
Large Language Model as Universal Retriever in Industrial-Scale Recommender System

Junguang Jiang*, **Yanwen Huang***, **Bin Liu**, **Xiaoyu Kong**, **Xinhang Li**,
Ziru Xu, **Han Zhu**, **Jian Xu**, **Bo Zheng[✉]**

Taobao & Tmall Group of Alibaba, China

{jiangjunguang.jjg, huangyanwen.hwy, zhuoli.lb, linlin.kxy, xinhangli.lxh,
ziru.xzr, zhuhan.zh, xiyu.xj, bozheng}@alibaba-inc.com

Abstract

In real-world recommender systems, different retrieval objectives are typically addressed using task-specific datasets with carefully designed model architectures. We demonstrate that Large Language Models (LLMs) can function as universal retrievers, capable of handling multiple objectives within a generative retrieval framework. To model complex user-item relationships within generative retrieval, we propose multi-query representation. To address the challenge of extremely large candidate sets in industrial recommender systems, we introduce matrix decomposition to boost model learnability, discriminability, and transferability, and we incorporate probabilistic sampling to reduce computation costs. Finally, our Universal Retrieval Model (URM) can adaptively generate a set from tens of millions of candidates based on arbitrary given objective while keeping the latency within tens of milliseconds. Applied to industrial-scale data, URM outperforms expert models elaborately designed for different retrieval objectives on offline experiments and significantly improves the core metric of online advertising platform by 3%.

1 Introduction

Recommender systems have become an integral part of people’s daily lives, revolutionizing the way users interact with content and services. To meet the diverse needs of users, recommender systems are evolving to become more sophisticated and multifaceted, leading to the emergence of a wide variety of objectives. For example, due to shifts in user behavior across contexts — such as different apps or interfaces within the same app — multi-scenario objectives are introduced [54]. To accurately capture user intent, objectives like click prediction, purchase prediction, and favorite prediction are formulated [96]. To avoid information cocoons and provide novel options, goals such as serendipity [72; 37] and long-tail item recommendations [50] are established.

Most recommender systems use multi-channel retrieval to address the above issues [27; 28]. For each of the above objectives, a large amount of training data is collected, and specialized retrieval models are then designed, trained, evaluated, and deployed separately. This approach has served well to make progress on narrow objectives, but when the objectives change, it requires collecting new data and training a new retrieval model carefully, which is time-consuming, lacks scalability, and sometimes faces challenges due to insufficient objective-specific data. This pattern has more trouble when dealing with unclear objectives. In industry, a common unresolved issue in retrieval is that for a given online metric, such as advertising revenue, there is often no clear corresponding offline metric [26]. The current strategy for tackling this issue is using human prior knowledge to develop multiple heuristic offline training objectives and then conducting A/B tests to assess their effectiveness.

*Equal contribution.

Another approach is learning multiple retrieval objectives simultaneously, which is known as multi-task learning [7]. However, this will easily lead to *seesaw phenomenon* [68], where performance declines compared to learning objectives separately. Traditionally, objective conditioning is handled at the architectural level, where the parameter for each objective is manually designed according to its data size, and as the number of objectives continues to increase, model design becomes increasingly challenging. However, the emergence of LLMs offers a new approach in which objectives can be defined through text sequences [60; 61; 6]. Previous works [33; 21] indicate that increasing model size in such a structure can effectively reduce the seesaw phenomenon without handcrafted designs. Simultaneously, since text sequences can describe an infinite number of objectives, when retrieval objectives are unclear, we can search within a larger objective space with just one trained model.

Despite the excellent versatility of LLMs, a key challenge to using them in retrieval is how to improve performance and efficiency while retaining versatility. There are two main approaches. The first feeds both user features and item candidates to the LLMs for evaluation, which performs well on academic datasets, but is challenging to scale to billions of items in industrial systems [41; 4; 5]. The second approach feeds user feature sequences to LLMs to generate target items, enhancing computational feasibility within the scope of generative retrieval. Yet generative retrieval faces three inherent challenges: (1) **Model Expressiveness**: While LLMs are highly expressive, generative retrieval struggles to effectively model user-item relationships because they generate target items using a single linear matrix [100; 99; 16]. (2) **Learnability**: In industrial applications, the vast number of item candidates, often in the tens of millions, complicates learning the mapping from LLMs’ hidden representations to the item candidate space due to the expansive parameter scale [62]. (3) **Efficiency**: Both LLM inference and generation within large candidates involve substantial computational costs. By incorporating semantic IDs and generating autoregressively within a reduced space [66; 52; 74; 45; 3], the above issues can be alleviated to some extent. However, the adoption of LLMs in generative retrieval introduces challenges to deployment, particularly due to the significant increase in online computational costs.

In this paper, we introduce the **Universal Retrieval Model (URM)**, a novel pathway towards implementing generative retrieval. URM unifies various retrieval objectives into a cohesive input-output framework and utilizes LLMs as feature generators to enhance versatility, facilitating prompt tuning in industrial recommender systems. By adopting multi-query representation, URM significantly boosts the expressive capacity of generative retrieval models. Matrix decomposition further enhances URM’s learnability, discriminability, and transferability, while probabilistic sampling effectively reduces training and generation costs when faced with an extremely large candidate set. Finally, URM is capable of dynamically adjusting retrieval outputs based on input instructions and efficiently returning collections of items with latency within tens of milliseconds. Our extensive experiments on both public and industrial-scale datasets demonstrate the robust performance and versatility of URM. Furthermore, online A/B tests on our advertising platform confirm the effectiveness of the URM in real-world settings, achieving improvements of over 3% in the key metric.

2 Retrieval in Recommender Systems

In typical recommender systems, the main goal of retrieval is to identify the subset of the candidate set \mathcal{C} that has the highest value according to a specific objective, i.e., $\text{argTop}_{v \in \mathcal{C}} f(u, v)$, where function $f(u, v)$ is an objective function for user u and candidate item v .

Embedding-Based Retrieval (EBR). To handle the large candidate set \mathcal{C} , EBR [26] is commonly used in practice, where the inner product of user and item representations is used to express $f(u, v)$. However, EBR falls short in capturing the intricate dynamics of user-item preferences [100; 16; 85]. Consequently, researchers are actively developing techniques to address large-scale retrieval challenges using more sophisticated models, such as Model-Based Retrieval and Generative Retrieval.

Model-Based Retrieval. Advanced retrieval models [100; 99; 102; 16; 9] use multiple MLP layers to model $f(u, v)$. To avoid the substantial computational cost associated with scoring the entire candidate set, model-based retrieval uses techniques such as hierarchical tree structures [100; 99; 102] or learnable paths [16] to build indices that model user interests from coarse to fine detail. Model-based retrieval demonstrates strong fitting ability to a given objective, but its scalability is constrained in practice. For different objectives f , such as maximizing the probability of user interests (active behaviors like clicks, purchases, etc.) or maximizing exposure probability across different scenarios,

it is often necessary to design multi-task learning architectures specifically [31; 54; 68] or even maintain separate retrieval models for each objective (multi-channel retrieval) [27].

Generative Retrieval. The success of generative AI [60; 61; 6; 1] has led to an increasing amount of work focusing on using generative approaches for retrieval [62; 67; 15]. The goal of generative retrieval is to identify the subset that maximizes conditional probability $\text{argTop}_{v \in \mathcal{C}} P(v|u) = \text{softmax}(W^T F(u))|_v$, where $F(u) \in R^D$ is a feature generator for user u and $W \in R^{D \times |\mathcal{C}|}$ is a linear matrix that decodes hidden features into the candidate item space. The benefit of generative retrieval is its better compatibility with Transformer architectures as well as LLMs, whose scalability has been thoroughly validated.

Generative retrieval also has its own issues, such as *model expressiveness*. In model-based retrieval, the relationship between items and users is modeled using multi-layer MLPs, which can theoretically express arbitrary functions. In contrast, the generative retrieval method models the user-item relationship only through inner products, which significantly limits the model’s capacity [100; 85; 27]. Generative retrieval also has issues in *learnability* and *efficiency*. Unlike NLP, where the candidate is typically in the hundreds of thousands, industrial recommender systems deal with millions to billions of candidates. Previous research has identified significant challenges in these high-dimensional spaces, pointing out that learning the W matrix is particularly challenging and that operations like inner product and softmax incur substantial inference costs [62; 66; 90; 89]. Therefore, they introduce semantic IDs, which implicitly break down the candidate set \mathcal{C} . Specifically, T semantic IDs $\{\bar{v}_t\}_{t=1}^T$ are generated in an autoregressive manner, $\bar{v}_t = \arg \max \bar{W}^T F(u, \bar{v}_1, \dots, \bar{v}_{t-1})$, where \bar{W} is low-dimensional matrices, thus significantly reducing the optimization difficulty. Then, the T semantic IDs collectively determine the final item ID. However, some research found that semantic IDs often struggle with fine-grained similarity modeling (discriminability) and cold-start problems (transferability) [79; 78; 76; 101; 39].

