiency of PKM-ACT in comparison to state-of-the-art methods on multiple public datasets, where PKM-ACT consistently outperforms baseline methods by a large margin.

Related Work

Transformer-based SRs

In recent years, the Transformer architecture (Vaswani et al. 2017; Hou et al. 2022; Kang and McAuley 2018; Sun et al. 2019; Wu et al. 2020; Zhang et al. 2019) has significantly advanced the field of sequential recommendation systems by leveraging its superior ability to model long-range dependencies and capture user-item interaction patterns effectively. The key component of transformer is the self attention mechanism which computes the corresponding attention matrix for distinguishing items' importance by a dot-product operation between the query and key matrices. Specifically, ATTRec (Zhang et al. 2018) leverages the self-attention mechanism to capture both short-term item-item interactions and long-term user-item relationships. SASRec (Kang and McAuley 2018) employs multi-head self-attention mechanisms to effectively capture complex and dynamic user preferences, allowing for more accurate modeling of user behavior over time. BERT4Rec (Sun et al. 2019) adapts the bidirectional self-attention mechanism from Transformers, enabling a more comprehensive understanding of user preferences by capturing transition patterns from both left and right contexts in the sequence, overcoming limitations of unidirectional models. FDSA (Zhang et al. 2019) incorporates feature-level self-attention mechanisms to generate accurate and context-aware sequential representation. While these methods have proved to be effective, they primarily rely on traditional dot-product attention mechanisms which can lead to computational inefficiencies when scaling up to long-term user behviour sequence.

Efficient Transformers

Recently, various "*X*-former" models have been proposed for improving the computational and memory efficiency of original Transformer architecture. The earliest attempts design sparse attention score matrix by limiting the receptive field to fixed patterns such as local attention (Parmar et al. 2018; Qiu et al. 2019) and strided attention (Beltagy, Peters, and Cohan 2020; Child et al. 2019). Another line of research designs the learnable patterns in a data-driven manner. For example, Reformer (Kitaev, Kaiser, and Levskaya 2020) designs the reversible residual layers and locality-sensitive hashing (Andoni et al. 2015) attention, decreasing computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(NlogN)$. And Routing Transformer (Roy et al. 2021) introduces an efficient content-based sparse attention mechanism using online *k*-means clustering. Subsequently, low-rank approximations methods emerged as a solution by assuming low-rank structure of original attention matrix. Linformer (Wang et al. 2020) projects the keys and values to a low-rank dimensions using the additional projection layers. Linear Transformers (Katharopoulos et al. 2020) rewrite dot product attention as a linear dot product of kernel feature maps, enabling recurrent computation, and faster autoregressive inference. Besides, recent works implement linear attention mechanism (Liu et al. 2023) to estimate original dot-product attention or leverage linear recurrent units (Yue et al. 2024) to address the dilemma of training and inference efficiency in long-term sequential recommendation scenarios. Despite existing methods are proved to be efficient in computational complexity, they may sacrifice recommendation accuracy and stability when compared with SARs methods with regular attention. In contrast, our proposed PKM-ACT decouples computational cost from model's parameter capacity which improves recommendation performance with negligible extra computational overhead.

Method

This section introduces our proposed novel PKM-ACT framework for sequential recommendation systems. We begin by defining the problem formulation for sequential recommendation tasks. Next, we present proposed product key memory retrieval mechanism (PKM) for sparsely retrieving compressed user's interests representation. Then, we intro-

