

UniSearch: Rethinking Search System with a Unified Generative Architecture

Jiahui Chen[†], Xiaoze Jiang^{*†}, Zhibo Wang[†], Quanzhi Zhu[†], Junyao Zhao[†], Feng Hu[†], Kang Pan,
 Ao Xie, Maohua Pei, Zhiheng Qin, Hongjing Zhang, Zhixin Zhai, Xiaobo Guo, Runbin Zhou,
 Kefeng Wang, Mingyang Geng, Cheng Chen, Jingshan Lv, Yupeng Huang, Xiao Liang, Han Li
 Kuaishou Technology, Beijing, China

{chenjiahui11,jiangxiaoze,wangzhibo07,zhuquanzhi03,zhaojunyao,hufeng}@kuaishou.com

Abstract

Modern search systems play a crucial role in facilitating information acquisition. Traditional search engines typically rely on a cascaded architecture, where results are retrieved through recall, pre-ranking, and ranking stages. The complexity of designing and maintaining multiple modules makes it difficult to achieve holistic performance gains. Recent advances in generative recommendation have motivated the exploration of unified generative search as an alternative. However, existing approaches are not genuinely end-to-end: they typically train an item encoder to tokenize candidates first and then optimize a generator separately, leading to objective inconsistency and limited generalization. To address these limitations, we propose UniSearch, a unified generative search framework for Kuaishou Search. UniSearch replaces the cascaded pipeline with an end-to-end architecture that integrates a Search Generator and a Video Encoder. The Generator produces semantic identifiers of relevant items given a user query, while the Video Encoder learns latent item embeddings and provides their tokenized representations. A unified training framework jointly optimizes both components, enabling mutual enhancement and improving representation quality and generation accuracy. Furthermore, we introduce Search Preference Optimization (SPO), which leverages a reward model and real user feedback to better align generation with user preferences. Extensive experiments on industrial-scale datasets, together with online A/B testing in both short-video and live search scenarios, demonstrate the strong effectiveness and deployment potential of UniSearch. Notably, its deployment in live search yields the largest single-experiment improvement in recent years of our product's history, highlighting its practical value for real-world applications.

CCS Concepts

- Information systems → Novelty in information retrieval; Combination, fusion and federated search.

Keywords

Generative Search; Information Retrieval

^{*}Corresponding author. [†]Equal contribution.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

UniSearch, Beijing, China

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN xxxx/2025/09

<https://doi.org/10.1145/xxxxxx>

ACM Reference Format:

Jiahui Chen[†], Xiaoze Jiang^{*†}, Zhibo Wang[†], Quanzhi Zhu[†], Junyao Zhao[†], Feng Hu[†], Kang Pan,[†], Ao Xie, Maohua Pei, Zhiheng Qin, Hongjing Zhang, Zhixin Zhai, Xiaobo Guo, Runbin Zhou,[†], Kefeng Wang, Mingyang Geng, Cheng Chen, Jingshan Lv, Yupeng Huang, Xiao Liang, Han Li. 2025. UniSearch: Rethinking Search System with a Unified Generative Architecture. In *the Preprint Version of UniSearch, Beijing, China*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/xxxxxx>

1 Introduction

Search systems are fundamental to modern industrial applications and information networks, enabling users to efficiently access relevant content from massive item collections [1, 13]. Over the past two decades, search technology has advanced significantly, evolving from classical inverted indexing to sophisticated deep neural models. Despite these advances, most industrial search engines still adopt a multi-stage cascaded architecture (MCA), which filters candidate items in a coarse-to-fine manner through recall, pre-ranking, and ranking stages [8], as shown in Figure 1 (a).

Although this cascaded design has been refined through years of practice, it suffers from several inherent drawbacks. First, each stage employs distinct models with separate optimization objectives, resulting in misaligned training signals across stages [38]. This discrepancy often prevents optimal end-to-end relevance modeling, limiting the system's upper-bound performance. Second, the multi-stage structure introduces substantial inference latency and high maintenance overhead, as each stage requires dedicated algorithmic design and system support [2]. These limitations motivate the search for *unified generative solutions* that can replace the cascaded pipeline with a single end-to-end framework.

The rapid progress of large-scale generative models in recent years has made this direction increasingly viable [4, 14, 23, 26, 30]. In recommendation systems, several works have demonstrated that generative models can replace the entire cascaded pipeline with a unified solution, achieving remarkable results [2, 40]. However, most existing approaches are not truly end-to-end, as shown in Figure 1 (b): they typically involve two separate training steps—first training a video encoder to produce embeddings and semantic tokens, followed by training a generator to predict these tokens. This discrepancy between tokenization and generation objectives leads to inconsistent optimization and suboptimal generation quality. More importantly, extending generative models to search poses unique challenges: unlike recommendation, which remains an intra-domain task (mapping user/item embeddings to items), search requires *cross-domain generation* from text queries directly to items. Existing studies have only explored generative models in

early stages such as recall [15, 31], and the exploration of unified generative methods as a replacement for the full cascaded search architecture remains underexplored.

To address these challenges, we propose UniSearch, an end-to-end generative search framework deployed at industrial scale. As illustrated in Figure 1 (c), UniSearch adopts a unified architecture comprising a *Search Generator* and a *Video Encoder*. The Generator follows an encoder-decoder paradigm, taking query text and user features as input and autoregressively generating the semantic identifiers (SID) of relevant items. The Video Encoder, in turn, learns latent embeddings from multi-modal signals (e.g., textual, visual, and statistical data), and discretizes them into semantic IDs using VQ-VAE [24], thereby constructing a compact representation space that bridges queries and items within a shared generative framework. Built upon this architecture, we introduce a unified pre-training strategy that seamlessly integrates semantic tokenization with autoregressive generation. In contrast to prior work that separates item encoding and query generation into independent objectives [2, 5], UniSearch jointly optimizes both tasks. Specifically, residual contrastive learning, guided by a coarse-to-fine optimization strategy, aligns the semantics between queries and videos. Simultaneously, the discretization process in VQ-VAE produces semantic IDs that are co-optimized with the generative model. This design eliminates the objective mismatch common in previous generative recommendation approaches, enabling mutual reinforcement between video encoding and query generation, and ultimately leading to more robust representations and higher-quality outputs. Beyond supervised pre-training, UniSearch incorporates an online post-training phase for *Search Preference Optimization* (SPO). In a controlled online environment, UniSearch generates candidate results that are evaluated through a reward system combining system-estimated signals (e.g., relevance and content quality) with real user feedback, such as clicks and watch time. Leveraging the SPO algorithm, UniSearch refines its generation policy based on these preference signals, aligning its outputs more closely with user intent. To ensure efficiency and scalability in real-world deployment, we employ a Trie-based inference mechanism that constrains generation to valid semantic ID paths. This not only prevents invalid outputs but also enhances computational efficiency, making UniSearch practical for large-scale applications. In Kuaishou’s production environment, UniSearch is integrated with a reinforcement learning (RL) training system, an online inference service accelerated by TensorRT and KV-cache, a dynamic Trie server, and a reward feedback system—together forming a complete, end-to-end generative search solution.

