# KuaiFormer: Transformer-Based Retrieval at Kuaishou

Chi Liu
Kuaishou Technology, Beijing, China
liuchi05@kuaishou.com

Jiangxia Cao
Kuaishou Technology, Beijing, China
caojiangxia@kuaishou.com

Rui Huang
Kuaishou Technology, Beijing, China
huangrui06@kuaishou.com

Kai Zheng
Kuaishou Technology, Beijing, China
zhengkai@kuaishou.com

Qiang Luo
Kuaishou Technology, Beijing, China
luoqiang@kuaishou.com

Kun Gai
Unaffiliated, Beijing, China
gai.kun@qq.com

Guorui Zhou
Kuaishou Technology, Beijing, China
zhouguorui@kuaishou.com

## ABSTRACT

In large-scale content recommendation systems, retrieval serves as the initial stage in the pipeline, responsible for selecting thousands of candidate items from billions of options to pass on to ranking modules. Traditionally, the dominant retrieval method has been Embedding-Based Retrieval (EBR) using a Deep Neural Network (DNN) dual-tower structure. However, applying transformer in retrieval tasks has been the focus of recent research, though real-world industrial deployment still presents significant challenges. In this paper, we introduce KuaiFormer, a novel transformer-based retrieval framework deployed in a large-scale content recommendation system. KuaiFormer fundamentally redefines the retrieval process by shifting from conventional score estimation tasks (such as click-through rate estimate) to a transformer-driven Next Action Prediction paradigm. This shift enables more effective real-time interest acquisition and multi-interest extraction, significantly enhancing retrieval performance. KuaiFormer has been successfully integrated into Kuaishou App's short-video recommendation system since May 2024, serving over 400 million daily active users and resulting in a marked increase in average daily usage time of Kuaishou users. We provide insights into both the technical and business aspects of deploying transformer in large-scale recommendation systems, addressing practical challenges encountered during industrial implementation. Our findings offer valuable guidance for engineers and researchers aiming to leverage transformer models to optimize large-scale content recommendation systems.

## CCS CONCEPTS

• **Information systems → Recommender systems**.

## KEYWORDS

Short-Video Recommendation; Transformer; User Interest Modeling;

## 1 INTRODUCTION

The Transformer [32] architecture has demonstrated significant success across multiple domains, with notable models such as BERT [7] and GPT [1, 3] in natural language processing (NLP), and Vision Transformers [2, 19, 23] in computer vision (CV). These achievements underscore the Transformer's remarkable capabilities in sequence modeling and parallelization. In the field of recommendation systems, Transformer-based architectures like SASRec [14] and Bert4Rec [29] have also shown potential. However, these academic efforts often fail to address certain industrial challenges, which has limited their effectiveness in driving business success in large-scale recommendation systems, such as those at Kuaishou.

Short-video recommendation poses unique challenges that demand advanced modeling techniques. The diverse nature of short-video content and the rapid evolution of user interests necessitate real-time adaptation to accurately capture these dynamic preferences. Users typically watch hundreds of short-videos each day, expressing preferences across a wide range of interest domains, while the system actively pushes diverse content to mitigate aesthetic fatigue and avoid the "filter bubble" effect. As a result, models that rely on daily updates, such as PinnerFormer[25], struggle to adapt to users' evolving content needs. Moreover, traditional approaches like SASRec and Bert4Rec, which compress user behavior into a single interest vector, lack the capacity to accurately capture the full spectrum of user interests reflected in these interactions.

To more effectively capture complex user interests, models like MIND [18] and ComiRec [4] employ techniques such as capsule networks to extract multiple interest vectors from user action sequences. However, because these models do not utilize native Transformer-based architectures, they are limited in fully leveraging the advantages offered by Transformers. Furthermore, they do not address the performance overhead associated with processing long sequences, which is a significant concern in industrial applications.

To effectively implement the Transformer model within Kuaishou's large-scale short video recommendation system, this study conducts a detailed analysis of these specific challenges in the recommendation field and proposes a series of concise and effective solutions tailored to address these issues.

- **How to train with billion-scale item set**:
  Large language models (LLMs) typically leverage next token prediction as the pretraining task. This involves calculating the probabilities of all tokens in the vocabulary being the next token, based on the historical sequence, and selecting the one with the highest probability as the next predicted token. General-purpose language models usually contain fewer than 100,000 tokens in their vocabularies [30]. In the context of the Kuaishou short video recommendation system, the candidate pool contains

billions of short videos, making it computationally prohibitive to compute probabilities for all candidates using a naive softmax approach. Efficient methods are therefore required to address the challenges of large-scale candidate selection while maintaining model performance.

- **How to capture user's multi-interests**: Different with language which have a clear semantics direction when predicting the next token. In our short-video services, users always have multiple interest points and a higher tolerance for short-videos watching. As a result, there maybe exists multiple short-videos with completely different semantics that are served as 'positive' next items at same time, which is unfriendly to vanilla Transformer learning.
- **How to extend to longer sequences with fewer computation resources**: Different with large language model could stack very deeper and wider Transformers to achieve best performance with extremely higher computation resources. As a recommendation model, our model need to response a large amount request (about > 50 billion requests every day), thus our Retrieval model should achieve the balance between efficiency and effectiveness. Particularly, the Transformer time complexity is $\mathcal{O}(n^2 d)$, where $n$ denotes the input sequence length, $d$ means hidden state dimension [15]. Thereby Transformer-based model is sensitive with sequences length and we need to devise specific module to accelerate longer sequences training.

In this work, we present KuaiFormer, our latest advancements in real-time industrial retrieval, which delivered the most significant improvements in the Retrieval stage at Kuaishou over the past year. Specifically, we introduce several reliable modifications to adapt Transformer to industrial retrieval scenarios: a customized softmax learning objective for stable model training, multiple query tokens to capture users' diverse interests, and a historical sequence compression mechanism to improve the efficiency of long-sequence modeling.