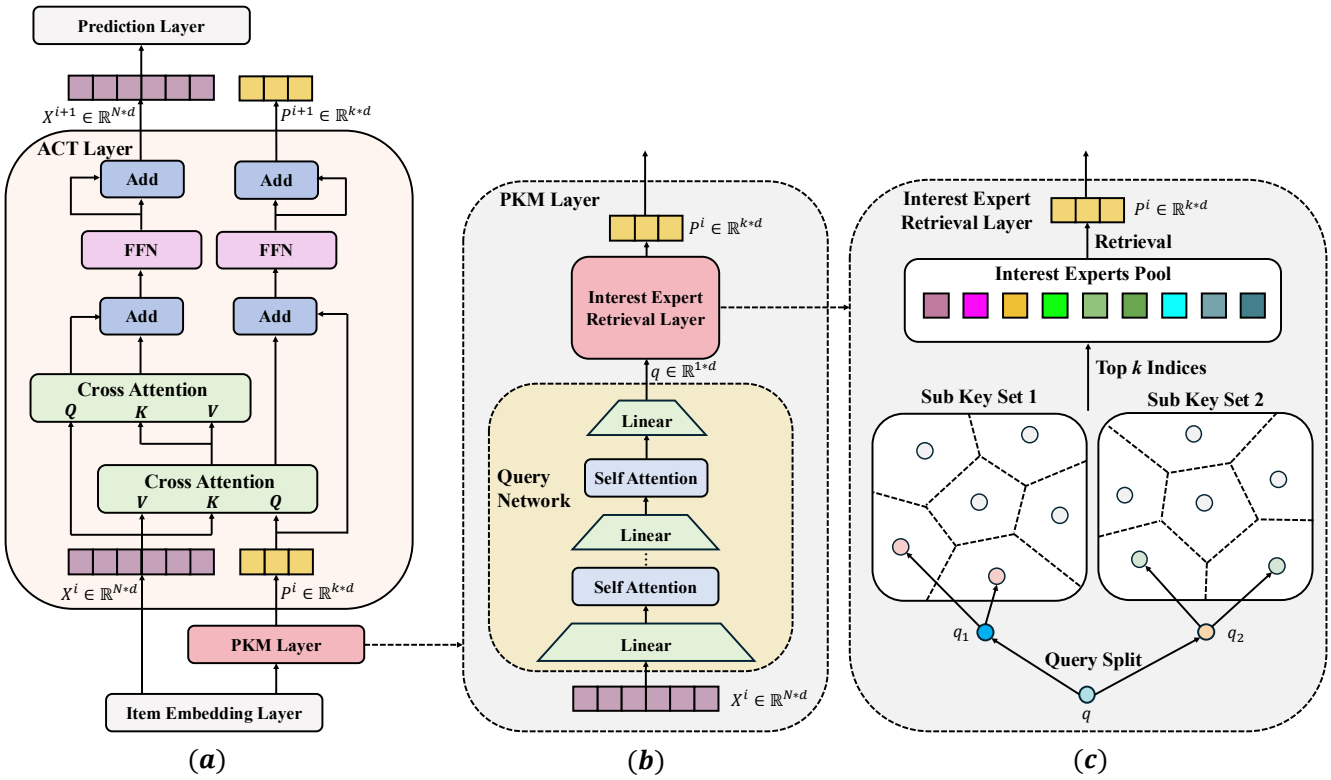

Figure 2: Framework of proposed PKM-ACT. The core part of this framework is the ACT layer and PKM layer. ACT layer includes two aggregating and dispatching cross attention mechanisms. PKM layer consists of hierarchical query network and interest experts retrival layer.

duce the efficient action transformer (ACT) with a novel dispatcher mechanism. We will discuss the optimization process of the model and present the pseudo-code.

Problem Statement

Given a set of user $\mathcal{U} = \{u_1, u_2, \cdots, u_{|\mathcal{U}|}\}$ and a set of item $\mathcal{X} = \{x_1, x_2, \cdots, x_{|\mathcal{X}|}\}$, we represent u_i's the historical item interaction list as $X = \{x_1^i, \cdots, x_t^i, \cdots, x_N^i\} \in \mathbb{R}^{N \times d}$, including the item's ID embeddings and positional embeddings with the dimension of d, where x_t^i is the t-th item interacted by user u_i and N represents the maximum historical window length. Sequential recommendation systems aim to predict a user's next item selection based on their interaction history. Our proposed PKM-ACT takes the user's past N interactions as input and predict the most possible items for the $N+1$ time step based on the generated a list of top-k items from the available set $\mathcal{X}$.

Product Key Memory Layer

In this section we introduce the Product Key Memory (PKM) layer which adopts product keys (Lample et al. 2019) technique retrieve compressed user interest representation from a pool of interest expert. The overall structure of the PKM layer is presented in Figure 2 (b) and (c). The PKM layers is composed of two components: a hierarchical query network and interest expert retrieval layer.

Query Network: The hierarchical query network $Q : X \mapsto Q(X) = q \in \mathbb{R}^{1 \times d}$ takes the user historical behviour sequences $X \in \mathbb{R}^{N \times d}$ as input and applies the hierarchical linear pooling along the sequence length dimension with self attention:

$$\begin{aligned} X &\leftarrow Reshape(X, (\frac{N}{stride}, stride \cdot d)) \\ X &\leftarrow LinearProjection(X) \\ X &\leftarrow SelfAttention(X) \end{aligned} \tag{1}$$

where $stride$ represents the pool size and N is the sequence length dimension. This shorten process reduce the temporal redundancy of user's behviour sequence and significantly alleviate the computational costs of query network.