We extensively evaluate UniSearch on large-scale industrial datasets and real-world online traffic. Offline experiments demonstrate that UniSearch outperforms strong cascaded baselines in both relevance and efficiency. Online A/B testing on Kuaishou’s live and short-video search systems further confirms its effectiveness. The contributions of this paper are summarized as follows:

(1) UniSearch is the **generative search solution deployed at scale**, replacing the traditional cascaded architecture with a unified model that surpasses it in both retrieval quality and computational efficiency.

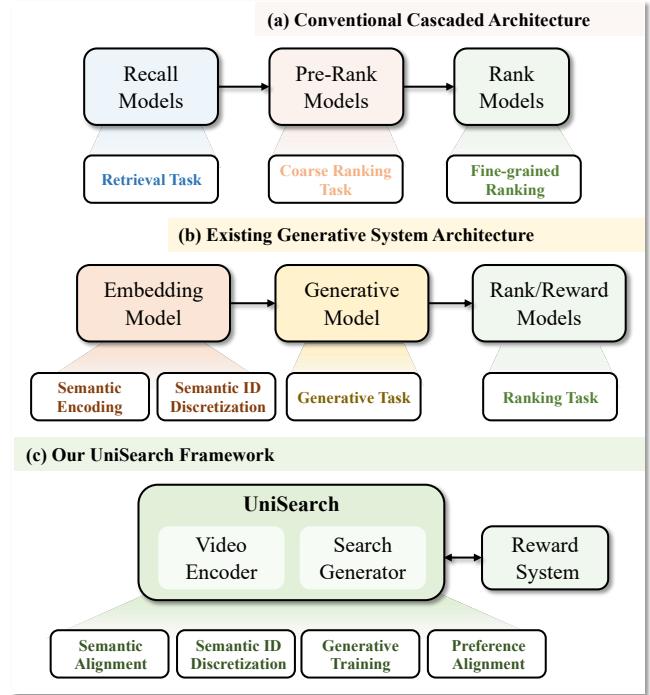


Figure 1: A schematic illustration of existing search architecture and our UniSearch. (a) Conventional Cascaded Architecture, (b) Existing Generative System Architecture and (c) Our UniSearch Framework.

(2) We propose a **unified training framework** that jointly integrates tokenization and generation tasks, leading to more effective item encoding and higher-quality generation compared with conventional multi-stage training.

(3) We successfully deploy UniSearch in Kuaishou’s live and short-video search systems. Large-scale online A/B experiments verify its practical effectiveness and efficiency, serving hundreds of millions of active users.

2 Related Works

2.1 Search System

Traditional search systems are typically built upon a cascaded pipeline architecture, including recall, pre-ranking and ranking. Within this framework, extensive research has been devoted to retrieving query-relevant content across different stages of the pipeline [8, 28, 29, 33, 36]. Modern search systems adopt BERT-based [3] discriminative models across different stages to better capture relevance, with distinct architectures and training objectives tailored to each phase. In the recall stage, a dual-tower architecture combined with contrastive learning is commonly employed to enhance the model’s semantic retrieval capability. For pre-ranking, a dual-tower model with post-interaction is used to efficiently filter the large set of candidates returned by the recall stage. Finally, the fine-ranking module adopts a single-tower, fully interactive architecture, where point-wise learning is applied to accurately estimate

query-document relevance, with a strong emphasis on model precision. This requires sustained and significant algorithmic design and investment. However, the objectives of different components often diverge, potentially leading to suboptimal overall performance. In contrast to prior methods, we use a single generative model to replace the traditional multi-stage cascade, unifying objectives and improving end-to-end consistency.

2.2 Generative Retrieval and OneModels

Recently, autoregressive generation-based Large Language Models (LLMs) [18] have exhibited remarkable capabilities across multiple domains, such as recommendation [2, 6, 7, 32, 35, 39, 40]. This significant breakthrough has led to the emergence of a new paradigm in retrieval [10–12, 15, 17, 19–22, 26, 34, 37] - the shift from matching-based to generation-based approaches, known as generative retrieval. Generative retrieval represents a recent paradigm shift in information retrieval (IR), where retrieval is formulated as a text generation task [11, 12]. Instead of retrieving documents by scoring candidates, a generative retriever directly generates document identifiers, passages, or even answers conditioned on the input query [21, 26]. In the field of text-to-text information retrieval, such as query auto-completion, recent approaches directly generate the corresponding answer conditioned on the input text [17, 26, 34]. Compared to dual-encoder (bi-encoder) models, this generative paradigm breaks the limitations of traditional indexing and has achieved significant improvements in performance metrics. Other works in the recommendation domain typically encode documents into discrete paths, which are then used to generate candidates as a novel source for retrieval [15, 19, 20, 22, 37]. The above-mentioned studies focus on applying generative models within the retrieval component, while neglecting their potential to substitute the entire cascaded architecture through their strong generative capabilities. By contrast, recent efforts have started to use one generative model as a replacement for the traditional pipeline, such as OneRec on short-video recommendation [2, 40], OneSug on query auto-completion [5], POI recommendation [25] and local life service [27]. However, many of these methods rely on pre-tokenized candidate items, which can limit generalization and hinder the model’s ability to adapt to unseen or dynamic content. In contrast, our work introduces a unified generative framework that removes the reliance on fixed candidate tokenization, enabling open-ended generation of item identifiers.

3 Methodology

In this section, we first present the overall architecture of UniSearch, followed by its training scheme, which consists of unified pre-training and search preference alignment post-training. Finally, we describe its inference process and deployment.

3.1 UniSearch Architecture

Unlike previous cascaded search systems that rely on multiple models across different stages, UniSearch adopts a unified architecture to perform both training and inference within a single framework. This design eliminates the inconsistencies between stage-specific objectives and reduces system complexity, enabling more robust and efficient end-to-end learning. As illustrated in Figure 2 (a),

UniSearch consists of two main components: a **Search Generator** and a **Video Encoder**.

Search Generator. The Generator follows an encoder-decoder generative paradigm. The encoder is a bidirectional Transformer that processes the user query along with auxiliary user features (e.g., GSU sequences representing personalized historical behaviors). To further capture the holistic semantics of a search request, we prepend a special `<cls>` token to the input sequence. The resulting representation of `<cls>` serves as the global query embedding q , which summarizes the intent of the current search request. The decoder then receives the contextualized query representation from the encoder, and autoregressively generates a sequence of tokens corresponding to the semantic IDs of the relevant videos. This operation effectively transforms a natural language query into a structured semantic representation of search results.

Video Encoder. The Video Encoder is a unidirectional Transformer designed to learn latent embeddings and semantic IDs for each video item. Specifically, each video is represented through its textual metadata, multi-modal content features, and side statistical features. These inputs are concatenated with k learnable tokens and passed through the Transformer, producing a sequence of k latent embeddings: $D = \{d^{(1)}, d^{(2)}, \dots, d^{(k)}\}$, that capture the semantic properties of the video. To discretize these continuous embeddings into semantic identifiers, the Video Encoder is augmented with a VQ-VAE module, which maps the latent embeddings into a codebook space. The resulting semantic ID (SID) sequence is: $S = \{s^{(1)}, s^{(2)}, \dots, s^{(k)}\}$, which provides a compact and interpretable representation of each video item, ensuring that items can be indexed and retrieved consistently in the generative search framework.