- **Smooth In-Batch Softmax Loss with LogQ Correction**: To avoid directly training on a billion-scale item set, we first employ in-batch softmax as the learning objective for KuaiFormer. However, in-batch softmax inevitably introduces sampling bias, deviating from uniform item sampling: popular items have more chances to be selected as negative samples, which can lead to performance degradation. We apply the widely-used logQ correction method [35] to correct for sampling bias. Additionally, in short-video services, users often have higher tolerance for watching, which reduces the confidence that negative samples in in-batch sampling truly represent items users dislike. Therefore, instead of using strict 0/1 labels for training, we incorporate label smoothing techniques [24] to mitigate training noise and enhance model robustness.
- **Multi-interests Query Tokens**: To capture users' diverse interests, we drew inspiration from the [CLS] token in BERT, which introduces a learnable token to compress the original input information into a holistic sequence representation. We extend this concept in KuaiFormer by introducing multiple learnable tokens, combined with a multi-interest training strategy to extract distinct user interest representations from historical item sequences. Specifically, KuaiFormer's learnable query tokens leverage a causal masking mechanism, enabling subsequent interest tokens to fully interact with preceding interest token representations, thereby achieving more effective interest disentanglement.
- **Adaptive Item Compression Mechanism**: To address the efficiency challenges of modeling longer sequences, we made an intuitive assumption: compared with the most recently watched short videos, users' memories of earlier videos are more vague. Therefore, we can apply coarse-grained modeling to earlier items while using fine-grained modeling for the latest ones. Based on this, we devised an adaptive item compression mechanism: first, we divide the earlier item sequences into several groups, compressing each group into a single representation to reduce the input sequence length. This series of compressed representations is then concatenated with the most recent items, forming the token sequence input for the model.

The main contributions of our work are as follows:

- We propose our next-generation retrieval approach, KuaiFormer, to our knowledge, this work is the first real-time retrieval model by the pure Transformer architecture in industrial-scale RecSys.
- We conduct extensive offline and online experiments to show KuaiFormer superiors, which contribute +0.360%/+0.126%/+0.411% online video watch time gains to Kuaishou short-video services.
- We offer insights into both model and architectural aspects, addressing practical challenges encountered in industrial RecSys deployment. Our experience offer valuable guidance for engineers and researchers aiming to leverage Transformer to build better industrial RecSys.

## 2 METHODOLOGY

This section describes how KuaiFormer works in our recommendation system. We first briefly explain the problem definition of our KuaiFormer in training and inference. Then, we present KuaiFormer base workflow including the feature engineering and model architecture. Next, we show the details how to model longer sequence and capture user's multiple interests. Finally, we give our loss function to achieve a stable training in billion-scale item set.

### 2.1 Problem Statement

In a general recommendation task, we leverage users' historical watched items to model their interests and predict the next video they are likely to engage with.

For each user, let $\{(x_1, \mathrm{f}_1), (x_2, \mathrm{f}_2), \ldots, (x_n, \mathrm{f}_n)\}$ represents his/her recent positive watched short-video information in chronological order, where the $x_i \in \mathbb{R}$ denotes to a short-video ID from entire short-video set $\mathcal{X}$, the $\mathrm{f}_i \in \mathbb{R}^{|\mathcal{F}|}$ denotes the watching side-information according to attribute set $\mathcal{F}$. Generally, we hand-craft about 10+ short-video watching attributes for interests modeling, including: watching time, interaction labels (e.g., like, comment), short-video duration, multi-modal category tag ID and so on.

According to the sequence $\{(x_1, \mathrm{f}_1), (x_2, \mathrm{f}_2), \ldots, (x_n, \mathrm{f}_n)\}$, in training procedure, since we know the next item $x_{n+1}$, thus we could maximize the following likelihood to capture users' real-time interests:

$$\{\mathbf{u}_1,\ldots,\mathbf{u}_k\} = \texttt{model}(\{(x_1,\mathrm{f}_1),(x_2,\mathrm{f}_2),\ldots,(x_n,\mathrm{f}_n)\}), \quad \texttt{argmax}_{x_{n+1}\in\mathcal{X}} \mathrm{P}\left(x_{n+1}|\{\mathbf{u}_1,\ldots,\mathbf{u}_k\}\right) \tag{1}$$

where $\mathrm{P}(\cdot)\in\mathbb{R}^{|\mathcal{X}|}$ is the probability of the candidate item in $\mathcal{X}$ and $\mathbf{u}$ denotes the user interest representation.

In inference, as a retrieval model, our goal is to find a small group of candidate items that related with user's real-time interests:

$$\{\hat{x}^1_{n+1},\hat{x}^2_{n+1},\ldots\} = \texttt{ANN}(\{\mathbf{u}_1,\ldots,\mathbf{u}_k\},\mathcal{X}), \tag{2}$$

where $\texttt{ANN}(\cdot)$ denote the Approximate Nearest Neighbor technique, which is a fast search system (e.g., faiss [13]) to find the candidates with user interest representation $\mathbf{u}$. The $\{\hat{x}^1_{n+1},\hat{x}^2_{n+1},\ldots\}$ means the highest hundreds related items.

## 2.2 Base Backbone

In this section, we introduce the base backbone of our KuaiFormer, which contains short-video embedding module and a stacked Transformer module.

*2.2.1 Embedding Module.* In embedding mapping stage, we utilize two type embedding layers to consider different type of attributes to form as model input:

- *One-hot discrete attributes embedding layer*: As the most common type of embedding layer, it is mainly used to model **discrete features, including short-video ID, tag ID, interaction labels, etc**. Take the short-video ID mapping stage as example, we assign a parameter matrices $\mathbf{X}\in\mathbb{R}^{|\mathcal{X}|\times d}$ to store their embeddings, where $d$ is the embedding dimension. Thereby we can easily obtain the corresponding embedding $\mathbf{x}_i\in\mathbb{R}^d$ by a straightforward lookup operation for arbitrary short-video ID $x_i$:

$$\mathbf{x}_i = \texttt{LookUp}(\mathbf{X}, x_i), \tag{3}$$

- *Bucket continuous attributes embedding layer*: Instead of the discrete attributes, this layer aims to embed the **continuous attributes, including watching time, short-video duration, etc**. For these continuous attributes, since their prior distributions are different, we actually customize each attribute and design a special bucketing strategy to discretize them while retaining the distribution characteristics. Take the short-video duration $\mathrm{f}^{\texttt{dura}}_i$ as example, we utilize a uniform bucket strategy to divide it in 1000 buckets with maximize 300s, and the lookup its embedding from parameter matrix $\mathbf{F}^{\texttt{duration}}\in\mathbb{R}^{1000\times d}$:

$$\begin{aligned} \mathrm{f}^{\texttt{bucket\_dura}}_i &= \texttt{int}\left(\frac{\texttt{min}(\mathrm{f}^{\texttt{dura}}_i, 300)}{300} * 1000\right) \\ \mathrm{f}^{\texttt{dura}}_i &= \texttt{LookUp}(\mathbf{F}^{\texttt{dura}}, \mathrm{f}^{\texttt{bucket\_dura}}_i) \end{aligned} \tag{4}$$

where the $\mathrm{f}^{\texttt{bucket\_dura}}_i$ is a integer denotes the mapping discrete ID index for continuous attribute $\mathrm{f}^{\texttt{dura}}_i$.

*2.2.2 Transformer Module.* On top of the above embedding module, we first utilize them to form the watched short-video sequence information. For each information $(x_i,\mathrm{f}_i)$, we can map them as:

$$\mathbf{t}_i = \texttt{MLP}([\mathbf{x}_i,\mathbf{f}_i]) \tag{5}$$

where the $\texttt{MLP}(\cdot)\in\mathbb{R}^{d\times d}$ is a feed-forward neural network, $[\cdot,\cdot]$ indicates concat operator, and the $\mathbf{t}_i$ is the final unit token representation. By analogy, we could transform the input representation sequences as $\{\mathbf{t}_1,\mathbf{t}_2,\ldots,\mathbf{t}_n\}$.

Afterwards, we utilize Transformer modeling the input sequence, to extract user interest representation. Specifically, we follow the Llama Transformer architecture [31] as our backbone, which mainly includes: (1) RMS Norm technique to avoid the gradient vanishing/explored problems, (2) multi-head masked self-attention mechanism captures the complex token dependency, (3) a point-wise feed-forward layer enhances model's non-linearity capacity.

$$\mathbf{u} = \texttt{Causal\_Transformer}(\{\mathbf{t}_1,\mathbf{t}_2,\ldots,\mathbf{t}_n\}, L, M) \tag{6}$$

As shown in Figure, $\mathbf{u}$ is the top layer lastest position output, $L$ is a hyper-parameter to control the depth of Transformers, $M$ is the number of multi-heads.

## 2.3 Towards Longer Sequence

Actually, the generated representation $\mathbf{u}$ could support the training stage (in Eq.1) and inference stage (in Eq.2).

However, we need to handle a large volume of requests within just a few milliseconds. At peak times, this means managing over 100,000 requests per second. Thus the straightforward way is hard to scale to a longer input sequence to provide a high-quality representation for training and inference. In our experiments, we find when the sequence length $n$ is extended from 64 to 256, the corresponding computing resources increases by 6 times, which is unacceptable. Therefore, a challenging problem is how to model a longer sequences with fewer computation resources? For users, the most recent videos leave a stronger impression, while older browsing history is relatively vague. This observation motivates us to compress the earlier watched short-videos to reduce the sequence length. In KuaiFormer, we devise a simple-yet-effective adaptive item compression mechanism in three steps:

(1) We first divide the input sequence into three parts according to their position as earlier part, middle part, and latest part. For the earlier part and middle part, we merge 64 adjacent items and 16 adjacent items as one group, respectively.
(2) For the earlier/middle item groups, we then utilize a single-layer bidirectional Transformer without mask strategy to aggregate them as a grouped item representation. Take the earlier item group as example, such process can be formulated as:

$$\begin{aligned} \mathbf{t}^{\texttt{early}}_1 &= \texttt{Mean}(\texttt{Bi\_Transformer}(\{\mathbf{t}_1,\ldots,\mathbf{t}_{64}\}, M) \\ \mathbf{t}^{\texttt{early}}_2 &= \texttt{Mean}(\texttt{Bi\_Transformer}(\{\mathbf{t}_{65},\ldots,\mathbf{t}_{128}\}, M) \\ \mathbf{t}^{\texttt{mid}}_1 &= \texttt{Mean}(\texttt{Bi\_Transformer}(\{\mathbf{t}_{129},\ldots,\mathbf{t}_{145}\}, M) \\ &\ldots \\ \mathbf{t}^{\texttt{mid}}_5 &= \texttt{Mean}(\texttt{Bi\_Transformer}(\{\mathbf{t}_{193},\ldots,\mathbf{t}_{208}\}, M) \end{aligned} \tag{7}$$

Similarly, the calculation also holds for middle item groups, and leads to the results $\mathbf{t}^{\texttt{mid}}_i$.