Interest Expert Retrieval Layer: Formally, let q be the query and $\mathcal{T}_k$ represents the top-k operation. We first initialize a set of interest experts $\mathcal{E} = \{e_1, \cdots, e_K\}$ and a corresponding set of K product keys $\mathcal{K} = \{k_1, \cdots, k_K\}$. Standard expert retrieval process first compute the inner products of query q and keys and outputs the indices of selected indices of k expert with highest scores. Therefore, the selected compressed k interest expert P is retrieved as follows:

$$\begin{aligned} \mathbb{I} &= \mathcal{T}_k(\{q^T k_i\}_{i=1}^K) \\ P &= \{e_i | i \in \mathbb{I}\} \end{aligned} \tag{2}$$

However, in practical scenario, if we intend to use a vast pool of interest experts, the naively retrieval process with Equation 2 is computational expensive. Hence, instead of

Algorithm 1: A Pytorch-style Pseudocode for PKM layers.

```
def pkm_layers(self, x, n_keys, dim, top_k)
    # initialize two sub_keys with half dimension and size n_keys
    # initialize one interest experts pool with complete dimension and size n_keys*n_keys
    self.sub_keys = nn.Parameter(2, n_keys, dim // 2)
    self.values = nn.Embedding(n_keys ** 2, dim)

    # project the input with hierarchical query network
    # compute the indices of the top matching interest experts using product keys
    query = self.query_network(x)
    scores, indices = self.get_indices(query, self.sub_keys, top_k)

    # gather the selected interest expert from values pool
    output = self.values(indices)

    return output
```

initializing K independent d-dimensional keys, we first create two independent sub-keys set $\mathcal{C}$ and $\mathcal{C}'$ with dimension of $d/2$ and cardinality of $\sqrt{K}$. Then, we can formulate the overall product keys set K with Cartesian product construction:

$$\mathcal{K} = \{(c, c')|c \in \mathcal{C}, c' \in \mathcal{C}'\} \tag{3}$$

So instead computing query q with all K keys in $\mathcal{K}$, we can split q into two sub queries q_1 and q_2 and compare the k most similar keys in sub-keys in $\mathcal{C}$ and $\mathcal{C}'$ independently:

$$\mathbb{I}_{\mathcal{C}} = \mathcal{T}_k(\{q_1^T c_i\}_{i=1}^{\sqrt{K}}), \;\; \mathbb{I}_{\mathcal{C}'} = \mathcal{T}_k(\{q_1^T c_i'\}_{i=1}^{\sqrt{K}}) \tag{4}$$

As explained in (Lample et al. 2019), the k most similar keys in $\mathcal{K}$ are mathematically guaranteed with the form of $\{(c_i, c_j')|i \in \mathbb{I}_{\mathcal{C}}, j \in \mathbb{I}_{\mathcal{C}'}\}$ and the complexity of top k interest expert retrieval in Equation 1 reduces from $\mathcal{O}(Kd)$ to $\mathcal{O}((\sqrt{K}+k^2)d)$. The pseudocode of the PKM layer forward pass is shown in Algorithm 1.

Action Transformer Layer

As a critical part of transformers, the idea of dot-product attention is to compute the pairwise item-to-item relationship at every input tokens. The standard self-attention mechanism directly applied to the user behviour sequences X^i in i-th layer can be defined as:

$$X^{i+1} = \text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d}})V \tag{5}$$

with the query matrix $Q = X^i W_Q \in \mathbb{R}^{N \times d}$, the key matrix $K = X^i W_K \in \mathbb{R}^{N \times d}$ and the value matrix $V = X^i W_V \in \mathbb{R}^{N \times d}$. The self attention helps the model to capture intra-dependencies among all historical tokens. However, the original self attention mechanism results in an attention map with the memory and complexity of $\mathcal{O}(N^2)$, which is very costly when the input sequences have a large number of N.

To address the potential complexity causing from a large number of tokens N, we introduce a dispatcher mechanism which aggregates and dispatch sequence token dependencies more efficiently which adopts the $k(k \ll N)$ user interest expert from the PKM layer as aggregation points. We first aggregate the information from all sequence tokens by using the dispatcher embeddings P^i from i-th layer as the query and the original sequence token embeddings as key and value:

$$\begin{aligned} P^{i+1} &= \text{Attention}(P^i W_{Q_1}, X^i W_{K_1}, X^i W_{V_1}) \\ &= \text{Softmax}(\frac{P^i W_{Q_1} (X^i W_{K_1})^T}{\sqrt{d}}) X^i W_{V_1} \end{aligned} \tag{6}$$

where the complexity is $\mathcal{O}(Nk)$. Then the dispatchers distribute the dependencies relationship to all sequence tokens by setting the original sequence token embedding as the query and the aggregated interest embeddings P^{i+1} as the key and value:

$$\begin{aligned} X^{i+1} &= \text{Attention}(X^i W_{Q_2}, P^{i+1} W_{K_2}, P^{i+1} W_{V_2}) \\ &= \text{Softmax}(\frac{X^i W_{Q_2} (P^{i+1} W_{K_2})^T}{\sqrt{d}}) P^{i+1} W_{V_2} \end{aligned} \tag{7}$$

where he complexity is also $\mathcal{O}(Nk)$. Therefore, the proposed dispatcher mechanism achieves a overall computational complexity of $\mathcal{O}(Nk)$, which is significantly more efficient than the $\mathcal{O}(N^2)$ complexity of direct self-attention on the original user historical behviour sequences. By utilizing this method, we can effectively capture complex interactions within the long-term behviour sequence without sacrificing processing speed and scalability.