By jointly employing the Generator and Video Encoder, UniSearch directly bridges the gap between textual queries and item representations, allowing the entire search process to be handled within a single, coherent framework.

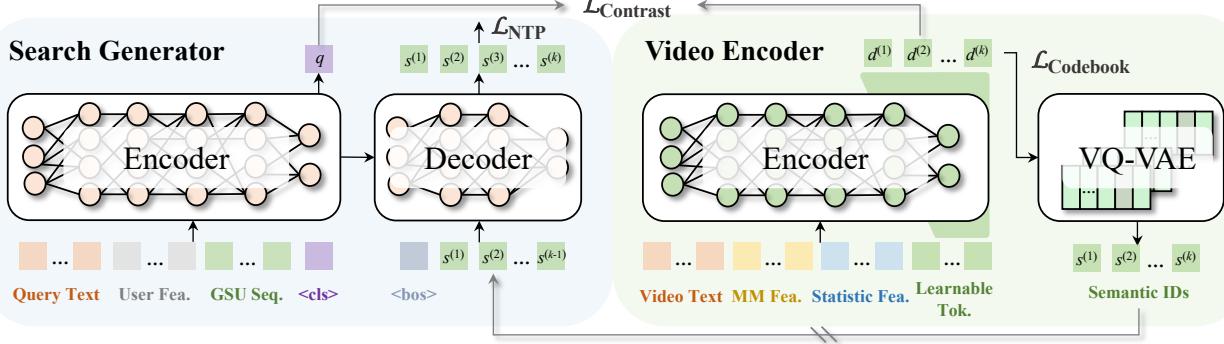
3.2 Unified Pre-training

To enable the Generator and Video Encoder to work cooperatively, we design a **unified pre-training framework** that simultaneously learns video encoding and query-to-video generation. Unlike prior generative recommendation methods that separate item tokenization and generation into two disjoint stages, UniSearch optimizes both objectives in a single training pipeline, ensuring semantic consistency and eliminating task misalignment.

Our pre-training data are sampled from large-scale search logs, where each record contains: query, user features, candidate video set $\{d_1, d_2, \dots, d_N\}$, label set $\{l_1, l_2, \dots, l_N\}$. Here, the candidate video set includes both exposed items and unexposed negative items for the query, while the label set provides graded relevance annotations for each candidate, reflecting both query-video relevance and video quality.

Residual Contrastive Learning for Semantic Alignment. The first objective of UniSearch pre-training is to align the semantics of user queries with the latent representations of relevant items. Let q denote the query embedding produced by the Generator encoder and D_i the latent representation of the i -th candidate video encoded by the Video Encoder. To achieve this, we introduce *residual*

(a) Model Architecture and Unified Pre-training



(b) UniSearch Deployment and Online Post-training

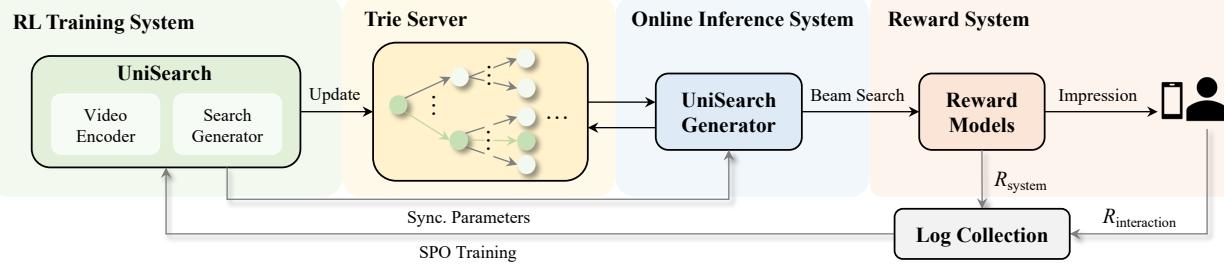


Figure 2: The overall architecture of the UniSearch, where “Seq.” denotes “Sequence”, “Fea.” stands for “Feature”, “Learned Tok.” refers to the learned tokens for documents, and “MM Fea.” represents multi-modal features.

contrastive learning, where each video embedding is represented as a sequence of residual components $d_i^{(n)}$. At step n , the current residual $d_i^{(n)}$ is optimized against the query embedding q_i while accumulating information from all previous residuals, as defined in Equation 1. This residual formulation encourages each token to capture complementary information, thereby reducing redundancy across tokens and making more efficient use of the codebook, while also alleviating the token-collapsing issue common in discrete representation learning.

$$\mathcal{L}_{\text{contrast}} = \sum_{n=1}^k \mathcal{L}\left(q_i, \text{sg}\left[\sum_{m<n} d_i^{(m)}\right] + d_i^{(n)}\right), \quad (1)$$

$$\mathcal{L}(q, d) = -\log \frac{\exp(\text{sim}(q, d)/\tau)}{\exp(\text{sim}(q, d)/\tau) + \sum_{d^- \in \mathcal{N}} \exp(\text{sim}(q, d^-)/\tau)}, \quad (2)$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operation, τ is the temperature parameter, and \mathcal{N} is the negative sample set. In contrast to conventional cosine similarity, we adopt L2 similarity $\text{sim}(q, d) = 1 - \|q - d\|_2^2$, which better fits the residual accumulation mechanism and preserves the geometric structure of embeddings.

Coarse-to-Fine Strategy. To further strengthen semantic alignment, we design a coarse-to-fine training strategy that mimics the

cascaded ranking process in traditional search systems. The intuition is that the first residual token $d^{(1)}$ should learn an easy, coarse-grained matching task, ensuring robust recall capability by distinguishing relevant from irrelevant items at a broad level. Subsequent tokens $d^{(2)}, \dots, d^{(k)}$ progressively take on more complex tasks, refining the representation toward fine-grained ranking and personalization. This staged learning process is achieved by gradually increasing the difficulty of the negative sample set \mathcal{N} : $d^{(1)}$ is trained against simple in-batch negatives, while later residuals are optimized against increasingly hard negatives sampled from semantically similar but irrelevant items.

Compared with residual quantization methods such as RQ-Kmeans that perform two-stage residual clustering, our approach enables end-to-end training, thereby reducing clustering loss and avoiding the mismatch between separate training objectives. The residual formulation not only decomposes the learning task into a sequence of increasingly difficult subtasks but also prevents token redundancy, making more effective use of the codebook. As a result, UniSearch simultaneously captures strong recall ability and refined ranking capacity within a single unified framework, achieving better overall search performance than approaches that optimize only for recall. **Discretization into Semantic IDs.** While semantic alignment ensures meaningful item representations, the Video Encoder must

also discretize latent embeddings into semantic IDs for generative modeling. We employ VQ-VAE [24] for this purpose. Unlike prior works that rely on post-clustering methods such as balanced K-means or RQ-Kmeans [2, 5], VQ-VAE updates the codebook jointly during training, enabling efficient online learning and inference. Concretely, for each embedding $d_i^{(n)}$, the VQ-VAE encoder performs nearest-neighbor lookup in the learnable codebook to obtain a quantized embedding $e_i^{(n)}$ and corresponding semantic ID $s_i^{(n)}$. The quantized embeddings $E_i = \{e_i^{(1)}, e_i^{(2)}, \dots, e_i^{(k)}\}$ are further required to reconstruct the original embeddings D_i . Codebook loss is introduced to ensure $e_i^{(n)}$ approximates d_i , as shown in Equation 3:

$$\mathcal{L}_{\text{codebook}} = \sum_{n=1}^k \alpha_1 \|\text{sg}[d_i^{(n)}] - e_i^{(n)}\|_2^2 + \alpha_2 \|d_i^{(n)} - \text{sg}[e_i^{(n)}]\|_2^2. \quad (3)$$

Specifically, we employ the SimVQ strategy [41] to further stabilize the discretization process and avoid codebook collapse. With the VQ module, videos can be discretized into semantic IDs in a fully end-to-end manner, without the inconsistencies of offline clustering.