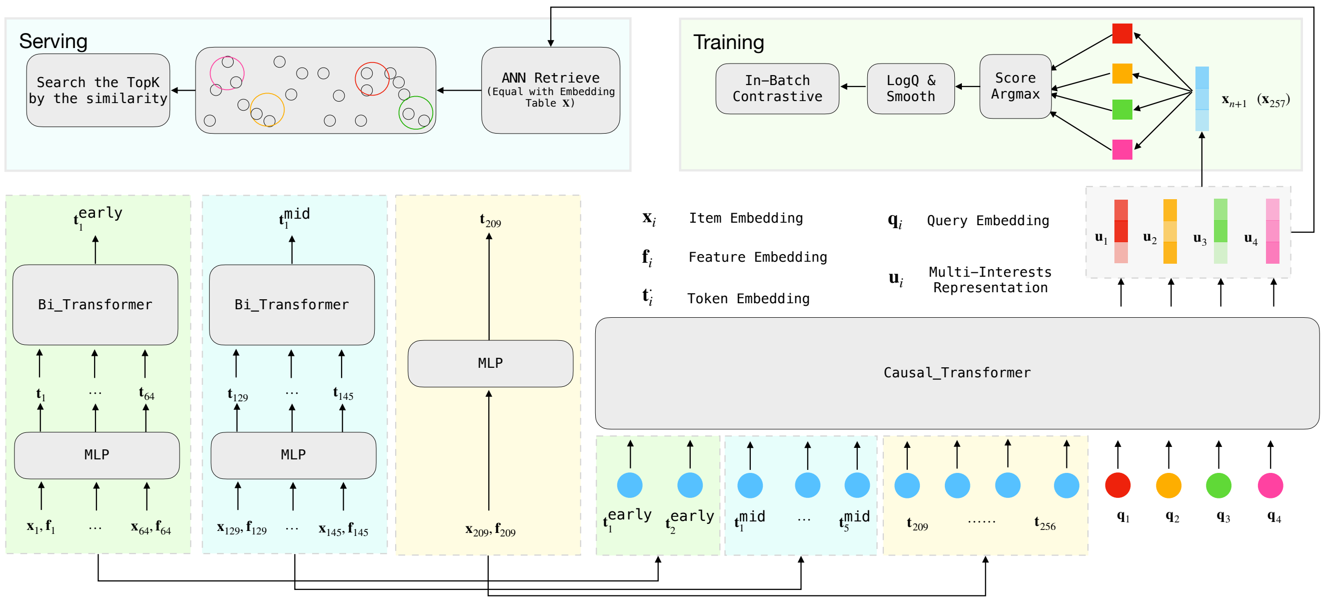


**Figure 1: KuaiFormer architecture under the length setting 256, and 4 query tokens, where the $\mathbf{t}_1^{\text{early}}$ denotes the early item compression, the $\mathbf{t}_1^{\text{mid}}$ denotes the middle item compression. We can effectively model a longer sequence of 256 through the use of feeding a shorter sequence of 55 for efficient training and inference.**

(3) Finally, we could explicit the compressed item sequences to replace the original input to generate user interests as:

$$\mathbf{u} = \text{Causal\_Transformer}(\{\mathbf{t}_1^{\text{early}}, \mathbf{t}_2^{\text{early}}\} \oplus \{\mathbf{t}_1^{\text{mid}}, \dots, \mathbf{t}_5^{\text{mid}}\} \oplus \{\mathbf{t}_{209}, \dots, \mathbf{t}_{256}\}, L, M) \tag{8}$$

Based on the adaptive item compression strategy, scaling the sequence length from 64 to 256, we find our model only increase 10% additional computation resources.

## 2.4 Towards Multiple Interests

Additionally, in our short-video services, users always have multiple interest points, therefore utilizing a single representation is hard to express users' complex real-time dynamic interests. For sequence modeling, capturing the diverse interests embedded within the sequence becomes increasingly important as the sequence length grows. With KuaiFormer, we also explore a similar challenge: how can we model users' multi-interests within the Transformer paradigm?

Motivates by the great success of special '[CLS]' token of BERT, which introduces a learnable token to summarize input sequence information for downstream tasks. We consider employ $k$ different special tokens act as queries token to extract different user interests.

$$\{\mathbf{u}_1, \dots, \mathbf{u}_k\} = \text{Causal\_Transformer}(\{\mathbf{t}_1^{\text{early}}, \mathbf{t}_2^{\text{early}}\} \oplus \{\mathbf{t}_1^{\text{mid}}, \dots, \mathbf{t}_5^{\text{mid}}\} \oplus \{\mathbf{t}_{209}, \dots, \mathbf{t}_{256}\} \oplus \{\mathbf{q}_1, \dots, \mathbf{q}_k\}, L, M) \tag{9}$$

where the $\{\mathbf{q}_1, \dots, \mathbf{q}_k\}$ are $k$ different user interests query tokens, and they are concatenated with the item compression sequence. And the $\{\mathbf{u}_1, \dots, \mathbf{u}_k\}$ are final user interests representations to fit user dynamic behaviours. It is worth noting that in the previous retrieval approaches, their multiple interest representations are separate learned and invisible to each other. At KuaiFormer, our query tokens also follow the auto-regressive paradigm, which allows the latter interests tokens can fully interact with former interests tokens representations to achieve better interests disentanglement.

Under the multiple interests representations, we calculate the prediction score utilize the argmax function. For instance, given the next positive short-video $x_{n+1}$, we formulate the process as:

$$\text{P}(x_{n+1}|\{\mathbf{u}_1, \dots, \mathbf{u}_k\}) = \frac{\text{Score}_{x_{n+1}}}{\sum_x^{\mathcal{X}} \text{Score}_x}, \quad \text{where} \quad \text{Score}_{x_{n+1}} = \text{argmax}(\{\mathbf{x}_{n+1}^\top \mathbf{u}_1, \dots, \mathbf{x}_{n+1}^\top \mathbf{u}_k\}) \tag{10}$$

where the prediction score is selected from the highest interest representation.

## 2.5 Towards Stable Training

Up to now, we have introduced the learning process to generate multiple user interests representations with a longer sequence. In this section, we will dive into the loss function designing to answer the most challenging problem: How to train a Transformer-based model with billion-scale item set.

Generally, the Transformer-based model always trained with softmax function, to maximize the next positive token probability while minimize all other token probabilities. However, in recommendation task, it is impractical to train our model in full billion-scale item set directly, thus we first employ the in-batch softmax as