In the action transformer layer, the outputted X^{i+1} and P^{i+1} is further passed to a feedforward layer with residual connections. After stacking several PKM and ACT layers, we get sequence representation $X^L \in \mathbb{R}^{N \times d}$ eventually.

Prediction and Model Optimization

After getting the item sequence representations $X^L \in \mathbb{R}^{N \times d}$, we proceed to make next-item prediction by calculating the probability distribution across the entire candidate item set to predict the most likely next item. At timestamp t,

Table 1: Dataset statistics.

Dataset	#users	#items	#actions	avg. length
ML-1M	6,040	3,416	1M	165.50
XLong	69,691	2,122,932	66.8M	958.8

we compute the recommendation score for every item embedding $x_i \in \mathbb{R}^d$ in the candidate pool:

$$s_i = X_t^L x_i^T \tag{8}$$

where X_t^L is the t-th item's representation in the behviour sequence. Therefore, the recommended probability distribution $\hat{y}_i$ of the next-item should be computed as follows:

$$\hat{y}_i = \frac{\exp(s_i)}{\sum_{x_j \in \mathcal{X}} \exp(s_i)} \tag{9}$$

Then, we formulate the sequential recommendation task as an binary classification problem which objective is to minimize the cross-entropy between the predicted recommendation results $\hat{y}$ and the ground truth label y:

$$\mathcal{L}(y, \hat{y}) = y\log(\hat{y}) + (1-y)(1-\log(\hat{y})) \tag{10}$$

Experiment

Experimental Settings

Datasets. We evaluate the effectiveness of our proposed ACT-PKM on two real-world datasets:

- **MovieLens**: A famous benchmark datasets for movie ratings and, we choose the widely used ML-1M (Harper and Konstan 2015) in this paper.
- **XLong**: This dataset is collected from Alibaba's online advertising platform which is known for long-term historical behaviour sequence for each user (Ren et al. 2019).

For preprocessing procedure, we follow the sequential recommendation settings in the previous studies (Kang and McAuley 2018; He et al. 2021; Yue et al. 2021; Wu et al. 2022; Raza and Ding 2022; Yuan et al. 2022) and exclude the users and items with fewer than 5 interaction. We group the user interaction history in chronological order and adopt the leave-one-out strategy on dataset splitting and use the most recent item as the test set, the second most recent item as the validation set, and the rest items in the sequences as the training set. For maximum sequence length, we set 256 for ML-1M, 1024 for XLong dataset. The statistics of these datasets after preprocessing are shown in Table 1.

Baseline Methods. We compare our PKM-ACT with several strong baseline methods, which include classic factorization and markov chain-based method FPMC, RNN-based method GRU4Rec and self-attention based and its various variant methods (e.g., SASRec, BERT4Rec, LinRec, HSTU):

- **FPMC**: A matrix factorization model which adopts Markov chains to capture user transition patterns (Rendle, Freudenthaler, and Schmidt-Thieme 2010).
- **GRU4Rec**: A classic recommender introduces GRU to user sequential behaviours which is originally designed for session-based recommendation (Hidasi and Karatzoglou 2018).
- **SASRec**: The first transformer-based sequential recommender employs unidirectional self-attention to capture user-item transition patterns (Kang and McAuley 2018).
- **BERT4Rec**: A bidirectional transformer encoder architecture for sequential recommendation to capture context from both left and right in behavior sequence (Sun et al. 2019).
- **LinRec**: A linear-attention based sequential recommender utilizes L2 norm to approximate softmax fitting. (Liu et al. 2023).
- **HSTU**: A novel generative recommendations baseline adopts hierarchical sequential transduction unit. We adopts the HSTU layers for sequential recommendation setting (Zhai et al. 2024).

Evaluation Metrics. In our evaluation process, we select the best-performing models based on their NDCG@10 scores on the validation set and then used to generate predictions on the test sets. We evaluate the models' performance using three metrics: NDCG@k, HR@k and MRR@k, where k is set to 10 and 20. To compute the final scores, we rank the predicted items against the entire candidate items in the dataset.

Implementation Details. Identical to previous studies (Hou et al. 2022; Kang and McAuley 2018), we train all models using cross entropy loss with Adam optimizer and use the default learning rate of 0.001 and default mini-batch size of 2,048. We choose the interest pool size $K = 16 * 16$ and number of selected interest experts $k = 8$ for ML-1M dataset while $K = 32 * 32$ and $k = 16$ for XLong dataset. During training, set the maximum epoch to be 500, validation is performed every epoch. Early stopping is triggered if validation NDCG@10 does not improve in 10 consecutive validation rounds. For the user whose interacted behavior sequence is less than maximum length, we pad the short-term sequences by zero to ensure the long-term sequential recommendation.