Reject-Sampling Generative Training. Once items are mapped into semantic IDs, the Generator is trained to produce these sequences autoregressively using a next-token prediction objective with cross-entropy loss. To enhance generation quality, we employ reject-sampling strategies during training: low-quality items (as determined by labels) are filtered out, and losses for items of different levels are reweighted accordingly, as shown in Equation 4:

$$\mathcal{L}_{\text{NTP}} = -w_i \sum_{n=1}^k \log p(s_i^{(n)} | q, u, s_i^{(<n)}) , \quad (4)$$

where w_i is a re-weight factor determined by label l_i . This ensures that the Generator prioritizes high-relevance, high-quality items when producing search results.

Overall. Through the above unified pre-training objectives, UniSearch achieves joint optimization of query–item semantic alignment, item discretization, and high-quality sequence generation. This provides a consistent and robust foundation for subsequent post-training and inference. The overall training loss is as follows:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{contrast}} + \lambda_2 \mathcal{L}_{\text{codebook}} + \lambda_3 \mathcal{L}_{\text{NTP}}, \quad (5)$$

where $\lambda_{1,2,3}$ are hyper-parameters to balance the loss scale and stabilize training process.

3.3 Post-training with SPO

After pre-training, UniSearch is already capable of generating usable search results. However, to further enhance the quality of generated results and better align them with user preferences, we introduce an online reinforcement learning framework termed Search Preference Alignment. This post-training stage is conducted in a controlled online environment, as illustrated in Figure 2 (b).

Specifically, the deployed UniSearch model (see Section 3.4 for details) generates N candidate result sequences using Beam Search. These candidates are passed to a Reward System, which assigns each result a reward score based on multiple factors such as relevance and video quality. The top- M results are then exposed to users. Subsequently, users' real interaction behaviors (e.g., clicks, likes, or downloads) are also collected and incorporated as an additional reward signal. The combined reward scores are used to update

UniSearch through the Search Preference Optimization (SPO) algorithm, thereby aligning the model's output distribution with user preferences in an online, data-driven manner.

Reward System. Modern industrial search engines have undergone years of refinement, particularly in ranking modeling. To leverage this, we adopt the existing fine-ranking module of the production search system as the Reward System. This module takes as input the query, user context, and candidate video features, and outputs multiple predictive scores covering dimensions such as query–item relevance, video quality, and expected user satisfaction. These scores constitute part of the reward signal. In addition, once the videos are presented to the user, explicit interaction feedback (e.g., click-through, watch time, likes, or downloads) provides another intuitive measure of search quality. The final reward score is a weighted combination of system-estimated scores and observed user interactions:

$$R = \gamma_1 R_{\text{system}} + \gamma_2 R_{\text{interaction}}, \quad (6)$$

where $\gamma_{1,2}$ balance the relative importance of system-predicted and user-behavioral signals.

Search Preference Optimization. From the online system logs, we obtain G generated search results under a given query, denoted as $\{S_1, S_2, \dots, S_G\}$, together with their corresponding reward scores $\{R_1, R_2, \dots, R_G\}$. For each candidate $S_i = \{s_i^{(1)}, s_i^{(2)}, \dots, s_i^{(k)}\}$, UniSearch assigns a generation probability $p_i = \pi_\theta(S_i | q, u)$ conditioned on the query q and user context u . Following GRPO [14], we first compute the relative advantage of each candidate by normalizing its reward against the expected reward of the batch:

$$A_i = \frac{R_i - \text{mean}(\{R_1, R_2, \dots, R_G\})}{\text{std}(\{R_1, R_2, \dots, R_G\})}. \quad (7)$$

This formulation encourages the model to prioritize results that are better than the batch average while suppressing worse ones. The final optimization objective is then defined as:

$$\begin{aligned} \mathcal{L}_{\text{SPO}}(\theta) = & -\frac{1}{G} \sum_{i=1}^G \frac{1}{k} \sum_{n=1}^k \left(\frac{\pi_\theta(s_i^{(n)} | q, u, s_i^{(<n)})}{\pi_{\theta_{\text{no_grad}}}(s_i^{(n)} | q, u, s_i^{(<n)})} A_i \right. \\ & \left. - \beta \mathbb{D}_{\text{KL}}[\pi_\theta || \pi_{\text{ref}}] \right), \end{aligned} \quad (8)$$

where the first term maximizes the likelihood of preference-aligned generations weighted by their relative advantages, and the second term regularizes the updated policy π_θ against the reference policy π_{ref} obtained from pre-training. The coefficient β controls the strength of the KL regularization.

This formulation enables UniSearch to align its generation behavior with both system-estimated and user-observed preferences, while maintaining stability and avoiding catastrophic deviation from the pre-trained model.

3.4 Inference and Deployment

To improve efficiency during inference, we constrain the generation of UniSearch with a prefix tree (Trie) structure that encodes all valid semantic ID sequences. A dedicated Trie service maintains the dictionary of permissible paths, ensuring that UniSearch only generates valid sequences. At each decoding step, UniSearch queries the Trie service to retrieve feasible continuation tokens, thereby eliminating invalid semantic IDs and significantly reducing

computational overhead. To further scale to large video corpora, the Trie is optimized by combining prefix-tree search with binary search, which effectively alleviates memory overhead under a massive semantic ID space.

UniSearch has been successfully deployed in real-world industrial search scenarios. As illustrated in Figure 2 (b), the deployment architecture integrates four key components into a unified system. First, a reinforcement learning (RL) training system continuously fine-tunes UniSearch with online preference alignment and synchronizes updated parameters to the inference service. Second, an online inference system serves real-time search requests, where models are deployed using TensorRT and accelerated by key-value (KV) caching to ensure low-latency responses. Third, the Trie server provides efficient constrained generation by maintaining valid semantic ID paths, thereby guaranteeing both accuracy and scalability. Finally, a reward system evaluates candidate results through fine-grained ranking scores and user interaction signals, feeding back preference-aligned rewards to guide further training.

Through this end-to-end design, UniSearch achieves scalable and low-latency inference while maintaining the ability to continuously improve search quality via online learning and preference optimization.

4 Experiments

In this section, we present the datasets, evaluation metrics, implementation details, and both offline and online experimental analyzes of UniSearch.

4.1 Implementation Details

Datasets. To evaluate the effectiveness of UniSearch, we construct large-scale training and testing datasets based on real search logs from the Kuaishou App. For live search, we collect approximately 80 million user search sessions spanning the period from July 9 to July 23, 2025. Data from July 9 to July 22 are used for training, while logs from July 23 form the held-out test set. All offline experiments reported in Sections 4.2 and 4.3 are conducted on this live dataset. Following the same procedure, we also construct a dataset for short-video search, which is used in Section 4.4 to validate the generalization of UniSearch beyond live scenarios.