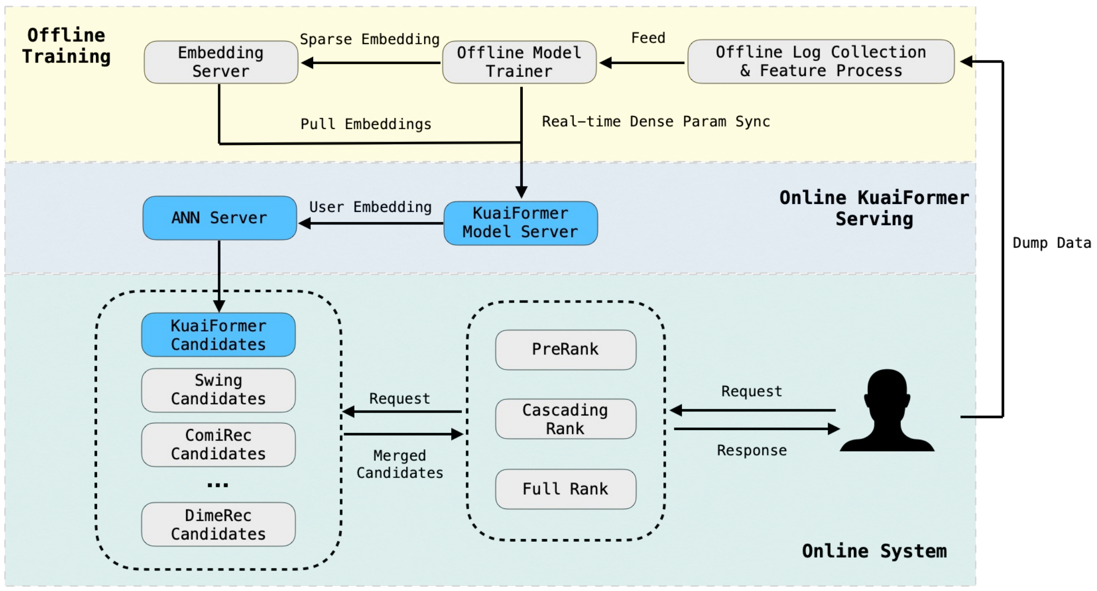


**Figure 2: Deployment Architecture**

our KuaiFormer learning objective.

$$\mathcal{L}(x_{n+1}) = -\log\left(\frac{e^{\text{Score}_{x_{n+1}}}}{e^{\text{Score}_{x_{n+1}}} + \sum_{\bar{x}_{n+1}}^{\mathcal{B}} e^{\text{Score}_{\bar{x}_{n+1}}}}\right) \tag{11}$$

where the $\mathcal{B}$ denotes the in-batch negative short-video set, the $x_{n+1}$ denotes the positive item and the $\bar{x}_{n+1}$ indicates an arbitrary negative item.

However, the in-batch softmax will inevitably introduce the sampling bias to away from uniform item sampling: the top popular items have more opportunities to become negative samples, which will lead model performance degeneration inevitably. Therefore, we follow the wide-used logQ correction method to refine items sampling probability to achieve a balanced in-batch softmax [35]:

$$\mathcal{L}_{\text{logQ}} = -\log(\text{Score}^{\text{logQ}}_{x_{n+1}})$$
$$= -\log\left(\frac{e^{\text{Score}_{x_{n+1}} - \log(Q_u(x_{n+1}))}}{e^{\text{Score}_{x_{n+1}} - \log(Q_u(x_{n+1}))} + \sum_{\bar{x}_{n+1}\in\mathcal{B}} e^{\text{Score}_{\bar{x}_{n+1}} - \log(Q_u(\bar{x}_{n+1}))}}\right)$$

In short-video services, users tend to have a higher tolerance for watching various content, making it challenging to classify the next item as definitively "positive" or "negative." Consequently, we avoid using strict binary (0/1) labels for model training. Instead, we employ a smoothing technique to mitigate training noise and improve reliability in content labeling [24].

$$\mathcal{L}^{\text{smooth}}_{\text{logQ}} = \begin{cases} -(1-\alpha)\log(\text{Score}^{\text{logQ}}_{x_{n+1}}) \\ \sum_{\bar{x}_{n+1}}^{\mathcal{B}} -\frac{\alpha}{|\mathcal{B}|}\log(\text{Score}^{\text{logQ}}_{\bar{x}_{n+1}}) \end{cases} \tag{12}$$

where the $\mathcal{L}^{\text{smooth}}_{\text{logQ}}$ is the final training objective in our KuaiFormer, $\alpha$ is a hyperparameter used to control the smoothing effect. After our model training convergence, we utilize the ANN technique to search the top-K short-videos for each user recommendation request in Eq.(2). The overall training and serving procedure are shown in Figure 1.

## 3 DEPLOYMENT ARCHITECTURE

In this section, we will present the comprehensive deployment architecture of KuaiFormer in an industrial streaming video recommender system, which serves the largest Kuaishou's recomendation scenario in the retrieval stage. KuaiFormer continuously receives online data for training and updates its parameters to the online model service at a minute-level frequency.

As shown in Figure 2, the KuaiFormer retrieval model is trained on an industrial curated distributed training framework in an online learning way. The short video content recommendation system responds to requests from hundreds of millions of users daily. User requests first pass through the retrieval system, which is composed of multiple independent retrieval pathways, such as the classic Swing[34], GNN[9], Comirec[4], Dimerec[21], GPRP [36], etc. KuaiFormer is introduced as a new retrieval pathway into the retrieval system. The retrieval candidates from all retrieval pathways are aggregated and deduplicated before being sent to the ranking system. In our system, the initial coarse-grained ranking is performed through a pre-rank stage, then a cascading rank stage, followed by a fine-grained full rank stage to obtain the final set of short videos presented to the user. The offline logging system records real-time user feedback, including dense feedback signals such as watch time and relatively sparse interactions (likes, follows, shares). All offline logs are processed into training records and aggregated for transmission to the offline training framework. To enhance system efficiency, we use a dedicated embedding server to store sparse

**Table 1: Offline Performance (%) comparison with increase from second-highest to highest.**

| Metrics | GPRP | SASRec | DimeRec | ComiRec | Swing | GNN | KuaiFormer | Improve over runner-up |
|---|---|---|---|---|---|---|---|---|
| HR@50 | 3.57% | 1.17% | 1.54% | 0.95% | 0.14% | <u>3.97%</u> | **4.21%** | 6.05% |
| HR@100 | <u>5.21%</u> | 2.51% | 2.49% | 1.93% | 0.29% | 4.6% | **6.58%** | 26.30% |
| HR@500 | <u>11.31%</u> | 5.37% | 6.99% | 7.02% | 1.54% | 8.89% | **15.61%** | 38.12% |
| HR@1000 | <u>15.70%</u> | 7.22% | 9.17% | 10.06% | 2.03% | 10.26% | **19.88%** | 26.62% |