In this work, we offer a new implementation of generative retrieval that enhances model expressiveness, learnability, and efficiency, and upgrade it to universal retrieval that surpasses the performance of model-based retrieval while offering superior task scalability.

3 Universal Retrieval Model

Formally, the goal of universal retrieval model is to use a single model to identify the subset of the candidate set \mathcal{C} that maximizes conditional probability for user u under any given objective $o \in \mathcal{O}$, i.e., $\text{argTop}_{v \in \mathcal{C}} P(v|u, o)$, where o might be encountered during training (multi-task learning), or may never appear during training (zero-shot learning). Universal retrieval demands higher model capacity, making generative retrieval more suitable due to its scalability and compatibility with LLMs. Specifically, $P(v|u, o) = \text{softmax}(W^T F(u, o))|_v$. Section 3.1 will detail obtaining the universal representation $F(u, o)$ for a user u and any objective o . Section 3.2 will tackle the learnability challenges of W with tens of millions of candidates. Section 3.3 will focus on reducing the inference cost of $\text{softmax}(W^T F(u, o))$ using sampling techniques.

3.1 Representations for Users & Any Objective

Following the standard practice of Generative Pre-trained Transformers (GPTs) [61], we define different retrieval tasks in natural language and represent users and objectives in sequential form.

User Description. To improve efficiency when handling long user behavior sequences, URM treats items as a kind of special token following [93; 42; 44; 94; 67]. Typical inputs are sentences composed of common text tokens and item IDs (such as [7502]) as follows:

The user attributes are as follows: age {AGE}, gender {GENDER}, located in {PROVINCE}, {CITY}.
 The user has favorited [7502]..., purchased [8274]..., clicked [8380]....

We have aggressively serialized all user features, including static attribute features, statistical features, and various behavioral sequence features. In model-based retrieval methods, when a feature is missing, it is replaced with zeros, and sequence features often require truncation or padding to fit a fixed length. In contrast, our serialized approach omits the text description (e.g., *user has purchased*) for any empty feature, and only truncates sequence features without padding, which reduces computational costs and enhances the model’s robustness against varied inputs.

Retrieval Objectives. We define objectives using different text descriptions, such as: (1) Multi-scenarios: Please retrieve items for scenario A/B/C. (2) Multi-behaviors: Please retrieve items that the user will click on / purchase / favorite. (3) Long-tail item: Please retrieve long-tail items. (4) Serendipity: Please retrieve items from new categories. (5) Long-term interest: Please find items that match user’s long-term interests. (6) Search: Please retrieve items that match the given query. We construct the training set with specific positive samples under these constraints to ensure that the model generates appropriate results for each objective.

Representations for Users & Objectives. In universal retrieval, the model must adapt its output to various user descriptions u and retrieval objectives o , demanding high model capacity. Thus, we use pre-trained LLMs as the feature generator F due to their general capabilities across tasks [1]. Text tokens are mapped to embeddings via the LLM’s vocabulary embedding table. With industry item IDs reaching billions, we employ a separate Distributed HashTable [40] to convert item IDs into unique embeddings, and then use a one-layer MLP to map these embeddings to the same dimension as the token embedding. As shown in Figure 1, the embedding at each position in the sequence is then the sum of the position embedding and either the token or item embedding. These embeddings are then processed through the LLMs to generate the features $F(u, o)$.

Multi-Query Representation. Generative retrieval faces challenges in expressing user-item relationships effectively. Function approximation theory suggests that a linear combination of appropriately chosen basis functions or feature representations can approximate certain classes of complex functions [20]. Inspired by this conclusion, we propose adding multiple special query tokens to the end of the input sequence, allowing the LLMs to generate M user representations $\mathbf{F}(u, o) \in R^{D \times M}$ at these positions during a single LLM forward process. These representations are then used to compute inner products with $W \in R^{D \times |C|}$, and their max value is used as the final score, i.e. $P(v|u, o) = \text{softmax}(\max(W^T F(u, o), \text{axis}=1))|_v$ (in practice, we found that the max function works better than linear combinations). This approach fully preserves the model’s ability to capture complex interactions between users and items, thus enhancing the model’s potential performance. Note that while the query tokens are fixed, their embeddings are learnable. Besides, the forward attention inherent in LLMs allows the hidden features of different query tokens to access the features

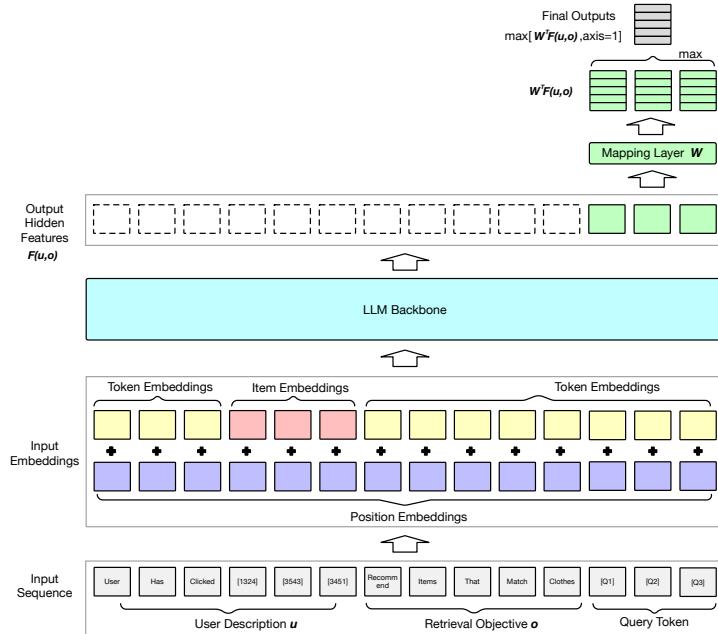


Figure 1: URM architecture. The input sequence consists of user description u , retrieval objective o , and several fixed query tokens. Item IDs in the user description are mapped to item embeddings by a distributed hashtable, and other tokens are mapped to token embeddings. The item embeddings or token embeddings are summed up with position embeddings, and fed into the LLM backbone (For LLMs using RoPE [64], position embedding is not explicitly added). The outputs corresponding to the query tokens are then mapped to the item candidate space through W . To optimize for retrieval objectives in recommender systems, the parameters of LLM backbone are fully fine-tuned.

from preceding query tokens, which enables representations of different query tokens to capture different user interests (see Appendix C.5 for visualizations).

3.2 Mapping to Large-scale Candidate Sets

Using LLMs as feature generators offers strong capability but also increases computational costs. For example, with a sequence length of 1024, a single forward pass of a 1.5B model can achieve a latency of 50 ms, which is the limit for our online recommendation retrieval systems. Generative retrieval methods using semantic IDs require multiple calls to the feature generator, making the computational cost prohibitive when using LLMs as feature generators. This necessitates exploring a new implementation for generative retrieval, one that only requires a single LLM forward pass.

Recall that the primary aim of introducing semantic IDs is to address the issue that matrix W is too large and difficult to learn. We propose an alternative solution to improve its learnability using matrix decomposition $W = UV^T$, where the shapes of U and V are $D \times H$ and $|\mathcal{C}| \times H$ respectively, and H is a lower rank dimension. This approach reduces the parameter size, thereby facilitating easier learning and significantly decreasing computational demands.

Further, we can improve the mapping matrix W to incorporate both discriminability and transferability. In retrieval problems, discriminability refers to the model’s ability to accurately rank two relatively similar items under a given objective. When there is sufficient training data for certain items, assigning distinct parameters for these items is the most effective way to increase discriminability. We denote these distinct parameters as V_{dis} , which correspond to the fully learnable item-side representations during training. Transferability refers to the retrieval model’s ability to identify items that have not been seen in the training set based on their generalized features. In industrial recommender systems, online item candidate sets are constantly changing every minute, making transferability crucial for the retrieval model. To improve transferability, we first serialize the generalized features of each item, including static attributes such as title and category, as well as statistical features like sales and click-through rates, into text. Examples are as follows:

The item title is {TITLE}. The category is {CATEGORY}. The price is {PRICE}. The shop name is {SHOP}. Over the past 7 days, its sales volume is {SALE} and click-through rate is {CTR}.

This serialized text is then fed into a General Text Embedding [92; 43] LLM to obtain a fixed high-dimensional LLM representation. Then, a learnable linear layer is applied to produce a lower-dimensional matrix V_{trans} . Then final mapping matrix is $W = U(V_{\text{dis}} + V_{\text{trans}})^T$, where the total learnable parameters are $(|\mathcal{C}| + 2D) \times H$. For items that never appear in the training set, the mapping matrix $W = UV_{\text{trans}}^T$ allows for the retrieval of cold-start items through transferability.