Details of the UniSearch. The UniSearch framework consists of a Search Generator and a Video Encoder, instantiated using BART [9] and BERT [3] architectures, respectively. For the live search task, where the candidate pool contains around 500K live sessions, we employ lightweight 6-layer BART and BERT models with a hidden size of 768. The item encoder discretizes video embeddings into semantic IDs using a three-level codebook ($k = 3$), with each level containing 512 entries. For the short-video search task, the candidate pool scales up to roughly one billion videos, which requires a larger model to handle the substantially increased complexity. In this setting, we adopt a 12-layer UniSearch configuration with a codebook of three levels and 8192 entries per level. We use the Adam optimizer with a batch size of 64. During pre-training, the initial learning rate is set to 2×10^{-4} and scheduled with a cosine decay strategy. For online post-training, we use a fixed learning rate of 1×10^{-5} . During inference, UniSearch employs beam search with a beam size of 256 to efficiently generate high-quality semantic

IDs. This setup balances computational efficiency with the ability to explore diverse candidate items, ensuring robust retrieval performance under both live-streaming and short-video search scenarios.

Evaluation Metrics. To evaluate the performance of our model, we adopt Recall@300 and Mean Reciprocal Rank (MRR) as evaluation metrics. Recall@300 assesses the model’s ability to retrieve relevant results, while MRR reflects its ranking ability. For Recall@300, let T_i denote the number of true positive instances among the top 300 retrieved results (*i.e.*, instances that are both predicted and labeled as positive), let P_i denote the total number of positive instances in the dataset, and let N represent the total number of samples:

$$\text{Recall}@300 = \frac{1}{N} \sum_{i=1}^N \frac{T_i}{P_i}. \quad (9)$$

In the MRR metric, K_i denotes the position of the ground-truth result for the i -th sample in the ranked list of candidates:

$$\text{MRR} = \frac{1}{N} \sum_{i=1}^N \frac{1}{K_i}. \quad (10)$$

We conduct experiments on both the *ranking subset* (RK) and *click subset* (CK). The RK test set includes the top videos composed to users by the search system, used to evaluate the alignment between generated videos and system preferences. The CK test set treats clicked items as positives, demonstrating immediate user preferences.

4.2 Main Results

To verify the effectiveness of UniSearch, we conduct extensive experiments on the live search dataset and compare it with existing generative search frameworks. All experiments are performed under the same training and testing conditions to ensure a fair comparison.

Baselines. As introduced in the Introduction, existing generative search methods are not fully end-to-end. They typically involve a two-stage pipeline: (1) training an embedding model and constructing a codebook using methods such as VQ-VAE [24], FSQ [16], or RQ-Kmeans; and (2) training a generative model on the tokenized items. We reproduce these approaches as baselines, and the results are reported in Table 1 (the first five rows). Although such frameworks achieve reasonable performance (*e.g.*, a 6-layer baseline with RQ-Kmeans reaches an MRR of 14.81 on the CK test set), their reliance on multiple models and inconsistent training objectives often leads to suboptimal results.

Advantages of Unified Pre-training. In contrast, UniSearch unifies feature representation learning, codebook construction, and generative modeling within a single pre-training framework. This joint optimization eliminates the inconsistency across objectives and enables stronger query understanding and higher-quality generation. As shown in Table 1, UniSearch consistently outperforms baseline methods in offline evaluations. Under the same model scale, UniSearch-6 Layer significantly surpasses the 6-layer baselines in terms of MRR and achieves Recall@300 performance close to that of 12-layer baselines. This demonstrates that a unified objective enables the model to generate more relevant results while improving overall generation quality. Furthermore, UniSearch exhibits strong scalability: both UniSearch-12 Layer and UniSearch-24

Table 1: Performance comparison of different model on live-streaming search dataset.

	Model		Recall@300(%)↑		MRR(%)↑		
	Embed. Model	Codebook	Gen. Model	RK	CK	RK	CK
Existing Generative Search Arch.	BERT-6 Layer	FSQ	55.42	62.26	8.36	9.93	
		VQ-VAE	64.83	63.67	7.98	9.50	
		RQ-Kmeans	65.56	64.12	9.24	14.81	
	BERT-12 Layer	RQ-Kmeans	BART-12 Layer	67.79	67.90	12.13	15.68
	BERT-24 Layer		BART-24 Layer	69.05	69.27	13.84	16.80
	UniSearch- 6 Layer		67.45	68.17	9.65	15.36	
UniSearch (Ours)	UniSearch-12 Layer		68.83	70.11	12.35	15.73	
	UniSearch-24 Layer		69.78	72.03	14.62	16.02	
	UniSearch- 6 Layer (w/ Online SPO)		68.29	68.48	12.19	16.04	
	UniSearch-12 Layer (w/ Online SPO)		69.53	70.25	13.56	16.97	

Layer continue to achieve substantial improvements over baselines of the same scale, highlighting the framework’s robustness at larger capacities.

Effect of Search Preference Optimization (SPO). We further enhance UniSearch with Search Preference Optimization (SPO), applying it to both 6-layer and 12-layer models. The results are presented in the last two rows of Table 1. With SPO, UniSearch achieves moderate gains in Recall@300 but exhibits substantial improvements in MRR. This indicates that SPO effectively guides the model toward generating results that are not only relevant but also of higher quality and better aligned with user preferences. In particular, SPO encourages the model to prioritize items that are more likely to be favored by users, such as popular or high-quality videos, thereby improving the overall user experience.

4.3 Ablation Study

We conduct ablation studies to evaluate the impact of several core design choices in UniSearch, focusing on the training scheme, codebook configuration, and inference strategy.

Effectiveness of Coarse-to-Fine Strategy and Residual Contrastive Learning. To disentangle the contributions of unified pre-training, we compare three ablated variants of UniSearch, with results summarized in Table 2. The analysis is as follows.

(1) *UniSearch-Plain* treats each token independently and optimizes them toward the same objective. This setup weakens token-level diversity, which ultimately yields the weakest performance on both Recall@300 and MRR. (2) *UniSearch-Plain w/ CF* augments the model with a coarse-to-fine (CF) task. By gradually learning from simpler to more complex objectives, the model gains stronger ranking capability and achieves noticeable improvements in recall and accuracy. However, because the coarse and fine tasks are inherently correlated, the resulting token representations often become similar. This may cause the *path collapsing problem*, where multiple candidate items correspond to a single decoding path, thereby reducing discriminative power and harming precision. (3) *UniSearch-Plain w/ RCL* incorporates residual contrastive learning (RCL). Instead of

Table 2: Analysis results of Coarse-to-Fine Strategy and Residual Contrastive Learning, where CF denotes Coarse-to-Fine, RCL denotes Residual Contrastive Learning, and UniSearch-Plain refers to UniSearch without CF and RCL.