Overall Performance

Our main performance on ML-1M and XLong dataset are shown in Table 2 where each row represents the dataset and evaluation metric and each column represents different baselines. We mark the best results in bold and the second best results with underline. We also compute the relative improvement of PKM-ACT compared to the best-performing baseline method and all improvements are statistically significant by performing two-sided t-test with $p < 0.05$. The following observations can be drawn: (1) Our proposed PKM-ACT demonstrates great advantages with a significant performance margin compared to different sequential recommenders across all datasets, with an average performance improvement of 2.37%. It validates the effectiveness of our proposals. (2) The performance gains of PKM-ACT are more pronounced on long-term dataset. For ex-

Dataset	Metric	FPMC	GRU4Rec	SASRec	BERT4Rec	LinRec	HSTU	Ours	Improv.
ML-1M	NDCG@10 ↑	0.3220	0.4033	0.4190	0.3182	0.4158	0.4090	**0.4249**	1.41%
	NDCG@20 ↑	0.3606	0.4343	0.4471	0.3494	0.4444	0.4364	**0.4540**	1.54%
	HR@10 ↑	0.5233	0.6382	0.6606	0.5389	0.6556	0.6586	**0.6613**	0.11%
	HR@20 ↑	0.6760	0.7603	0.7712	0.6623	0.7689	0.7662	**0.7757**	0.58%
	MRR@10 ↑	0.2599	0.3300	0.3435	0.2497	0.3412	0.3312	**0.3517**	2.21%
	MRR@20 ↑	0.2705	0.3386	0.3512	0.2583	0.3490	0.3388	**0.3597**	2.28%
XLong	NDCG@10 ↑	0.2309	0.3409	0.3122	0.2363	0.2109	0.2632	**0.3507**	2.87%
	NDCG@20 ↑	0.2317	0.3417	0.3138	0.2375	0.2155	0.2672	**0.3534**	3.42%
	HR@10 ↑	0.3199	0.3824	0.4036	0.3054	0.3598	0.3457	**0.4199**	4.04%
	HR@20 ↑	0.3228	0.3928	0.4101	0.3104	0.3779	0.3612	**0.4228**	3.10%
	MRR@10 ↑	0.2008	0.3308	0.2813	0.2129	0.1624	0.2361	**0.3406**	2.96%
	MRR@20 ↑	0.2010	0.3310	0.2818	0.2133	0.1637	0.2372	**0.3413**	3.11%

Table 2: Main performance results, best results are marked in bold, second best results underlined.

Variants	Metric	ML-1M		
		NDCG ↑	HR ↑	MRR ↑
Ours	@10	**0.4249**	**0.6613**	**0.3517**
	@20	**0.4540**	**0.7757**	**0.3597**
Ours w/o `Query Net`	@10	0.4097	0.6558	0.3328
	@20	0.4381	0.7677	0.3406
Ours w/o `PKM`	@10	0.4132	0.6515	0.3387
	@20	0.4433	0.7699	0.3470
Ours w/o `Dispatcher`	@10	0.4099	0.6366	0.3390
	@20	0.4370	0.7443	0.3464

Table 3: Ablation studies on proposed hierarchical query network, product key memory layer and dispatcher mechanism on ML-1M.

ample, PKM-ACT achieves 1.49% average improvements on the relatively short-term ML-1M. The improvement is much more significant on long-term XLong dataset with average improvement of 3.25%, suggesting the substantial benefits of PKM-ACT in modelling long-term user behavior sequences.

Ablation Study

We perform a series ablation studies to demonstrate the contributions of the proposed components in PKM-ACT. As shown in Table 3, we conduct ablation on the hierarchical query network, the need for product key memory layer, and the dispatcher mechanism in action transformer. For the hierarchical query network, we replace it with average pooling of original behavior sequences as the query. When removing the entire PKM layer, we initialize the same learnable interest expert P for all users. We observe the following results on the ablation of PKM-ACT components: (1) All proposed components contribute to the overall performance of PKM-ACT. For example, removing PKM results in an average performance drop of 3.36% on ML-1M. (2) The proposed hierarchical query network is responsible for generating personalized query representation for each user, the naive average pooling of original sequences is inadequate for depicting different interest of users. (3) Equipped with the PKM layer, the capacity of model is significantly improved, which obviously boosts the recommendation performance. To be specific, removing PKM results in an average performance drop of 2.51% on ML-1M, which evidences the necessity of improve network capacity with PKM layer. (4) Dispatcher mechanism effectively aggravates and dispatch information between the original sequence and compressed interest expert which contributes an average performance of 3.80% on ML-1M. Overall, the ablation results suggest that all proposed components and the designed architecture in PKM-ACT are effective for sequential recommendation.