3.3 Approximation of Large Matrix Multiplication

The learnability issue of W can be solved by matrix decomposition, yet the computation costs remain high due to the high dimension of $|\mathcal{C}|$. During training, we can approximate the negative log-likelihood using Noise Contrastive Estimation (NCE) Loss [19], which avoids large matrix operations and significantly accelerates the training process,

$$\min \mathcal{L}_{\text{NCE}}(u, o) = - \sum_{v \in \mathcal{P}(u, o)} \log \frac{\exp[\max(W_v^T \mathbf{F}(u, o))]}{\sum_{z \in \{v\} \cup \mathcal{N}} \exp[\max(W_z^T \mathbf{F}(u, o))]}, \quad (1)$$

where $\mathcal{P}(u, o)$ denotes the set of positive samples for user u under objective o and \mathcal{N} represents negative examples sampled from the item candidate set according to their occurrence frequency.

During inference, we can also sample a subset S for retrieval,

$$\text{TopK}_{v \in S} P(v|u, o) = \text{softmax}(\max(W_S^T \mathbf{F}(u, o), \text{axis} = 1))|_v \quad (2)$$

where W_S refers to the sub-matrix of W corresponding to the subset S . Although this method can reduce computational cost, many potential items of user interest might never be retrieved. To tackle this issue, we employ probability sampling repeatedly, as detailed in Algorithm 1. Specifically, we construct an ANN index [56] for W that allows each item s to locate its neighbors $\text{NBR}(s)$. Initially, a fixed subset is randomly selected from the candidate set, and item probabilities are computed using the relevant sub-matrix. Then K items are sampled based on their probabilities, forming a new subset influenced by their neighbors. Since probabilities are calculated on a subset shaped by the previous step’s outcomes, this method resembles a 1-gram autoregressive model. The underlying assumption

of Algorithm 1 is that items close in W matrix will also have similar $P(v|u, o)$. Appendix A.3 provides a strict theoretical guarantee and its proof.

Here, we follow the strategy used in NLP by employing probabilistic sampling instead of TopK selection [24; 75]. Theoretically, probabilistic sampling provides a fairer treatment for items ranked at positions K and $K+1$. As the temperature τ approaches 0, the retrieval result of probabilistic sampling will gradually approximate that of TopK selection. Besides, we use the same $\mathbf{F}(u, o)$ in each probability calculation, thus it does not increase the inference cost of LLMs. After optimization, the complexity of matrix computation is $\mathcal{O}(MH(D + TK \times \max(|\text{NBR}(\cdot)|)))$, while using the full W for matrix computation incurs a cost of $\mathcal{O}(MD|\mathcal{C}|)$. In practice, we have $\tau = 0.07$, $M = 128$, $T = 4$, $H = 128$, $D = 4096$, $K = 1000$, $\max(|\text{NBR}(\cdot)|) = 32$, $|\mathcal{C}| \approx 10^7$, thus, the FLOPS is reduced from $5000G$ to $2G$, relatively small compared to LLMs.

4 System Implementation

Figure 2 illustrates the online serving system of URM. Considering the latency and computational cost of LLMs, we develop an asynchronous workflow to meet online requests. URM inference is triggered asynchronously in response to user actions, such as clicks and purchases. Then the engine service reads user features and concatenates them with the retrieval objective texts, and sends a request to the inference service. The inference service reads item embeddings from a distributed hashtable, concatenates them with token embeddings, and then calls vLLM [38] for inference (a single prefill operation). Finally, the sampling technique is used to generate the final retrieval results, which are then stored persistently for use during the online retrieval phase. By aggregating user behavior within a configurable window (we use ten-minute online) to trigger a URM inference, the computational resources can be reduced to those needed for model-based retrieval.

5 Experiments

We validate URM’s effectiveness using 4 public datasets (Section 5.1), an industrial-scale offline dataset (Section 5.2), and online A/B tests (Section 5.5). Ablation studies are presented in Section 5.3, while universal retrieval performance is assessed in Section 5.4. We use Qwen-7B [2] as the feature generator by default, and the results for other LLMs can be found in C.2. The inference efficiency is given in Appendix B.4. Our code is based on LLaMA-Factory [95] and we will release our code related to the public datasets upon acceptance.

5.1 Public Dataset Experiments

Dataset. We first evaluate the performance of URM on four widely-recognized public datasets: Sports & Outdoors, Beauty and Toys & Games [58], and Yelp [81]. We follow the preprocessing methods used in recent works [17; 42; 98] to construct the training and test datasets.

Models. We use multiple retrieval methods as baseline: (1) HGN [53] leverages a hierarchical structure to model user-item interactions. (2) GRU4Rec [23] employs GRUs to model sequential user behavior. (3) Caser [69] leverages CNNs to capture sequential patterns in user behavior. (4) BERT4Rec [65] adapts the BERT architecture for sequential recommendation. (5) FDSA [91] applies a self-attention module to model the relationships between features. (6) SASRec [32] employs self-attention mechanisms to model user behavior sequences. (7) S3-Rec [98] enhances sequential recommendation by incorporating self-supervised learning. (8) E4SRec [42] integrates item IDs within LLMs [70]. (9) P5 [18] employs the T5 [63] model and takes recommendation tasks as purely natural language tasks. (10) TIGER [62] introduces semantic IDs into generative retrieval. (11) IDGenRec [67] utilizes LLMs to create semantically rich and unique textual IDs for items. (12) COBRA [79] incorporates a cascaded sparse-dense representation framework to integrate sparse semantic IDs with dense vectors. The user behavior sequence length is limited to 100.

Algorithm 1: Probabilistic Sampling.

```

 $N(0) = \text{Subset}(\mathcal{C})$ 
 $\hat{\mathbf{F}}(u, o) = U^T \mathbf{F}(u, o)$ 
 $\text{for } t \leftarrow 1 \text{ to } T \text{ do}$ 
     $P(v|u, o) =$ 
         $\text{softmax}[\max(V_{N(t-1)} \hat{\mathbf{F}}(u, o), \text{axis} = 1)/\tau]|_v$ 
         $S(t) \sim \text{Sample}_{v \in N(t-1)}^K P(v|u, o)$ 
         $N(t) = \cup_{s \in S(t)} (\text{NBR}(s) \cup s)$ 
 $\text{end}$ 
 $\text{return } S(T)$ 

```

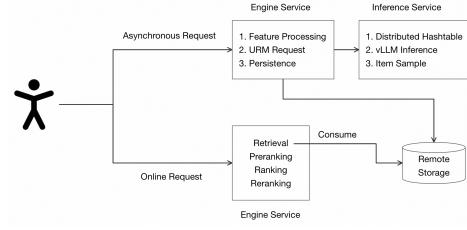


Figure 2: Online Serving System.

Table 1: Performance on 4 public datasets (Only results for $K = 5$ are shown here, while those for $K = 10$ are available in Appendix C.1). RI: Relative Improvement.

Methods	Sports		Beauty		Toys		Yelp	
	HR@5	NDCG@5	HR@5	NDCG@5	HR@5	NDCG@5	HR@5	NDCG@5
HGN	0.0189	0.0120	0.0325	0.0206	0.0321	0.0221	0.0186	0.0115
GRU4Rec	0.0129	0.0086	0.0164	0.0099	0.0097	0.0059	0.0152	0.0099
Caser	0.0116	0.0072	0.0205	0.0131	0.0166	0.0107	0.0151	0.0096
BERT4Rec	0.0115	0.0075	0.0203	0.0124	0.0116	0.0071	0.0051	0.0033
FDSA	0.0182	0.0122	0.0267	0.0163	0.0228	0.0140	0.0158	0.0098
SASRec	0.0233	0.0154	0.0387	0.0249	0.0445	0.0236	0.0162	0.0100
S3-Rec	0.0251	0.0161	0.0387	0.0244	0.0443	0.0294	0.0201	0.0123
E4SRec	0.0281	0.0196	0.0525	0.0360	0.0566	0.0405	0.0266	0.0189
P5	0.0387	0.0312	0.0508	0.0379	0.0648	0.0567	<u>0.0574</u>	<u>0.0403</u>
TIGER	0.0264	0.0181	0.0454	0.0321	0.0521	0.0371	-	-
IDGenRec	0.0429	<u>0.0326</u>	<u>0.0618</u>	<u>0.0486</u>	<u>0.0655</u>	0.0481	0.0468	0.0368
COBRA	0.0305	0.0215	0.0537	0.0395	0.0619	0.0462	-	-
URM	0.0733	0.0488	0.0929	0.0671	0.0888	0.0619	0.0724	0.0476
RI	+70.9%	+49.7%	+50.3%	+38.1%	+35.6%	+9.2%	+26.1%	+18.1%

Table 2: Performance on the industrial dataset (metric: R@1000).