Method	Recall@300(%)↑		MRR(%)↑	
	RK	CK	RK	CK
UniSearch-Plain	63.54	63.15	6.55	12.16
UniSearch-Plain w/ CF	64.63	65.61	7.01	12.97
UniSearch-Plain w/ RCL	66.18	65.41	8.59	14.71
UniSearch	67.45	68.17	9.65	15.36

optimizing toward the final objective in a single step, RCL progressively refines representations through residual summation. This encourages different tokens to capture complementary semantic aspects, mitigating path collapse and substantially boosting the ranking metric MRR.

Our full UniSearch model integrates both CF and RCL. The CF strategy provides a structured learning path from easy to difficult objectives, while the residual design preserves token diversity and prevents collapse. Together, these components enable more effective end-to-end discretization, reduce clustering loss, and ultimately yield superior overall performance compared to any single variant. **Ablation on Codebook Depth and Size.** Codebook depth and size are critical hyperparameters in candidate tokenization, directly influencing retrieval quality and system efficiency. We systematically analyze their effects as follows.

(1) To study the impact of depth, we fix the codebook size at 512. Intuitively, deeper codebooks should better capture semantic information by generating longer token sequences. However, as shown in Figure 3(a), performance saturates once the depth k exceeds 3, with both Recall@300 and MRR showing diminishing returns. A plausible explanation is that deeper codebooks increase dispersion in the candidate space, thereby weakening generalization. In addition, greater depth inevitably prolongs decoding and increases

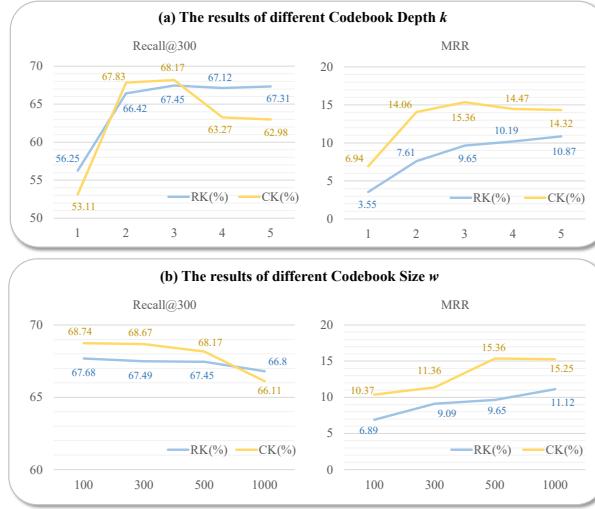


Figure 3: The results of different Codebook Depth and Codebook Size on RK and CK test set. (a) The results of different Codebook Depth k , and (b) The results of different Codebook Size w .

inference latency. Considering both accuracy and efficiency, we adopt a depth of $k = 3$. (2) Next, we fix the depth at $k = 3$ and vary the size of each codebook layer. The results in Figure 3(b) reveal divergent trends: Recall@300 decreases as size grows, while MRR steadily improves. Further analysis suggests that larger codebooks encourage finer semantic partitioning, which enhances ranking precision but reduces recall by fragmenting candidate coverage. Since practical systems require a balance between recall and precision, we set the codebook size to $w = 512$, which achieves strong performance across both metrics.

Impact of the Trie Constraint. The validity of generated paths (*i.e.*, whether the paths correspond to actual candidate items) is critical for system reliability. Without constraints, invalid generations may lead to empty result sets, posing significant risks to search availability. To mitigate this, we incorporate a Trie-based constraint during inference.

As shown in Figure 4, the introduction of the Trie markedly improves both Recall@300 and MRR, while path validity increases from 51.3% to 99.8%. This ensures that nearly all generated results map to valid candidates, enhancing both reliability and computational efficiency. In live search scenarios, dynamic updates to the Trie are also essential: as live streams start or end, the set of valid paths evolves, and even for the same stream, changing content can alter its valid identifiers. Maintaining an up-to-date Trie is thus crucial to guaranteeing robustness in real-world deployments.

4.4 Online Testing

We further evaluate UniSearch in real-world settings by deploying it to Kuaishou’s live-streaming and short-video search systems, and conduct online A/B tests against the legacy production system.

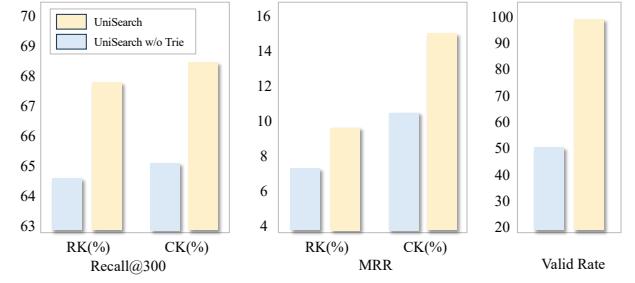


Figure 4: The results of incorporating the Trie on RK and CK test set, and Valid Rates of path to candidates.

Table 3: The improvements of UniSearch in online A/B test compared to the production baseline.

	TPC \uparrow	CTR \uparrow	CQR \downarrow	FCP \downarrow
UniSearch - Live	+3.31%	+0.202%	-0.382%	-0.107%
	VPD \uparrow	PVD \uparrow	CQR \downarrow	LPC \uparrow
UniSearch - Video	+0.213%	+0.993%	-0.602%	+0.830%

Online A/B Testing Setup. We conduct a 7-day online A/B test, where both the experimental and control groups are randomly assigned 10% of actual search traffic. For the live-streaming search scenario, we directly replace the entire Multi-stage Cascading Architecture (MCA) with UniSearch, thereby providing a clean comparison between the unified generative framework and the traditional multi-stage pipeline. In contrast, the short-video search scenario carries the highest traffic volume across Kuaishou. To ensure stability and reliability during deployment, we adopt a progressive integration strategy: UniSearch is first introduced as a new resource in the ranking stage, while several recall sources are gradually deactivated. This allows us to evaluate UniSearch’s effectiveness under strict production constraints without risking user experience.

Impact on Business Metrics. We focus on business metrics that directly reflect user satisfaction and engagement, and summarize the results in Table 3. *In the live-streaming search scenario*, UniSearch achieves a +3.31% improvement in total play counts (TPC), a +0.202% increase in page click-through rates (CTR), and reductions of -0.382% and -0.107% in change query rates (CQR) and position of first-click (PFC), respectively. Among these, TPC is the most critical metric, as it directly reflects search scale and platform vitality. The +3.31% gain represents the most significant improvement in recent years. The CTR increase indicates stronger user engagement, while decreases in CQR and PFC suggest that UniSearch produces more satisfactory results with fewer reformulations and earlier clicks. *In the short-video search scenario*, UniSearch leads to a +0.213% improvement in video playback duration (VPD), a +0.993% increase in page view depth (PVD), a -0.602% reduction in CQR, and a +0.830% increase in long play count per PV (LPC). This reflects that UniSearch not only satisfies immediate search needs but also encourages users to explore a wider variety of content, leading to more diversified interests.

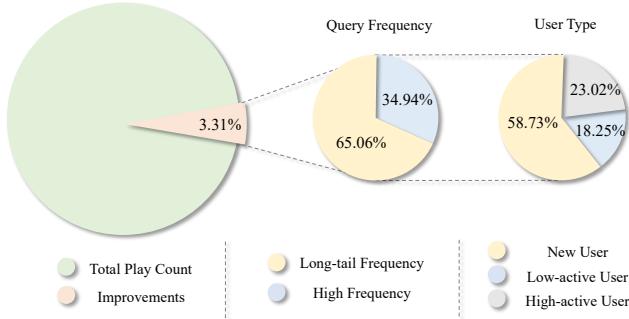


Figure 5: The analysis of the improvements in Total Play Count (TPC) from query frequency and user type.