**Table 2: Online A/B testing results of Short-Video services at Kuaishou.**

| Applications | Video Watch Time | Total App Usage Time | Usage Time per User | Avg. Time per Video View | Video Views | Likes | Follows | Comments | Novel Surprise |
|---|---|---|---|---|---|---|---|---|---|
| Kuaishou Single Page | +0.360% | +0.184% | +0.157% | +0.265% | +0.095% | +0.256% | +0.600% | +0.553% | +0.235% |
| Kuaishou Lite Single Page | +0.126% | +0.127% | +0.107% | +0.060% | +0.158% | +0.286% | +0.222% | +0.062% | +0.181% |
| Kuaishou Double Page | +0.411% | – | – | – | – | +0.399% | +1.254% | +1.463% | – |

embeddings, while the fewer dense model parameters are periodically transmitted to the online system for real-time inference of user embeddings. The final retrieval results are obtained through efficient ANN retrieval algorithms (such as Faiss[13], ScaNN[8]). In our practice, we employ GPU brute-force methods to retrieval the TopK candidates.

## 4 EXPERIMENTS

In this section, we give detailed analyses to answer the following major research questions (RQs):

(1) **RQ1**: Does our KuaiFormer achieves SOTA offline performance compared with strong retrieval methods?
(2) **RQ2**: Does our KuaiFormer contributes online gains in our Short-Video service significantly?
(3) **RQ3**: How does different hyper-parameter settings influence KuaiFormer performance?

### 4.1 Experimental Setting

*4.1.1 Dataset.* We conduct experiments at our short-video data-streaming, which is the **largest** recommendation scenario at Kuaishou, including over **400 Million users and 50 Billion logs** every day.

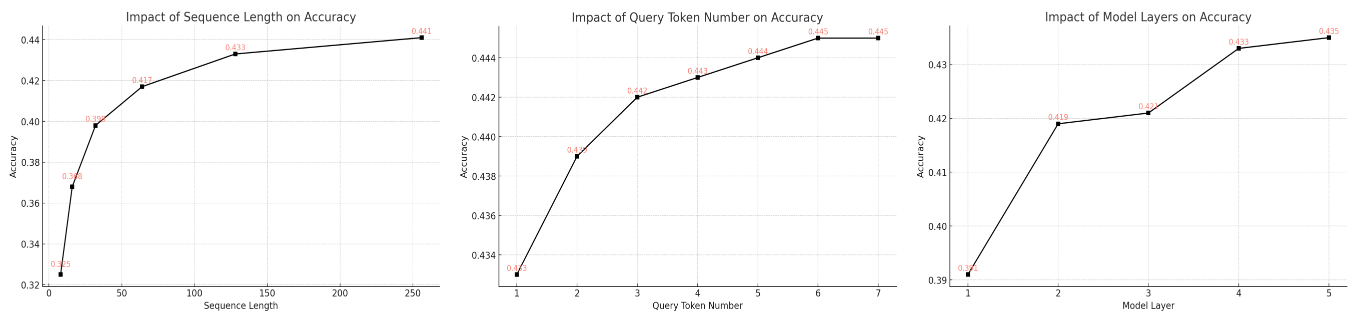
*4.1.2 Evaluation Protocol.* For the performance estimation, we apply two metrics to compare with baselines and our model variants, the online Hit Rate and offline Accuracy.

- Online **Hit Rate**: For a fair comparison with different methods, we utilize the users' online requests results to estimate different model ability. Specifically, we calculate the hit rate between the real user viewed items whether they are retrieved by the corresponding models, e.g., does the viewed item is retrieved in the top 50/100/500/100 results.
- Offline **Accuracy**: For evaluating the performance of our KuaiFormer variants, since the models are trained in an in-batch setting and use online learning to continuously incorporate the latest logs, the accuracy when the training loss stabilizes with the same batch size can assess the model's capability. This offline training accuracy reflects the model's ability to fit user behavior.

*4.1.3 Baselines.* In Kuaishou's short video recommendation scenario, multiple retrieval models are employed simultaneously to maximize the diversity of content supply and meet user needs. We compared KuaiFormer with the following representative strong retrieval methods which are deployed at Kuaishou: (1) Item2Item based methods: Swing [34]; (2) Multi-interests based methods: ComiRec [4]; (3) Graph based methods: GNN [9]; (4) Diffusion based methods: DimeRec [21]; (5) List-wise based methods: GPRP [36]. Except them, we also implement the SASRec for a fair comparison, although it has not achieved online contribution at Kuaishou [4].

### 4.2 Performance Comparisons (RQ1)

To accurately assess the offline coverage rate of the retrieval model, we replay real online requests and invoke these models, each returning 50-1000 results. The Hit rate between the model recommendations and the actual content viewed by users is shown in Table 1. According it, we have the following observations: (1) The classic sequence modeling approach, SASRec, performs poorly in terms of hit rate, suggesting that the multi-interest module in KuaiFormer contributes to covering a broader range of user interests. (2) Compared with the famous multi-interests method ComiRec [4], our KuaiFormer shows consistent improvement in all metrics, which validates that utilizing the Transformer as base encoder to extract user preference is effective. (3) Since GPRP is trained using recommendation funnel data from retrieval to ranking stages, it achieves a higher hit rate than other traditional methods with a retrieval number range of 100 to 1000. However, KuaiFormer improves hit rate by over 25% compared to GPRP. This indicates that the Next



a. Impact of Sequence Length on Accuracy with Fixed Query token number(1) and Layer number(4)

b. Impact of Query token number on Accuracy with Fixed Sequence Length(128) and Layer number(4)

c. Impact of Layer number on Accuracy with Fixed Sequence Length(128) and Query token number(1)

**Figure 3: Impact of Sequence Length, Query token number and Layer number on Accuracy**

Action Prediction modeling approach provides a stronger fit to the training set, even without explicitly incorporating recommendation funnel data.