Model	Learning Method	CPR	RSA	RSB	RSC	SR	LR	LIR	PPR	RQ	Avg
Two-tower Model	STL	0.129	0.271	0.166	0.129	0.069	0.066	0.117	0.146	0.355	0.161
	MTL	0.120	0.205	0.166	0.135	0.064	0.115	0.103	0.173	0.257	0.149
Transformer-based Model	STL	0.198	0.409	0.293	0.208	<u>0.104</u>	0.115	0.213	0.143	0.593	0.253
	MTL	0.192	0.390	0.319	0.221	<u>0.076</u>	0.218	0.207	0.401	0.744	0.308
Attention-DNN	STL	0.253	<u>0.477</u>	0.338	0.260	0.106	0.213	<u>0.251</u>	0.353	0.651	0.323
	MTL	0.238	<u>0.456</u>	0.375	<u>0.277</u>	0.062	<u>0.336</u>	0.265	0.478	0.671	0.351
	MTL-SharedBottom	0.243	0.442	0.376	<u>0.270</u>	0.072	0.337	0.224	<u>0.505</u>	0.745	0.357
	MTL-MMoE	0.233	0.439	0.375	0.257	0.070	0.325	0.218	0.491	0.736	0.349
	MTL-PLE	<u>0.256</u>	0.451	<u>0.397</u>	0.274	0.062	0.327	0.224	0.512	<u>0.761</u>	<u>0.363</u>
URM	MTL	0.263	0.530	0.439	0.362	0.093	0.285	0.240	0.581	0.835	0.403

The performance is evaluated using Hit Rate (HR@ K) and Normalized Discounted Cumulative Gain (NDCG@ K), computed at different ranking positions. As presented in Table 1, URM outperforms the strongest baseline by an average of **46%** and **29%** in terms of HR@5 and NDCG@5, respectively.

5.2 Industrial-scale Experiments

Dataset. Next, we verify the effectiveness of URM in an industrial-scale dataset, which is obtained from real traffic logs of the online system. The typical objectives include Click Prediction Retrieval (**CPR**), Retrieval for Scene A (**RSA**), Scene B (**RSB**), Scene C (**RSC**), Serendipity Retrieval (**SR**), Long-term Retrieval (**LR**), Long-tail Item Retrieval (**LIR**), Purchase Prediction Retrieval (**PPR**), and Retrieval with Query (**RQ**). This dataset contains hundreds of millions of samples and more than one billion distinct items in user behavior sequences. The candidate set contains tens of millions of items. A more detailed definition of the dataset is in Appendix B.1. We use samples from day 1 to day \mathcal{T} for training and the samples of day $\mathcal{T} + 1$ for testing. To improve training efficiency, we aggregate each user’s daily behaviors as the target set, which makes these retrieval tasks more challenging. All methods are compared in this setting to facilitate a more effective result.

Models. For baseline, we use the most commonly used methods in practice. (1) Two-tower Model, which uses two multi-layer MLPs to convert user-side and item-side features into embeddings, and uses the inner product to obtain the final score [26]. (2) Transformer-based Model, which uses a multi-layer transformer to convert user behavior sequence features into user embeddings [86; 46]. (3) Attention-DNN, which calculates cross-attention between the user behavior sequence and the target, and then uses multiple layers to calculate the final score. For each method, we implement a single-task learning version (STL) with only one objective and a multi-task learning version (MTL) with multiple objectives. For the Attention-DNN model, which has the best performance among them, we further provide the Shared Bottom version [55], the MMoE version [54], and the PLE version [68]. More details can be found in Appendix B.2. We tuned the hyperparameters of each method, including the embedding size and number of layers, using the validation set. The user behavior sequence length is 300, and the total length of text tokens, query tokens, and item IDs in URM is truncated to 1024.

We use recall to evaluate the effectiveness of our proposed method. Denote the output item set as \mathcal{P} and the ground truth as \mathcal{G} , then recall $R@K = |\mathcal{P} \cap \mathcal{G}|/|\mathcal{G}|$, where $|\mathcal{P}| = K$. We show results in Table 2. Due to the complex task relationships, it is difficult for a single approach to perform

well on all objectives, which leads to multi-task training being sometimes effective and sometimes causing seesaw phenomenon. In contrast, our proposed URM achieves the best performance in 6 out of 9 objectives and achieves a relative improvement of **11.0%** in average across all objectives. Besides, for RQ (Retrieval with Query), there is a specifically designed three-tower model, which adds an independent tower for queries and reaches 0.822 on R@1000. In contrast, our model achieves comparable results without any specific design. More results can be found in Appendix C.3.

5.3 Ablation Study

Multi-Query Representation. As shown in Figure 3, as the number of query tokens increases, URM’s ability to represent the target item set becomes stronger, resulting in better performance.

Matrix Decomposition. As demonstrated in Table 3, the discriminability matrix V_{dis} significantly enhances the model’s retrieval capabilities across all items, while the transferability matrix V_{trans} boosts the model’s ability to retrieve items unseen during training. This underscores the effectiveness of matrix decomposition and highlights the importance of different item representations. Appendix C.6 further provides a visualization comparison of different item representations.

Figure 3: The effect of query token number M .

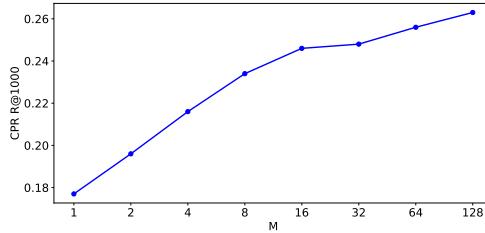


Table 3: The effect of item representation V in matrix decomposition on CPR R@1000.

V	All	Unseen
V_{dis}	0.256	0.116
V_{trans}	0.152	0.101
$V_{\text{dis}} + V_{\text{trans}}$	0.263	0.130

Table 4: The effect of sampling steps T .

T	Recall Precision
1	0.2%
2	2.1%
3	41.7%
4	91.0%
5	91.1%

Probabilistic Sampling. We use probability sampling to approximate the probability calculation for the entire candidate set. As the number of sample steps T increases, the retrieval precision ($\frac{\text{R@1000 w/ sampling}}{\text{R@1000 w/o sampling}}$) continually increases and tends to converge when $T = 4$.

5.4 Universal Retrieval Performance

The performance of universal retrieval can be assessed from two perspectives: firstly, its effectiveness in handling objectives encountered during training (in the multi-task learning setting), and secondly, its transferability to objectives not seen during training (in the zero-shot learning setting). Only several examples are provided here, and more examples can be found in Appendix C.3 and C.4.

Multi-Task Learning. Although the candidates and generation are both different from pretrained LLMs, we observe that URM remains sensitive to the text input. As shown in Table 5(a), with objectives for specific scenarios, URM aligns more closely with the data distribution and achieves higher performance, with a relative improvement of over 20%. As illustrated in Table 5(b), after using the SR (Serendipity Retrieval) objective, the percent of the new category increased from 18.8% to 46.2%, leading to a relative increase of 82.3% in SR R@1000.

Table 5: Universal retrieval performance.

(a) multi-scenario retrieval			
Objective	RSA R@1000	RSB R@1000	RSC R@1000
CPR	0.440	0.335	0.278
RSA	0.530	0.409	0.240
RSB	0.522	0.439	0.257
RSC	0.444	0.327	0.362
RI	+20.5%	+31.0%	+30.2%

(b) serendipity retrieval			
Objective	CPR R@1000	SR R@1000	Percent of New Category
CPR	0.263	0.051	18.8%
SR	0.213	0.093	46.2%
RI	-	+82.3%	+145.7%

(c) hybrid objectives		
Objective	RQ R@1000	Percent of Long-tail Items
RQ	0.835	79.6%
LIR	0.630	81.6%
RQ × LIR	0.836	82.4%

Zero-shot Learning. We find that URM effectively adapts to hybrid objectives, such as long-tail item retrieval with a specific query. As illustrated in Table 5(c), combining RQ and LIR objectives not only improves relevance with the query but also increases the proportion of long-tail items. We also examine URM’s performance on seen and unseen queries, noting that unseen queries

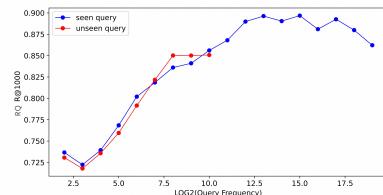


Figure 4: Performance on unseen queries.

are often long-tail and have distinct distributions. To enable fair comparison, we plot performance across varying query frequencies. Figure 4 illustrates URM’s strong robustness to query input.