4.5 Further Analysis

From the online experiments reported in Section 4.4, UniSearch achieves substantial improvements in key business metrics for both live-streaming and short-video search. To better understand the underlying reasons, we perform a detailed analysis of the Live Search scenario, examining the contributions of query frequency, user activity status, and the diversity of generated results.

Enhanced Performance on Long-tail Queries. Given that TPC is the most critical metric for Live Search, we conduct a fine-grained decomposition of its gain. As shown in Figure 5, 65.06% of the TPC improvement is driven by long-tail queries—nearly double the contribution from head queries. This underscores the importance of effectively modeling long-tail traffic in industrial search. Long-tail queries are typically characterized by their sparse occurrence and semantic complexity, making them challenging for traditional systems like MCA to handle. UniSearch, by contrast, benefits from unified pre-training and improved semantic representations, enabling it to capture subtle user intents and generalize more effectively across diverse and low-frequency queries.

Stronger Appeal to New Users. We further segment the TPC gains by user activity status. As illustrated in Figure 5, 58.73% of the increase originates from new users, accounting for more than half of the total gain. This suggests that UniSearch delivers results that are particularly attractive to users who are less familiar with the platform. A plausible explanation is that existing users have already adapted to the behavioral patterns of the legacy MCA system, while new users are more sensitive to improved semantic understanding and richer result sets. This improvement in new-user engagement is especially valuable, as it directly contributes to expanding the active search user base and sustaining long-term growth.

Richer and More Diverse Results. Case studies further validate that UniSearch generates results that are both semantically relevant and diverse. For example, Figure 6 presents the query “MOBA game”. Under MCA, the top three results across users are consistently “Honor of Kings”, showing strong bias toward a single popular item. In contrast, UniSearch surfaces additional relevant games such as “League of Legends” and “Heroes Evolved”. This broader coverage ensures that users with different preferences all find satisfactory content. Without such diversity, users whose interests are not reflected (e.g., those preferring “League of Legends”)

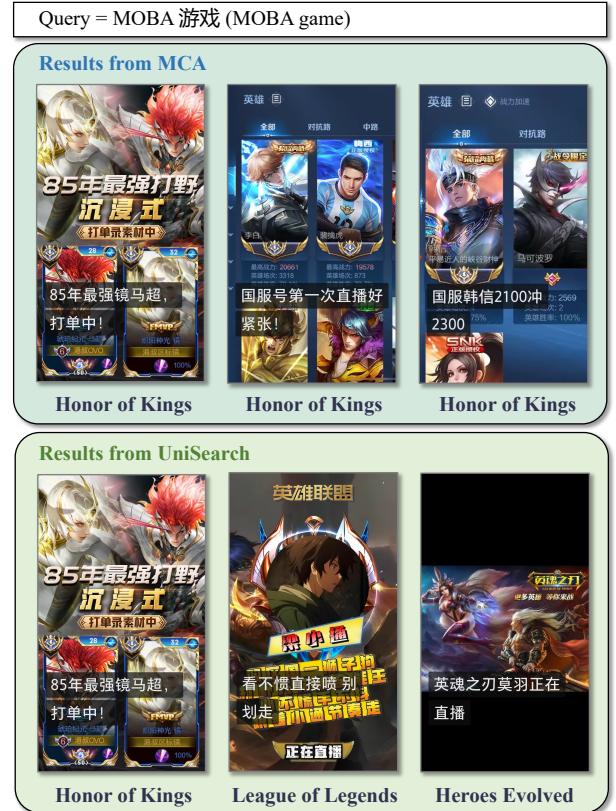


Figure 6: Search results comparison between the Multi-stage Cascading Architecture (MCA) and our UniSearch.

may abandon the platform and reduce future engagement. By balancing relevance with diversity, UniSearch not only improves user satisfaction but also mitigates the risk of content homogenization.

5 Conclusion

We proposed **UniSearch**, a unified generative search framework that replaces the traditional multi-stage cascaded architecture with an end-to-end solution. By jointly optimizing a Search Generator and a Video Encoder, UniSearch integrates tokenization and generation within a single training paradigm, leading to improved representation learning and generation quality. In addition, the proposed *Search Preference Optimization* leverages online reward system and user feedback to align generation with user preferences. Extensive offline experiments and online A/B testing on large-scale industrial datasets demonstrate that UniSearch outperforms strong baselines in both retrieval quality and efficiency. Its deployment in Kuaishou live-streaming search achieved the most significant metric improvement in recent years, highlighting its practical value. **Limitations and Future Work.** Currently, UniSearch generates candidates in a *point-wise* manner using beam search, which may limit result diversity. Future work will focus on enhancing *list-wise generation* to improve both diversity and ranking accuracy, as well as exploring finer-grained reward methods to further strengthen alignment with user preferences.