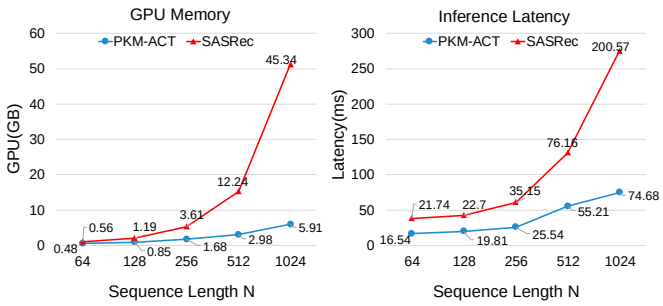

Figure 3: Model efficiency: PKM-ACT vs SASRec for training GPU memory uasge and inference latency in XLong.

Computational Scalability Study

To investigate the efficiency of proposed PKM-ACT, we evaluate the computational usage of the proposed PKM-ACT as compared to traditional Transformer baseline SASRec in XLong dataset, including GPU memory and inference latency.

Efficiency of Different Backbones. In Figure 3, we observe that PKM-ACT mechanism exactly reduces the GPU memory and inference latency cost, demonstrating the high efficiency of PKM-ACT. For example, PKM-ACT extraordinarily improve the modelling efficiency on Xlong, which reduces the approximately 90% GPU memory cost with $\times 2.68$

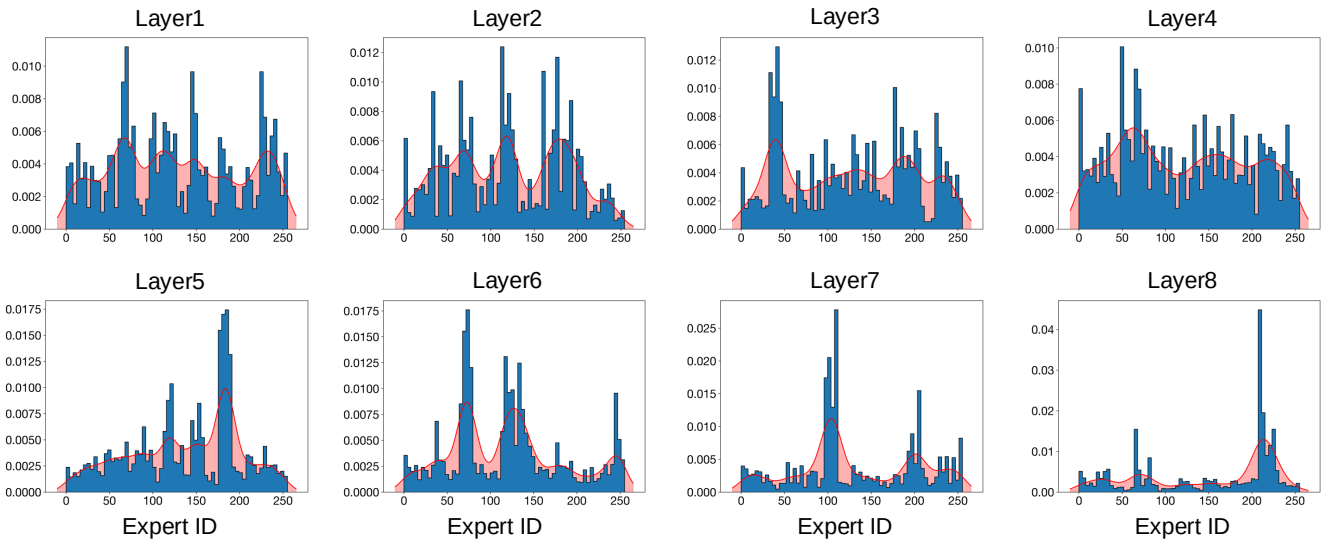

Figure 4: Visualization of the expert activation pattern of each layer on ML-1M.

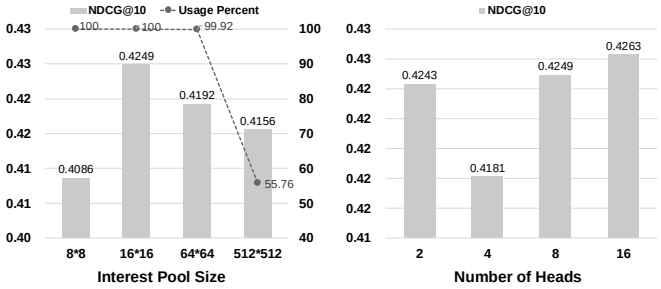

Figure 5: Hyperparameter sensitivity on ML-1M.

inference speed up on 1024 sequence length. Such result validates the theoretical analysis of linear complexity of PKM-ACT which reduces the complexity of original self attention from $\mathcal{O}(N^2)$ to $\mathcal{O}(Nk)$.

Efficiency of Sequence Lengths. To study the training cost of different sequence lengths for long-term sequential recommendation, we set the maximum sequence lengths in $\{64, 128, 256, 512, 1024\}$. In Figure 3, we observe that the computational complexity of the SASRec model exhibits a quadratic growth trend as the input sequence length increases. In contrast, the PKM-ACT model demonstrates an approximate linear increase in computational complexity. We also conclude that PKM-ACT reduces the computational complexity of SASRec consistently in all cases.