5.5 Online Results

We evaluate URM on our advertising platform using real traffic. A key challenge in retrieval is the discrepancy between offline and online metrics, where offline improvements often don’t lead to online gains. Traditional retrieval models require multiple steps to adjust online retrieval objectives and corresponding results. As a result, optimization for online performance heavily relies on human priors. In contrast, URM can dynamically adjust retrieval outputs based on input instruction, which means that we can conduct online A/B tests with various retrieval objectives and user feature combinations to optimize the online performance. 4 online metrics are utilized to assess performance: advertising revenue (the core metric for our advertising platform), CTR, CVR, and proportion of long-tail items.

In Table 6, we observe a 3.01% growth in advertising revenue. Additionally, the CTR and CVR metric sees a rise of 0.78% and 1.24% respectively, suggesting that URM delivers more precise results to users. Notably, the growth for long-tail items surpasses the overall increase, highlighting that URM is more fair and friendly to long-tail items.

6 Related Work

Multi-Task Learning For Recommendation. Multi-task learning methods can be categorized into different parameter sharing methods [59; 35; 49; 57; 22] and optimization strategies [34; 10; 82], and have been widely applied to recommender systems [54; 55; 68]. However, in practice, it is difficult to ensure that the performance of each task improves after multi-task learning, especially when the number of tasks continues to increase [84; 30]. Thus, it’s often necessary to carefully design the model structure based on the data proportions and the task relationships. This has also led to the isolated states among different recommendation problems, such as multi-scenario modeling [31], multi-objective modeling [55; 96], long-tail recommendation [50], etc. In addition, search tasks can be viewed as a specialized type of recommendation with explicit query constraints. Due to significant distribution discrepancies and limited model capacity, traditional recommendation models cannot simultaneously handle scenarios both including and excluding explicit inputs [47].

LLMs for Recommendation. Inspired by LLMs [60; 61; 6; 1], recent approaches treat recommendation tasks as natural language tasks, generating text results directly through prompting and in-context learning [17; 11; 48; 88; 36; 5; 4]. However, in real systems, users typically have hundreds or even millions of behaviors, leading to at least tens of thousands of text tokens, which increases inference costs and decreases LLM performance. Thus, some methods introduce a hierarchical structure that encodes each item’s text or image information into item representations and feeds them into LLMs to generate a high-level user representation [8; 29; 80; 25; 83; 73]. However, text-based recommendations lack discrimination because many items have similar images, titles, and categories, yet differ in recognition and exposure. Another approach employs traditional ID embeddings to represent items [93; 42; 44; 94; 67]. However, these methods do not fully exploit the advantages of LLMs in recommender systems when dealing with numerous objectives and even unseen objectives, nor do they address the generation problem within a candidate set of tens of millions.

7 Conclusion

In this paper, we answer the questions of why and how LLMs can be used for recommender systems, and introduce Universal Retrieval Model (URM). The value of LLMs in recommender systems lies in their ability to unify various retrieval objectives into a cohesive input-output framework, enabling them to address multiple retrieval objectives simultaneously and prompt tuning the online performance directly. To facilitate the implementation of LLMs in real-world recommender systems, we propose multi-query representation to enable complex user-item relationships modeling, introduce matrix decomposition to improve model learnability, discriminability, and transferability, and incorporate probabilistic sampling to reduce computation costs. Ultimately, URM can adaptively generate a retrieval set from tens of millions of candidates based on any given objective while maintaining latency within tens of milliseconds, achieving significant gains in both offline and online metrics.

Table 6: Online results from April 28, 2025 - May 14, 2025.

Metric	RI
Revenue	+3.01%
CTR	+0.78%
CVR	+1.24%
#Long-tail Items	+2.23%

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. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.
- [3] Yong Bai, Rui Xiang, Kaiyuan Li, Yongxiang Tang, Yanhua Cheng, Xialong Liu, Peng Jiang, and Kun Gai. Chime: A compressive framework for holistic interest modeling. *CoRR*, abs/2504.06780, 2025.
- [4] Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yancheng Luo, Chong Chen, Fuli Feng, and Qi Tian. A bi-step grounding paradigm for large language models in recommendation systems. *arXiv preprint arXiv:2308.08434*, 2023.
- [5] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *RecSys*, 2023.
- [6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.
- [7] Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.
- [8] Junyi Chen, Lu Chi, Bingyue Peng, and Zehuan Yuan. Hllm: Enhancing sequential recommendations via hierarchical large language models for item and user modeling. *arXiv preprint arXiv:2409.12740*, 2024.
- [9] Rihan Chen, Bin Liu, Han Zhu, Yaoxuan Wang, Qi Li, Buting Ma, Qingbo Hua, Jun Jiang, Yunlong Xu, Hongbo Deng, et al. Approximate nearest neighbor search under neural similarity metric for large-scale recommendation. In *CIKM*, 2022.
- [10] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, 2018.
- [11] Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, ZhongXiang Sun, Xiao Zhang, and Jun Xu. Uncovering chatgpt’s capabilities in recommender systems. *Proceedings of the 17th ACM Conference on Recommender Systems*, 2023.
- [12] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.
- [13] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [14] DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024.

- [15] Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *CoRR*, abs/2502.18965, 2025.
- [16] Weihao Gao, Xiangjun Fan, Chong Wang, Jiankai Sun, Kai Jia, Wenzhi Xiao, Ruofan Ding, Xingyan Bin, Hui Yang, and Xiaobing Liu. Deep retrieval: Learning a retrievable structure for large-scale recommendations, 2021.
- [17] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In *RecSys*, 2022.
- [18] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the Sixteenth ACM Conference on Recommender Systems*, 2022.
- [19] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.
- [20] D.C. HANDSCOMB, D.F. MAYER, and M.J.D. POWELL. Chapter 7 - the general theory of linear approximation. In D.C. HANDSCOMB, editor, *Methods of Numerical Approximation*, pages 61–71. Pergamon, 1966.
- [21] Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.
- [22] Falk Heuer, Sven Mantowsky, Saqib Bukhari, and Georg Schneider. Multitask-centernet (mcn): Efficient and diverse multitask learning using an anchor free approach. In *ICCV*, 2021.
- [23] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.
- [24] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *ICLR*, 2020.
- [25] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. Towards universal sequence representation learning for recommender systems. In *KDD*, 2022.
- [26] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. Embedding-based retrieval in facebook search. In *KDD*. ACM, 2020.
- [27] Junjie Huang, Jizheng Chen, Jianghao Lin, Jiarui Qin, Ziming Feng, Weinan Zhang, and Yong Yu. A comprehensive survey on retrieval methods in recommender systems, 2024.
- [28] Junjie Huang, Jiarui Qin, Jianghao Lin, Ziming Feng, Weinan Zhang, and Yong Yu. Unleashing the potential of multi-channel fusion in retrieval for personalized recommendations. In *WWW*, 2025.
- [29] Jian Jia, Yipei Wang, Yan Li, Honggang Chen, Xuehan Bai, Zhaocheng Liu, Jian Liang, Quan Chen, Han Li, Peng Jiang, and Kun Gai. Knowledge adaptation from large language model to recommendation for practical industrial application. *ArXiv*, abs/2405.03988, 2024.
- [30] Junguang Jiang, Baixu Chen, Junwei Pan, Ximei Wang, Liu Dapeng, Jie Jiang, and Mingsheng Long. Forkmerge: Mitigating negative transfer in auxiliary-task learning. In *NeurIPS*, 2023.
- [31] Yuchen Jiang, Qi Li, Han Zhu, Jinbei Yu, Jin Li, Ziru Xu, Huihui Dong, and Bo Zheng. Adaptive domain interest network for multi-domain recommendation. In *CIKM*, 2022.
- [32] Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *ICDM*, 2018.

- [33] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- [34] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.
- [35] Iasonas Kokkinos. Ubertnet: Training a ‘universal’ convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, 2017.
- [36] Xiaoyu Kong, Jiancan Wu, An Zhang, Leheng Sheng, Hui Lin, Xiang Wang, and Xiangnan He. Customizing language models with instance-wise lora for sequential recommendation. In *NeurIPS*, 2024.
- [37] Denis Kotkov, Alan Medlar, and Dorota Glowacka. Rethinking serendipity in recommender systems. In *SIGIR*, 2023.
- [38] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.
- [39] Kaiyuan Li, Rui Xiang, Yong Bai, Yongxiang Tang, Yanhua Cheng, Xialong Liu, Peng Jiang, and Kun Gai. Bbqrec: Behavior-bind quantization for multi-modal sequential recommendation. *CoRR*, abs/2504.06636, 2025.
- [40] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI’14, page 583–598, 2014.
- [41] Ruyu Li, Wenhao Deng, Yu Cheng, Zheng Yuan, Jiaqi Zhang, and Fajie Yuan. Exploring the upper limits of text-based collaborative filtering using large language models: Discoveries and insights. *CoRR*, abs/2305.11700, 2023.
- [42] Xinhang Li, Chong Chen, Xiangyu Zhao, Yong Zhang, and Chunxiao Xing. E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation. *CoRR*, abs/2312.02443, 2023.
- [43] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.
- [44] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, and Xiang Wang. Llara: Aligning large language models with sequential recommenders. *CoRR*, abs/2312.02445, 2023.
- [45] Guanyu Lin, Zhigang Hua, Tao Feng, Shuang Yang, Bo Long, and Jiaxuan You. Unified semantic and ID representation learning for deep recommenders. *CoRR*, abs/2502.16474, 2025.
- [46] Chi Liu, Jiangxia Cao, Rui Huang, Kai Zheng, Qiang Luo, Kun Gai, and Guorui Zhou. Kuaiformer: Transformer-based retrieval at kuaishou, 2024.
- [47] Jinhan Liu, Qiyu Chen, Junjie Xu, Junjie Li, Baoli Li, and Sulong Xu. A unified search and recommendation framework based on multi-scenario learning for ranking in e-commerce. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2890–2894, 2024.
- [48] Junling Liu, Chao Liu, Renjie Lv, Kang Zhou, and Yan Zhang. Is chatgpt a good recommender? A preliminary study. *CoRR*, abs/2304.10149, 2023.
- [49] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *CVPR*, 2019.