## 4.3 Online A/B Test (RQ2)

To measure the precise improvements of KuaiFormer contribution to our short-video service, we conduct comprehensive one week A/B testing with 10% of users in the three largest scenarios on Kuaishou: Kuaishou Single/Double page and Kuaishou Lite Edition Single page. In Table 2, we shown the watching-time and interaction metrics of our KuaiFormer. Due to varying priorities across different scenarios, certain metrics are omitted on the Double Page. However, the most critical video watch time metric has been preserved to ensure essential insights remain accessible. Notably, the about 0.1% improvement in Video Watch Time is a statistically significant change to give satisfactory contribution, since our platform has 400 Million active users per day. According Table 2, our KuaiFormer achieves +0.360%, +0.126%, +0.411% improvements in terms of the video watch time under the three largest scenarios, **which is one of the most significant retrieval experiments at Kuaishou in 2024**. KuaiFormer has also demonstrated significant improvements in engagement metrics, such as increased Likes, Follows, and Comments. This indicates that KuaiFormer effectively adapts to evolving user needs, accurately captures user interests, and recommends relevant content, thereby enhancing user engagement and satisfaction. Additionally, the Novel Surprise metric has improved, which measures the model’s ability to help users discover new interests. This suggests that the multi-interest mechanism can capture both mainstream and niche interests, introducing users to new areas of interest.

## 4.4 Hyperparameter Impact (RQ3)

This section explores four hyper-parameter effects in KuaiFomer: the sequence length, query token number, layer number and item compression mechanism.

*4.4.1 Sequence Length Impact.* As seen in the table 3 (a), increasing the sequence length from 8 to 256 steadily improves the accuracy from 0.325 to 0.441. This suggests that increasing the sequence length allows the model to capture more context, which improves its performance. However, the improvement rate decreases as the sequence length increases, indicating diminishing returns. While accuracy continues to increase, the gain from doubling the sequence length becomes smaller, especially for sequence lengths beyond 64.

*4.4.2 Query token number’s impact.* The data in Table 3 (b) reveals a clear relationship between the number of query tokens and the model’s accuracy. As the number of tokens increases, the accuracy improves consistently, suggesting that additional tokens provide the model with more context, enabling better predictions. However, the rate of improvement decreases as the token number rises, indicating diminishing returns. The gain in accuracy becomes smaller with each additional token, implying that the model may reach a saturation point beyond which further tokens contribute little to overall performance. For instance, the increase in accuracy from token 1 to token 2 is 0.006, while the increase from token 6 to token 7 is only 0.001, demonstrating that after a certain number of tokens, the benefit of adding more becomes minimal.

Since we retrieve a certain number of videos for each query token and then uniformly rank and truncate them to a fixed number, an excessive number of query tokens will result in fewer videos being retrieved per token. Additionally, too many query tokens will increase computational resource consumption, so the number of query tokens cannot be increased indefinitely. In this experiment, 6 tokens appear to represent an optimal balance, where accuracy gains begin to level off, making it unnecessary to further increase the number of tokens for substantial improvements.

*4.4.3 Layer number’s impact.* Table 3 (c) demonstrates the impact of increasing model layers on accuracy, while holding the sequence length and query token number constant. The results show a consistent improvement in accuracy as the number of layers increases from 1 to 5.

The increase in accuracy from Layer 1 (0.391) to Layer 2 (0.419) represents a substantial gain, suggesting that additional layers contribute to more refined feature extraction and better model performance. However, similar to the effect observed with query tokens, the improvement diminishes with each additional layer. For instance, the increase from Layer 4 (0.433) to Layer 5 (0.435) is only 0.002, indicating that after a certain depth, the model reaches a

| Modifications | Accuracy | | |
|---|---|---|---|
| | Variant1 | Variant2 | Variant3 |
| Original Sequence Length | 64 | 256 | 256 |
| Compressed Sequence Length | ✗ | ✗ | ✓ |
| Accuracy | 0.417 | 0.441 | 0.445 |

**Table 3: Comparison of Model Performance with Different Sequence Lengths and Compression**

point of diminishing returns, where adding more layers provides marginal benefits.

This pattern suggests that while deeper models generally perform better, there is a trade-off between complexity and performance. Beyond a certain depth (around Layer 4 or 5 in this case), the accuracy gains become marginal, and additional layers may not justify the increased computational cost. Thus, in this context, Layer 4 or 5 appears to be the optimal depth, balancing accuracy with efficiency.

*4.4.4 Item compression strategy impact.* Table 3 compares the performance of models with varying sequence lengths and compression strategies. Specifically, we explored the impact of compressing item sequences on model accuracy.

For the compression approach, we devised a method where a sequence of 256 items is progressively compressed with compression window sizes of 64, 64, 16, 16, 16, 16, and 16 items, starting from the oldest to the most recent. The latest 48 items are appended to this sequence without compression, and the resulting sequence is then fed into the model. The query token count is fixed at 1, and the model architecture comprises a 4-layer transformer.

Our findings indicate that the compressed sequence outperforms the model with a sequence length of 64 and, notably, achieves slightly better accuracy than the model with the full uncompressed sequence of 256 items. This suggests that the compression strategy effectively reduces noise in the sequence, contributing to improved model performance.

## 5 RELATED WORKS

### 5.1 Sequential Recommendation

In recommender systems, the recommendation process is typically divided into two key stages: retrieval and ranking. The retrieval stage aims to efficiently retrieve a subset of items from a large candidate pool that a user might be interested in. Due to computational constraints, retrieval models are often designed with lighter architectures, frequently employing dual-tower structures where separate towers model users and items, interacting only at the top layer [6, 11]. In contrast, the ranking stage scores the retrieved subset, allowing for the application of more complex models and intricate feature interactions [5, 26, 37, 38]. The focus of this work is to design more effective retrieval models.

Historically, collaborative filtering methods have been widely used in recommendation tasks, including item-based CF, user-based CF, and factorization machines [10, 16, 17]. With advancements in deep learning, models such as DNN and Embedding-Based Retrieval have become prevalent [6, 11, 12, 20]. In the realm of recommendation, leveraging users' historical behavior is crucial. Predicting the next item a user might engage with based on their behavior sequence is known as sequential modeling [22, 28, 33].