Hyperparameter Sensitivity

We investigate the hyperparameter sensitivity of PKM-ACT on the interest pool size K and the number of selected interest heads k on ML-1M dataset. Figure 5 illustrates the relationship between these hyperparameters and the NDCG@10 scores. For K, we observe that performance improved as the interest pool size increased from $8 * 8$ to $16 * 16$. However, further expansion to $512 * 512$ leads to a decline in performance. To explain this phenomenon, we examine the key usage rate (shown by the dashed line). At $512 * 512$, only 55% of keys are utilized, suggesting that an excessively large interest pool may be redundant for modeling diverse user interests, potentially harming model performance. Regarding k, we observe a significant performance improvement when increasing from 4 to 8, but only a modest gain when further increasing from 8 to 16. Overall, PKM-ACT demonstrates robust performance across varying numbers of interest heads k, while the interest pool size K requires careful selection for optimal performance.

Dissection of Expert Activation Patterns

To study the interest expert activation patterns of PKM layers, we visualize the results of each layer in Figure 4. The x-axis represents the different expert id and y-axis stands for the load of a particular expert. We find that the load distribution across experts is more balanced at bottom level while there exists multiple hot experts that always get a large share of tokens at higher level. This shows that the bottom PKM layers may learn common interest pattern of different users while the higher layer tends to model biased user's interest. This also validates that the hot experts and their hotness level varies across different layers and each PKM layer does learn diverse interest pattern among different users.

Conclusion

In this paper, we present a novel product key memory-based interest retrieval framework with action transformer, aiming at addressing the issue of high complexity of standard attention. Our proposed product key memory retrieval mechanism sparsely retrieves compressed user's interests representation from a number of interest expert pool with a negligible computational overhead. And the efficient action transformer with a novel dispatcher mechanism addresses the dilemma of training efficiency, inference efficiency and recommendation performance. Extensive investigations across various strong sequential recommendation architectures consistently demonstrate the performance improvements from our approach, on the challenging benchmark of ML-1M and XLong datasets.

References

Andoni, A.; Indyk, P.; Laarhoven, T.; Razenshteyn, I.; and Schmidt, L. 2015. Practical and optimal LSH for angular distance. *Advances in neural information processing systems*, 28.

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Chang, J.; Zhang, C.; Fu, Z.; Zang, X.; Guan, L.; Lu, J.; Hui, Y.; Leng, D.; Niu, Y.; Song, Y.; et al. 2023. TWIN: TWo-stage interest network for lifelong user behavior modeling in CTR prediction at kuaishou. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3785–3794.

Child, R.; Gray, S.; Radford, A.; and Sutskever, I. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Harper, F. M.; and Konstan, J. A. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4): 1–19.

He, Z.; Zhao, H.; Lin, Z.; Wang, Z.; Kale, A.; and McAuley, J. 2021. Locker: Locally constrained self-attentive sequential recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 3088–3092.

Hidasi, B.; and Karatzoglou, A. 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.

Hou, Y.; Hu, B.; Zhang, Z.; and Zhao, W. X. 2022. Core: simple and effective session-based recommendation within consistent representation space. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, 1796–1801.

Kang, W.-C.; and McAuley, J. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, 197–206. IEEE.

Katharopoulos, A.; Vyas, A.; Pappas, N.; and Fleuret, F. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, 5156–5165. PMLR.

Kitaev, N.; Kaiser, Ł.; and Levskaya, A. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Lample, G.; Sablayrolles, A.; Ranzato, M.; Denoyer, L.; and Jégou, H. 2019. Large memory layers with product keys. *Advances in Neural Information Processing Systems*, 32.

Liu, L.; Cai, L.; Zhang, C.; Zhao, X.; Gao, J.; Wang, W.; Lv, Y.; Fan, W.; Wang, Y.; He, M.; et al. 2023. Linrec: Linear attention mechanism for long-term sequential recommender systems. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 289–299.

Orvieto, A.; Smith, S. L.; Gu, A.; Fernando, A.; Gulcehre, C.; Pascanu, R.; and De, S. 2023. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, 26670–26698. PMLR.

Pancha, N.; Zhai, A.; Leskovec, J.; and Rosenberg, C. 2022. Pinnerformer: Sequence modeling for user representation at pinterest. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 3702–3712.

Parmar, N.; Vaswani, A.; Uszkoreit, J.; Kaiser, L.; Shazeer, N.; Ku, A.; and Tran, D. 2018. Image transformer. In *International conference on machine learning*, 4055–4064. PMLR.

Pi, Q.; Zhou, G.; Zhang, Y.; Wang, Z.; Ren, L.; Fan, Y.; Zhu, X.; and Gai, K. 2020. Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2685–2692.