- [50] Siyi Liu and Yujia Zheng. Long-tail session-based recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems*, 2020.
- [51] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015.
- [52] Xinch Chen Luo, Jiangxia Cao, Tianyu Sun, Jinkai Yu, Rui Huang, Wei Yuan, Hezheng Lin, Yichen Zheng, Shiyao Wang, Qigen Hu, Changqing Qiu, Jiaqi Zhang, Xu Zhang, Zhiheng Yan, Jingming Zhang, Simin Zhang, Mingxing Wen, Zhaojie Liu, Kun Gai, and Guorui Zhou. QARM: quantitative alignment multi-modal recommendation at kuaishou. *CoRR*, abs/2411.11739, 2024.
- [53] Chen Ma, Peng Kang, and Xue Liu. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, 2019.
- [54] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *SIGKDD*, 2018.
- [55] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *SIGIR*, 2018.
- [56] Yu Malkov and Dmitry Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.
- [57] Kevins-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In *CVPR*, 2019.
- [58] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, 2015.
- [59] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016.
- [60] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *Technical report, OpenAI*, 2018.
- [61] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *Technical report, OpenAI*, 2019.
- [62] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Keshavan, Trung Vu, Lukasz Heidt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. Recommender systems with generative retrieval. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.
- [63] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczeczla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. In *ICLR*, 2022.
- [64] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023.
- [65] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, 2019.

- [66] Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. Learning to tokenize for generative retrieval. In *NeurIPS*, 2023.
- [67] Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. Idgenre: Llm-recsys alignment with textual id learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2024.
- [68] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *RecSys*, 2020.
- [69] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, 2018.
- [70] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.
- [71] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mi-haylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poultion, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023.
- [72] Hastagiri P. Vanchinathan, Isidor Nikolic, Fabio De Bona, and Andreas Krause. Explore-exploit in top-n recommender systems via gaussian processes. In *Proceedings of the 8th ACM Conference on Recommender Systems*, 2014.
- [73] Jie Wang, Fajie Yuan, Mingyue Cheng, Joemon M. Jose, Chenyun Yu, Beibei Kong, Xiangnan He, Zhijin Wang, Bo Hu, and Zang Li. Transrec: Learning transferable recommendation from mixture-of-modality feedback. In *APWeb-WAIM*, 2024.
- [74] Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Learnable item tokenization for generative recommendation. In *CIKM*, pages 2400–2409, 2024.
- [75] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023.
- [76] Ye Wang, Jiahao Xun, Minjie Hong, Jieming Zhu, Tao Jin, Wang Lin, Haoyuan Li, Linjun Li, Yan Xia, Zhou Zhao, and Zhenhua Dong. EAGER: two-stream generative recommender with behavior-semantic collaboration. In *SIGKDD*, pages 3245–3254, 2024.
- [77] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.
- [78] Liu Yang, Fabian Paischer, Kaveh Hassani, Jiacheng Li, Shuai Shao, Zhang Gabriel Li, Yun He, Xue Feng, Nima Noorshams, Sem Park, Bo Long, Robert D Nowak, Xiaoli Gao, and Hamid Eghbalzadeh. Unifying generative and dense retrieval for sequential recommendation, 2024.

- [79] Yuhao Yang, Zhi Ji, Zhaopeng Li, Yi Li, Zhonglin Mo, Yue Ding, Kai Chen, Zijian Zhang, Jie Li, Shuanglong Li, and Lin Liu. Sparse meets dense: Unified generative recommendations with cascaded sparse-dense representations. *arXiv preprint arXiv:2503.02453*, 2025.
- [80] Yuyang Ye, Zhi Zheng, Yishan Shen, Tianshu Wang, Hengruo Zhang, Peijun Zhu, Runlong Yu, Kai Zhang, and Hui Xiong. Harnessing multimodal large language models for multimodal sequential recommendation. *arXiv preprint arXiv:2408.09698*, 2024.
- [81] Yelp. Yelp open dataset. <https://business.yelp.com/data/resources/open-dataset/>, 2025. Accessed: 2025-04-01.
- [82] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *NeurIPS*, 2020.
- [83] Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. Where to go next for recommender systems? ID- vs. modality-based recommender models revisited. In *SIGIR*, 2023.
- [84] Amir Roshan Zamir, Alexander Sax, William B. Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018.
- [85] Jiaqi Zhai, Zhaojie Gong, Yueming Wang, Xiao Sun, Zheng Yan, Fu Li, and Xing Liu. Revisiting neural retrieval on accelerators. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.
- [86] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, Yinghai Lu, and Yu Shi. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. In *ICML*, 2024.
- [87] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*, 2019.
- [88] Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. Recommendation as instruction following: A large language model empowered recommendation approach. *CoRR*, abs/2305.07001, 2023.
- [89] Kangning Zhang, Jiarui Jin, Yingjie Qin, Ruilong Su, Jianghao Lin, Yong Yu, and Weinan Zhang. Learning id-free item representation with token crossing for multimodal recommendation. *CoRR*, abs/2410.19276, 2024.
- [90] Taolin Zhang, Junwei Pan, Jinpeng Wang, Yaohua Zha, Tao Dai, Bin Chen, Ruisheng Luo, Xiaoxiang Deng, Yuan Wang, Ming Yue, Jie Jiang, and Shu-Tao Xia. Towards scalable semantic representation for recommendation. *CoRR*, abs/2410.09560, 2024.
- [91] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. Feature-level deeper self-attention network for sequential recommendation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019. ijcai.org*, 2019.
- [92] Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, et al. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. *arXiv preprint arXiv:2407.19669*, 2024.
- [93] Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. Collm: Integrating collaborative embeddings into large language models for recommendation. *CoRR*, abs/2310.19488, 2023.
- [94] Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2024.

- [95] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyuan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, 2024.
- [96] Yong Zheng and David (Xuejun) Wang. A survey of recommender systems with multi-objective optimization. *Neurocomputing*, 2022.
- [97] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1059–1068, 2018.
- [98] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1893–1902, 2020.
- [99] Han Zhu, Daqing Chang, Ziru Xu, Pengye Zhang, Xiang Li, Jie He, Han Li, Jian Xu, and Kun Gai. Joint optimization of tree-based index and deep model for recommender systems. *Advances in Neural Information Processing Systems*, 32, 2019.
- [100] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. Learning tree-based deep model for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1079–1088, 2018.
- [101] Jieming Zhu, Mengqun Jin, Qijiong Liu, Zexuan Qiu, Zhenhua Dong, and Xiu Li. Cost: Contrastive quantization based semantic tokenization for generative recommendation. In *RecSys*, pages 969–974, 2024.
- [102] Jingwei Zhuo, Ziru Xu, Wei Dai, Han Zhu, Han Li, Jian Xu, and Kun Gai. Learning optimal tree models under beam search. In *International Conference on Machine Learning*, pages 11650–11659. PMLR, 2020.

A Appendix

A.1 Limitations

Computational Cost. In our efforts to deploy LLMs within industrial recommender systems, we’ve incorporated strategies such as multi-query representation and probabilistic sampling techniques. Despite these advancements, it is important to acknowledge that LLMs inherently increase the costs associated with training and inference compared to traditional recommendation models. Consequently, in our online implementation, we ensure that the daily training data is processed within a 24-hour timeframe by aggregating user actions on an hourly basis and subsequently sampling only 5% of this aggregated data for supervised fine-tuning (SFT). During the inference phase, we apply a 10-minute window for asynchronous processing. These measures reflect necessary trade-offs between effectiveness and efficiency. While the computational expense is a known limitation of LLMs, our online experiments affirm their substantial value in enhancing recommender systems.