References

- [1] Yimeng Bai, Shunyu Zhang, Yang Zhang, Hu Liu, Wentian Bao, Enyun Yu, Fuli Feng, and Wenwu Ou. 2025. Unconstrained Monotonic Calibration of Predictions in Deep Ranking Systems. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- [2] Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. 2025. OneRec: Unifying Retrieve and Rank with Generative Recommender and Iterative Preference Alignment. *arXiv preprint arXiv:2502.18965* (2025).
- [3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.
- [4] Luciano Floridi and Massimo Chirietti. 2020. GPT-3: Its Nature, Scope, Limits, and Consequences. *Minds and Machines* 30 (2020), 681–694.
- [5] Xian Guo, Ben Chen, Siyuan Wang, Ying Yang, Chenyi Lei, Yuqing Ding, and Han Li. 2025. OneSug: The Unified End-to-End Generative Framework for E-commerce Query Suggestion. *arXiv preprint arXiv:2506.06913* (2025).
- [6] Ruidong Han, Bin Yin, Shangyu Chen, He Jiang, Fei Jiang, Xiang Li, Chi Ma, Mingcong Huang, Xiaoguang Li, Chunzhen Jing, Yueming Han, Menglei Zhou, Lei Yu, Chuan Liu, and Wei Lin. 2025. MTGR: Industrial-Scale Generative Recommendation Framework in Meituan. *arXiv preprint arXiv:2505.18654* (2025).
- [7] Yanhua Huang, Yuqi Chen, Xiong Cao, Rui Yang, Mingliang Qi, Yinghao Zhu, Qingchang Han, Yaowei Liu, Zhaoyu Liu, Xuefeng Yao, Yuting Jia, Leilei Ma, Yingqi Zhang, Taoyu Zhu, Lijie Zhang, Lei Chen, Weihang Chen, Min Zhu, Ruiwen Xu, and Lei Zhang. 2025. Towards Large-scale Generative Ranking. *arXiv preprint arXiv:2505.04180* (2025).
- [8] Sujay Khandagale, Bhawna Juneja, Prabhat Agarwal, Aditya Subramanian, Jaewon Yang, and Yuting Wang. 2025. InteractRank: Personalized Web-Scale Search Pre-Ranking with Cross Interaction Features. In *Companion Proceedings of the ACM on Web Conference 2025*. 287–295.
- [9] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 7871–7880.
- [10] Mingming Li, Huiwu Wang, Zuxu Chen, Guangtao Nie, Yiming Qiu, Guoyu Tang, Lin Liu, and Jingwei Zhuo. 2024. Generative Retrieval with Preference Optimization for E-commerce Search. *arXiv preprint arXiv:2407.19829* (2024).
- [11] Xiaoxi Li, Jiajie Jin, Yujia Zhou, Yuyao Zhang, Peitian Zhang, Yutao Zhu, and Zhicheng Dou. 2025. From Matching to Generation: A Survey on Generative Information Retrieval. *ACM Transactions on Information Systems* (2025), 1–62.
- [12] Xiaoxi Li, Yujia Zhou, and Zhicheng Dou. 2024. UniGen: A Unified Generative Framework for Retrieval and Question Answering with Large Language Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 8688–8696.
- [13] Juexin Lin, Sachin Yadav, Feng Liu, Nicholas Rossi, Praveen R Suram, Satya Chemboli, Prijith Chandran, Hrushikesh Mohapatra, Tony Lee, Alessandro Magnani, et al. 2024. Enhancing Relevance of Embedding-based Retrieval at Walmart. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 4694–4701.
- [14] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-V3 Technical Report. *arXiv preprint arXiv:2412.19437* (2024).
- [15] Yuxuan Liu, Tianchi Yang, Zihan Zhang, Minghui Song, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. 2024. ASI++: Towards Distributionally Balanced End-to-End Generative Retrieval. *arXiv preprint arXiv:2405.14280* (2024).
- [16] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. 2024. Finite Scalar Quantization: VQ-VAE Made Simple. *The Twelfth International Conference on Learning Representations*.
- [17] Agnès Mustar, Sylvain Lamprier, and Benjamin Piwowarski. 2020. Using BERT and BART for Query Suggestion. In *Joint Conference of the Information Retrieval Communities in Europe*, Vol. 2621.
- [18] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, et al. 2024. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2024).
- [19] Ming Pang, Chunyu Yuan, Xiaoyu He, Zheng Fang, Donghao Xie, Fanyi Qu, Xue Jiang, Changping Peng, Zhangang Lin, Ching Law, and Jingping Shao. 2025. Generative Retrieval and Alignment Model: A New Paradigm for E-commerce Retrieval. *arXiv preprint arXiv:2504.01403* (2025).
- [20] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghuandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichen Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. 2023. Recommender Systems with Generative Retrieval. *Advances in Neural Information Processing Systems* 36 (2023), 10299–10315.
- [21] Yubao Tang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, and Xueqi Cheng. 2024. Listwise Generative Retrieval Models via a Sequential Learning Process. (2024), 31 pages.
- [22] Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. 2022. Transformer Memory as a Differentiable Search Index. *Advances in Neural Information Processing Systems* 35 (2022), 21831–21843.
- [23] 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. *arXiv preprint arXiv:2302.13971* (2023).
- [24] Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural Discrete Representation Learning. *Advances in Neural Information Processing Systems* 30 (2017).
- [25] Dongsheng Wang, Yuxi Huang, Shen Gao, Yifan Wang, Chengrui Huang, and Shuo Wang. 2023. Generative Next POI Recommendation with Semantic ID. *arXiv preprint arXiv:2506.01375* (2025).
- [26] Zhibo Wang, Xiaoze Jiang, Zhiheng Qin, Enyun Yu, and Han Li. 2025. Personalized Query Auto-Completion for Long and Short-Term Interests with Adaptive Detoxification Generation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada*.
- [27] Zhipeng Wei, Kuo Cai, Junda She, Jie Chen, Minghao Chen, Yang Zeng, Qiang Luo, Wengcong Zeng, Ruiming Tang, Kun Gai, and Guorui Zhou. 2025. OneLoc: Geo-Aware Generative Recommender Systems for Local Life Service. *arXiv preprint arXiv:2508.14646* (2025).
- [28] Haoyi Xiong, Jiang Bian, Yuchen Li, Xuhong Li, Mengnan Du, Shuaiqiang Wang, Dawei Yin, and Sumi Helal. 2024. When Search Engine Services meet Large Language Models: Visions and Challenges. *IEEE Transactions on Services Computing* (2024).
- [29] Enqiang Xu, Yiming Qiu, Junyang Bai, Ping Zhang, Dadong Miao, Songlin Wang, Guoyu Tang, Lin Liu, and MingMing Li. 2024. Optimizing E-commerce Search: Toward a Generalizable and Rank-Consistent Pre-Ranking Model. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2875–2879.
- [30] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengan Huang, Chenxu Lv, et al. 2025. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388* (2025).
- [31] Tianchi Yang, Minghui Song, Zihan Zhang, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. 2023. Auto Search Indexer for End-to-End Document Retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 6955–6970.
- [32] Yuhao Yang, Zhi Ji, Zhaopeng Li, Yi Li, Zhonglin Mo, Yue Ding, Kai Chen, Zijian Zhang, Jie Li, Shuanglong Li, and Lin Liu. 2025. Sparse Meets Dense: Unified Generative Recommendations with Cascaded Sparse-Dense Representations. *arXiv preprint arXiv:2503.02453* (2025).
- [33] Cheng Ye. 2024. Exploring a learning-to-rank approach to enhance the Retrieval Augmented Generation (RAG)-based electronic medical records search engines. *Informatics and Health* 1, 2 (2024), 93–99.
- [34] Di Yin, Jiwei Tan, Zhe Zhang, Hongbo Deng, Shujian Huang, and Jiajun Chen. 2020. Learning to Generate Personalized Query Auto-Completions via a Multi-View Multi-Task Attentive Approach. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2998–3007.
- [35] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, Yinghai Lu, and Yu Shi. 2024. Actions Speak Louder than Words: Trillion-Parameter Sequential Transducers for Generative Recommendations. *arXiv preprint arXiv:2402.17152* (2024).
- [36] Shunyu Zhang, Yaobo Liang, Ming Gong, Daxin Jiang, and Nan Duan. 2022. Multi-View Document Representation Learning for Open-Domain Dense Retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 5990–6000.
- [37] Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024. Adapting Large Language Models by Integrating Collaborative Semantics for Recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 1435–1448.
- [38] 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 Web Conference 2024*. 3621–3631.
- [39] Zuowu Zheng, Ze Wang, Fan Yang, Jiangke Fan, Teng Zhang, and Xingxing Wang. 2025. EGA: A Unified End-to-End Generative Framework for Industrial Advertising Systems. *arXiv preprint arXiv:2505.17549* (2025).
- [40] Guorui Zhou, Jiaxin Deng, Jinghao Zhang, Kuo Cai, Lejian Ren, Qiang Luo, Qianqian Wang, Qigen Hu, Rui Huang, Shiyao Wang, et al. 2025. OneRec Technical Report. *arXiv preprint arXiv:2506.13695* (2025).
- [41] Yongxin Zhu, Bocheng Li, Yifei Xin, Zhihua Xia, and Linli Xu. 2024. Addressing Representation Collapse in Vector Quantized Models with One Linear Layer. *arXiv preprint arXiv:2411.02038* (2024).