Several works have employed Transformer architectures for sequential modeling. For instance, SASRec models the recommendation task as an autoregressive problem, predicting the next item in a sequence based on previous interactions [14]. In contrast, BERT4Rec utilizes bidirectional self-attention mechanisms to capture both past and future contexts within a sequence, enhancing the model's understanding of user behavior [29]. PinnerFormer, implemented in real-world industrial scenarios, leverages Transformer architectures to capture users' long-term interests for recommendation purposes [25].

These Transformer-based models have demonstrated significant improvements in capturing complex user behavior patterns, thereby enhancing the effectiveness of recommender systems.

### 5.2 Multi-Interest User Representation

In recommendation systems, users often exhibit diverse interests, necessitating the provision of varied content to meet their multifaceted preferences. Traditional models typically represent a user's interest with a single vector, which may not adequately capture the complexity of user behavior. To address this limitation, several approaches have been developed to model multiple user interests, particularly during the retrieval phase. The MIND model assigns a fixed number of vectors to each user, representing preferences for different content types [18]. It employs a dynamic routing mechanism from capsule networks to extract multiple interest representations from user behavior sequences, thereby capturing diverse user interests more effectively. Building upon MIND, the ComiRec model explores methods to integrate multi-interest retrieval results [4]. ComiRec introduces a controllable aggregation module designed to balance recommendation accuracy and diversity. Additionally, it incorporates multi-interest extraction modules based on dynamic routing and self-attention mechanisms to better capture users' multiple interests. The MIP utilizes a time-aware self-attention mechanism to extract multiple interest representations from user behavior sequences [27]. By incorporating temporal information, MIP more accurately captures the dynamic evolution of user interests, thereby enhancing recommendation performance.

## 6 CONCLUSION

KuaiFormer demonstrates the feasibility and effectiveness of a Transformer-based architecture for large-scale retrieval tasks in a short-video recommendation system. By leveraging multi-interest extraction, adaptive sequence compression, and stable training techniques, KuaiFormer effectively captures the complex, dynamic interests of users and scales efficiently across billions of requests. Our extensive offline and online evaluations confirm significant improvements in both offline hit rate and online ab test, validating KuaiFormer's ability to enhance user satisfaction and business performance. This work underscores the transformative potential of advanced neural architectures in industrial recommendation systems, offering a scalable framework that can inspire further innovations in content retrieval and recommendation.

## REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report.

[2] Dosovitskiy Alexey. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. In *arXiv*.

[3] Tom B Brown. 2020. Language models are few-shot learners. In *arXiv*.

[4] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable multi-interest framework for recommendation. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 2942–2951.

[5] Jianxin Chang, Chenbin Zhang, Yiqun Hui, Dewei Leng, Yanan Niu, Yang Song, and Kun Gai. 2023. Pepnet: Parameter and embedding personalized network for infusing with personalized prior information. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *ACM Conference on Recommender Systems (RecSys)*.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.).

[8] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*. PMLR, 3887–3896.

[9] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[10] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *ACM conference on Computer supported cooperative work*.

[11] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

[12] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *ACM International Conference on Information and Knowledge Management (CIKM)*.

[13] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. In *IEEE Transactions on Big Data*.

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

[15] Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. 2022. On The Computational Complexity of Self-Attention. In *arXiv*.

[16] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. 2022. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics* (2022).

[17] Yehuda Koren, Steffen Rendle, and Robert Bell. 2021. Advances in collaborative filtering. *Recommender systems handbook* (2021).

[18] 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 *ACM International Conference on Information and Knowledge Management (CIKM)*.

[19] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*.

[20] Sen Li, Fuyu Lv, Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng, Xiao-Ming Wu, and Qianli Ma. 2021. Embedding-based product retrieval in taobao search. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

[21] Wuchao Li, Rui Huang, Haijun Zhao, Chi Liu, Kai Zheng, Qi Liu, Na Mou, Guorui Zhou, Defu Lian, Yang Song, et al. 2024. DimeRec: A Unified Framework for Enhanced Sequential Recommendation via Generative Diffusion Models. In *arXiv*.

[22] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-aware sequential recommendation. In *IEEE international conference on data mining (ICDM)*.

[23] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10012–10022.

[24] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. 2019. When Does Label Smoothing Help?. In *arXiv*.

[25] Nikil Pancha, Andrew Zhai, Jure Leskovec, and Charles Rosenberg. 2022. PinnerFormer: Sequence Modeling for User Representation at Pinterest. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

[26] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on long sequential user behavior modeling for click-through rate prediction. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

[27] Hui Shi, Yupeng Gu, Yitong Zhou, Bo Zhao, Sicun Gao, and Jishen Zhao. 2023. Everyone's preference changes differently: A weighted multi-interest model for retrieval. In *International Conference on Machine Learning*. 31228–31242.

[28] Elena Smirnova and Flavian Vasile. 2017. Contextual sequence modeling for recommendation with recurrent neural networks. In *Proceedings of the 2nd workshop on deep learning for recommender systems*.

[29] 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 *ACM International Conference on Information and Knowledge Management (CIKM)*.

[30] Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. 2024. Scaling laws with vocabulary: Larger models deserve larger vocabularies. In *arXiv*.

[31] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. In *arXiv*.

[32] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)* (2017).

[33] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. 2019. Sequential recommender systems: challenges, progress and prospects. *arXiv* (2019).

[34] Xiaoyong Yang, Yadong Zhu, Yi Zhang, Xiaobo Wang, and Quan Yuan. 2020. Large scale product graph construction for recommendation in e-commerce. In *arXiv*.

[35] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Ajit Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. 2019. Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations. In *ACM Conference on Recommender Systems (RecSys)*.

[36] Kai Zheng, Haijun Zhao, Rui Huang, Beichuan Zhang, Na Mou, Yanan Niu, Yang Song, Hongning Wang, and Kun Gai. 2024. Full Stage Learning to Rank: A Unified Framework for Multi-Stage Systems. In *Proceedings of the ACM on Web Conference*.

[37] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*.

[38] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.