Task Versatility. We discover that while URM can adapt to certain new objectives, these are often closely linked to the original training objectives. When faced with entirely new objectives, it’s challenging for URM to adjust its output based on textual input. The solution to this problem is to increase the amount of training data, enabling URM to learn more mappings from retrieval objectives to results. For example, by incorporating search data, we aim to utilize user behavior data associated with diverse queries to enhance URM’s ability to generalize to new prompts.

A.2 Broader Impacts

URM integrates the vast world knowledge encapsulated in LLMs to offer personalized retrieval to users from a human-centered, interpretable perspective. It is also beneficial in promoting long-tail items, as evidenced by our online experimental results, which demonstrate that our approach is more accommodating to these long-tail items. Consequently, this contributes to more equitable market competition and may help mitigate monopolistic dominance.

A.3 Theoretical Guarantee for the Probability Sampling Algorithm

A potential assumption of Algorithm 1 is that items close in the W matrix, specifically those that become neighbors, will receive similar scores from URM. A theoretical guarantee is as follows:

Theorem A.1. *Assuming that the representations of two items v_1 and v_2 are very similar, i.e., $\|W_{v_1} - W_{v_2}\| \leq \epsilon$. Meanwhile, we apply bound constraints to the representation $F(u, o)$,*

$$\bar{F}(u, o) = \frac{F(u, o)}{\max(\|F(u, o)\|/B, 1)} \quad (3)$$

where scaling adjustments will only be applied to $F(u, o)$ if its norm exceeds B , then we have

$$\|\max(W_{v_1}^T \bar{F}(u, o)) - \max(W_{v_2}^T \bar{F}(u, o))\| \leq \epsilon B \quad (4)$$

Proof. First, we provide the proof under $M = 1$.

$$\begin{aligned} & \|\max(W_{v_1}^T \bar{F}(u, o)) - \max(W_{v_2}^T \bar{F}(u, o))\| \\ &= \|W_{v_1}^T \bar{F}(u, o) - W_{v_2}^T \bar{F}(u, o)\| \\ &\leq \|W_{v_1} - W_{v_2}\| \cdot \|\bar{F}(u, o)\| \\ &\leq \epsilon B \end{aligned}$$

This means that as long as the norm of $\bar{F}(u, o)$ remain within a constant range, the scores between neighboring items will be similar. Then we extend the conclusion to the case when $M > 1$.

Without loss of generality, let $\max(W_{v_1}^T \bar{F}(u, o)) = W_{v_1}^T \bar{F}_i(u, o)$ and $\max(W_{v_2}^T \bar{F}(u, o)) = W_{v_2}^T \bar{F}_j(u, o)$, where $1 \leq i, j \leq M$. By applying bound constraints to the representation, we have $\|\bar{F}_i(u, o)\| \leq B$, $\|\bar{F}_j(u, o)\| \leq B$.

When $i = j$, we have

$$\|\max(W_{v_1}^T \bar{\mathbf{F}}(u, o)) - \max(W_{v_2}^T \bar{\mathbf{F}}(u, o))\| \leq \|W_{v_1} - W_{v_2}\| \cdot \|\bar{F}_i(u, o)\| \leq \epsilon B \quad (5)$$

When $i \neq j$, we have

$$\max(W_{v_1}^T \bar{\mathbf{F}}(u, o)) - \max(W_{v_2}^T \bar{\mathbf{F}}(u, o)) = W_{v_1}^T \bar{F}_i(u, o) - W_{v_2}^T \bar{F}_j(u, o) \quad (6)$$

By the properties of the max function, we have $W_{v_1}^T \bar{F}_i(u, o) \geq W_{v_2}^T \bar{F}_j(u, o)$ and $W_{v_2}^T \bar{F}_j(u, o) \geq W_{v_1}^T \bar{F}_i(u, o)$, thus we have

$$W_{v_1}^T \bar{F}_j(u, o) - W_{v_2}^T \bar{F}_j(u, o) \leq \max(W_{v_1}^T \bar{\mathbf{F}}(u, o)) - \max(W_{v_2}^T \bar{\mathbf{F}}(u, o)) \leq W_{v_1}^T \bar{F}_i(u, o) - W_{v_2}^T \bar{F}_i(u, o) \quad (7)$$

According to the properties of the norm, we have

$$\begin{aligned} & \|\max(W_{v_1}^T \bar{\mathbf{F}}(u, o)) - \max(W_{v_2}^T \bar{\mathbf{F}}(u, o))\| \\ & \leq \max\{\|W_{v_1}^T \bar{F}_j(u, o) - W_{v_2}^T \bar{F}_j(u, o)\|, \|W_{v_1}^T \bar{F}_i(u, o) - W_{v_2}^T \bar{F}_i(u, o)\|\} \\ & \leq \max\{\|W_{v_1} - W_{v_2}\| \|\bar{F}_j(u, o)\|, \|W_{v_1} - W_{v_2}\| \|\bar{F}_i(u, o)\|\} \\ & \leq \max\{\epsilon B, \epsilon B\} \\ & = \epsilon B \end{aligned}$$

□

Here, B acts as a control hyperparameter for the model's complexity. A larger B leads to larger model complexity, but also larger inconsistency in $W^T \bar{\mathbf{F}}(u, o)$ between neighbors, thus reducing retrieval precision. In embedding-based retrieval, user representations are often normalized [26], which is somewhat similar to the situation where $B = 1$, resulting in very low model expressiveness but very high retrieval precision, suitable for various ANN retrieval methods. In URM, we have $B = 100$, and experiments (Table 4) have shown that retrieval precision still remains high under this setting after multiple samplings.

B Implementation Details

B.1 Dataset Details

Our industrial dataset contains hundreds of millions of samples and more than one billion distinct items in user behavior sequences. The candidate set contains tens of millions of items.² We use samples from day 1 to day T for training and the samples of day $T + 1$ for testing. We construct datasets for typical objectives, listed as follows:

- **Click Prediction Retrieval (CPR):** Positive samples are click behaviors.
- **Retrieval for Scene A (RSA), Scene B (RSB), Scene C (RSC):** Positive samples are the items exposed to users in 3 scenarios with distribution shift.
- **Serendipity Retrieval (SR):** Retrieve new items for users to mitigate the information cocoon effect. Positive samples are real click behaviors, whose categories have not appeared in user behaviors.
- **Long-term Retrieval (LR):** Retrieve items based on long-term behaviors of users. Positive samples are the users’ real click behaviors, whose categories only appear in the long-term user behavior sequence.
- **Long-tail Item Retrieval (LIR):** Retrieve long-tail items to mitigate the bias of the recommender system. Positive samples are the items that users have clicked, and the popularity of which is below a certain threshold.
- **Purchase Prediction Retrieval (PPR):** Positive samples are the items that the users have purchased.
- **Retrieval with Query (RQ):** Given a query, retrieve the most relevant items that the user is most likely to be interested in. Positive samples are the user’s click behaviors under a certain query.

B.2 Model Implementation Details

Two-tower Model. Both the user-side features and the item-side features are flattened to 2-dimensional, and then fed into 3-layer MLP networks with layers [256, 128, 64] to obtain the final 64-dimensional user representations and item representations. The user representations will not be normalized, while the item representations will be normalized.

Transformer-based Model. After adding the position embedding, the user behavior sequence embedding, task embedding, and query embedding are concatenated and fed into a 5-layer Transformer network, where each Transformer layer has 4 heads and the dimension of the FFN layer is 4 times larger than the input dimension. The embedding at the last position is taken as the user embedding. The item side follows the same as the two-tower model: features are flattened and then processed through a 3-layer MLP network with dimensions [256, 128, 64] to obtain the final 64-dimensional embedding, which is then normalized.

Attention-DNN. The user side includes multiple aggregated features and 3 sequence features of different lengths. The sequence features are used to compute cross-attention with the item features [97]. The item side also includes multiple features. After flattening all features, they are collectively fed into a fully connected network with layers [256, 128, 64, 1] to obtain the final score. Due to the increase in computation cost for each item, we build an HNSW index [56] on the item representations for faster inference following [100; 9].

Attention-DNN + Shared Bottom. Retain the attention module and all features in Attention-DNN, and use different MLPs for modeling each task separately.

Attention-DNN + MMoE. Retain the attention module and all features in Attention-DNN, and introduce 4 MLPs to generate the middle representations. The task embedding is fed into the gate network to obtain combination weights, which are then used to adaptively combine the outputs from the MLP.

Attention-DNN + PLE. Retain the attention module and all features in Attention-DNN and adopt 4 MLPs for shared module and task-specific module separately to generate the middle representations

²Due to data security concerns, specific numbers cannot be open.

with the dim of 64. These middle representations of the 4 shared modules and each task-specific module are then merged using a gate network and fed into each task’s 2-layer MLPs to obtain the final score.