Qiu, J.; Ma, H.; Levy, O.; Yih, S. W.-t.; Wang, S.; and Tang, J. 2019. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972*.

Raza, S.; and Ding, C. 2022. News recommender system: a review of recent progress, challenges, and opportunities. *Artificial Intelligence Review*, 1–52.

Ren, K.; Qin, J.; Fang, Y.; Zhang, W.; Zheng, L.; Bian, W.; Zhou, G.; Xu, J.; Yu, Y.; Zhu, X.; et al. 2019. Lifelong sequential modeling with personalized memorization for user response prediction. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 565–574.

Rendle, S.; Freudenthaler, C.; and Schmidt-Thieme, L. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, 811–820.

Roy, A.; Saffar, M.; Vaswani, A.; and Grangier, D. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9: 53–68.

Si, Z.; Guan, L.; Sun, Z.; Zang, X.; Lu, J.; Hui, Y.; Cao, X.; Yang, Z.; Zheng, Y.; Leng, D.; et al. 2024. TWIN V2: Scaling Ultra-Long User Behavior Sequence Modeling for Enhanced CTR Prediction at Kuaishou. *arXiv preprint arXiv:2407.16357*.

Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, 1441–1450.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, S.; Li, B. Z.; Khabsa, M.; Fang, H.; and Ma, H. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Wu, L.; Li, S.; Hsieh, C.-J.; and Sharpnack, J. 2020. SSE-PT: Sequential recommendation via personalized transformer. In *Proceedings of the 14th ACM conference on recommender systems*, 328–337.

Wu, S.; Sun, F.; Zhang, W.; Xie, X.; and Cui, B. 2022. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5): 1–37.

Yuan, E.; Guo, W.; He, Z.; Guo, H.; Liu, C.; and Tang, R. 2022. Multi-behavior sequential transformer recommender. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, 1642–1652.

Yue, Z.; He, Z.; Zeng, H.; and McAuley, J. 2021. Black-box attacks on sequential recommenders via data-free model extraction. In *Proceedings of the 15th ACM conference on recommender systems*, 44–54.

Yue, Z.; Wang, Y.; He, Z.; Zeng, H.; McAuley, J.; and Wang, D. 2024. Linear recurrent units for sequential recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 930–938.

Zhai, J.; Liao, L.; Liu, X.; Wang, Y.; Li, R.; Cao, X.; Gao, L.; Gong, Z.; Gu, F.; He, M.; et al. 2024. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. *arXiv preprint arXiv:2402.17152*.

Zhang, S.; Tay, Y.; Yao, L.; and Sun, A. 2018. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414*.

Zhang, T.; Zhao, P.; Liu, Y.; Sheng, V. S.; Xu, J.; Wang, D.; Liu, G.; Zhou, X.; et al. 2019. Feature-level deeper self-attention network for sequential recommendation. In *IJCAI*, 4320–4326.

Zhou, G.; Mou, N.; Fan, Y.; Pi, Q.; Bian, W.; Zhou, C.; Zhu, X.; and Gai, K. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 5941–5948.

Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; and Gai, K. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 1059–1068.

Reproducibility Checklist

This paper:

- Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes/partial/no/NA)
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes/no)
- Provides well marked pedagogical references for less-familiare readers to gain background necessary to replicate the paper (yes/no)

Does this paper make theoretical contributions? (yes/no)
If yes, please complete the list below.

- All assumptions and restrictions are stated clearly and formally. (yes/partial/no)
- All novel claims are stated formally (e.g., in theorem statements). (yes/partial/no)
- Proofs of all novel claims are included. (yes/partial/no)
- Proof sketches or intuitions are given for complex and/or novel results. (yes/partial/no)
- Appropriate citations to theoretical tools used are given. (yes/partial/no)
- All theoretical claims are demonstrated empirically to hold. (yes/partial/no/NA)
- All experimental code used to eliminate or disprove claims is included. (yes/no/NA)

Does this paper rely on one or more datasets? (yes/no)
If yes, please complete the list below.

- A motivation is given for why the experiments are conducted on the selected datasets (yes/partial/no/NA)
- All novel datasets introduced in this paper are included in a data appendix. (yes/partial/no/NA)
- All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes/partial/no/NA)
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. (yes/no/NA)
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. (yes/partial/no/NA)
- All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing. (yes/partial/no/NA)

Does this paper include computational experiments? (yes/no)
If yes, please complete the list below.

- Any code required for pre-processing data is included in the appendix. (yes/partial/no).
- All source code required for conducting and analyzing the experiments is included in a code appendix. (yes/partial/no)
- All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (yes/partial/no)
- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no)
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (yes/partial/no/NA)
- This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. (yes/partial/no)
- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (yes/partial/no)

- This paper states the number of algorithm runs used to compute each reported result. (yes/no)
- Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (yes/no)
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (yes/partial/no)
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (yes/partial/no/NA)
- This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (yes/partial/no/NA)