URM. The item IDs are first mapped to item embeddings using a distributed hashtable, and then an MLP is used to further map the embeddings into a 4096-dimensional tensor. Simultaneously, the other tokens are also transformed into this dimension by the token embedding table. As depicted in Figure 1, once merged with positional embeddings, the input embeddings are fed into the LLM backbone. The hidden state from the final transformer layer will subsequently pass through 2 MLP layers (U and V) to yield the final item probability. Between MLP layers, we add RMSNorm [87] to enhance the model expressiveness.

B.3 Training Details

For URM, we use AdamW with an initial learning rate of $2e^{-5}$, weight decay of 0.05, batch size of 8192, and cosine learning rate decay. Each batch consists of 10,000 negative examples, which are sampled from the item candidates according to the $3/4$ power of their occurrence frequency. The sampled training dataset is trained for a single epoch. To preserve the pre-trained knowledge in LLM, we follow transfer learning [51] and set the learning rate of the pre-trained layers to be $1/10$ of the learning rate of the other layers. Using Qwen-7B as the LLM backbone on 64 NVIDIA H20 GPUs, it takes 25 hours to process 50 million training examples. Due to the large volume of data in recommender systems and the high training cost of LLMs, we perform a 5% sampling of the data when training URM. Meanwhile, we initialize V_{dis} using the ID Embedding from the two-tower model.

B.4 Inference Efficiency of URM

In the inference stage, our model architecture differs from conventional LLMs in two aspects: (1) item IDs in the input sequence, and (2) probabilistic sampling in the vast item space. Specifically, for input item IDs, we utilize distributed hashtable lookup technology to reduce the embedding retrieval latency to within a few milliseconds. For the item sample module, we propose Algorithm 1, enabling efficient sampling within a ten-million-item database in under 10 milliseconds. As a result, the negative impact of these two additional modules on performance is relatively minor, resulting in our model being nearly equivalent to an LLM inference with an output token length of 1. This enables seamless utilization of existing inference frameworks (e.g., vLLM [38]) and acceleration techniques (including batching, key-value caching, FlashAttention [13; 12], etc.).

The 99th percentile latency (p99) for various model sizes at different input lengths on NVIDIA H20 GPUs is presented in Table 7. The results demonstrate that by reducing the model size, efficiency can be further enhanced, which in turn allows for the processing of longer input lengths of tokens. In contrast, generative retrieval methods based on semantic IDs, which require auto-regressively generating multiple tokens, result in online latencies on the order of T .

Table 7: P99 Latency (ms) for various model sizes at different input lengths.

Model Size	Input Length	256	1024	4096
		7.7	18	65
0.5B		18	50	197
1.5B		34	99	404
3B		63	233	947
7B				

C More Experimental Results

C.1 More Experiment Results on Public Dataset

As demonstrated in Table 8, URM exhibits superior performance over the most robust baseline, achieving average relative improvements of **49%** in HR@10 and **37%** in NDCG@10 on 4 public datasets, respectively.

Table 8: Performance on 4 public datasets($K = 10$).

Methods	Sports		Beauty		Toys		Yelp	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
HGN	0.0313	0.0159	0.0512	0.0266	0.0497	0.0277	0.0326	0.0159
GRU4Rec	0.0204	0.0110	0.0283	0.0137	0.0176	0.0084	0.0263	0.0134
Caser	0.0194	0.0097	0.0347	0.0176	0.0270	0.0141	0.0253	0.0129
BERT4Rec	0.0191	0.0099	0.0347	0.0170	0.0203	0.0099	0.0090	0.0045
FDSA	0.0288	0.0156	0.0407	0.0208	0.0381	0.0189	0.0276	0.0136
SASRec	0.0350	0.0192	0.0605	0.0318	0.0698	0.0318	0.0274	0.0136
S3-Rec	0.0385	0.0204	0.0647	0.0327	0.0700	0.0376	0.0341	0.0168
E4SRec	0.0410	0.0237	0.0758	0.0435	0.0798	0.0479	0.0418	0.0238
P5	0.0460	0.0336	0.0664	0.0429	0.0709	0.0587	0.0703	0.0445
TIGER	0.0400	0.0225	0.0648	0.0384	0.0712	0.0432	-	-
IDGenRec	0.0574	0.0372	0.0814	0.0541	0.0870	0.0551	0.0578	0.0404
COBRA	0.0434	0.0257	0.0725	0.0456	0.0781	0.0515	-	-
URM	0.1049	0.0590	0.1225	0.0766	0.1221	0.0726	0.0866	0.0558
RI	+82.8%	+58.6%	+50.5%	+41.6%	+40.3%	+23.7%	+23.2%	+25.4%

C.2 Experiments with More LLM Backbones

Table 9: Comparison between different LLM backbones on public dataset.

Methods	Beauty			
	HR@5	NDCG@5	HR@10	NDCG@10
E4SRec	0.0525	0.0360	0.0758	0.0435
URM(Qwen-7B)	0.0929	0.0671	0.1225	0.0766
URM(LLaMA2-13B)	0.0905	0.0607	0.1339	0.0747

Public Dataset. We conduct evaluations using the LLaMA2-13B model, as utilized in E4SRec, on the Amazon Beauty dataset as an example to isolate the effects of the backbone and provide a more aligned comparison. The results are presented in Table 9, demonstrating that performance improvements are primarily attributed to our framework rather than deriving from a more advanced backbone.

Industrial Dataset. We conduct experiments on Qwen-1.8B, LLaMA2-13B [71], and DeepSeek-V2-Lite(16B) [14] on the industrial-scale dataset. As shown in Table 10, alternative backbone models achieve performance comparable to Qwen-7B. This indicates that the performance improvement is not tied to any specific backbone; rather, it benefits from the versatility and general applicability of our URM framework design.

Table 10: Comparison between different LLM backbones on industrial dataset (metric: R@1000).

Methods	CPR	RSA	RSB	RSC	SR	LR	LIR	PPR	RQ	AVG
Qwen-7B	0.263	0.530	0.439	0.362	0.093	0.285	0.240	0.581	0.835	0.403
Qwen-1.8B	0.255	0.529	0.434	0.351	0.081	0.243	0.230	0.572	0.835	0.392
LLaMA2-13B	0.263	0.527	0.434	0.37	0.107	0.276	0.228	0.582	0.845	0.404
DeepSeek-V2-Lite(16B)	0.258	0.514	0.413	0.357	0.102	0.264	0.214	0.569	0.851	0.394

C.3 More Experiments on Multi-Task Learning

STL vs. MTL on URM. Table 11 gives the performance degradation of single-task learning compared to multi-task learning. Compared to other methods, URM performs better in multi-task settings, indicating that URM is less prone to seesaw phenomenon. Further, Figure 5 compares the performance of single-task and multi-task learning with different amounts of labeled data. The performance of URM on the single CPR task can also continuously approach that of multi-task learning as the amount of task-specific data increases. However, URM with multi-task learning can converge faster to better performance using much less task-specific data, which has significant value for recommendation scenarios where a large amount of task-specific data is not available.

Table 11: Performance on the industrial dataset (metric: R@1000).

Model	Learning Method	CPR	RSA	RSB	RSC	SR	LR	LIR	PPR	RQ	Avg
URM	STL	0.137	0.226	0.219	0.089	0.042	0.089	0.107	0.427	0.668	0.223
URM	MTL	0.263	0.530	0.439	0.362	0.093	0.285	0.240	0.581	0.835	0.403

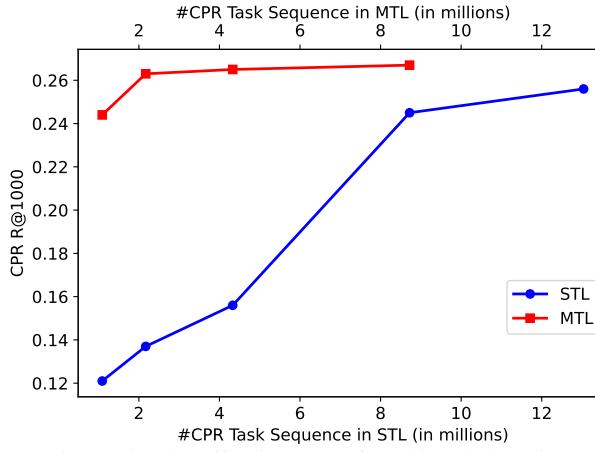


Figure 5: The effectiveness of multi-task learning.

Long-tail Item Retrieval. As shown in Table 12, after using the LIR objective, the proportion of long-tail items in the output item set increases by 49.5% relatively.

Table 12: Performance on Long-tail Item Retrieval.

Objective	CPR R@1000	LIR R@1000	Percent of Long-tail Items
CPR	0.263	0.240	54.6%
LIR	0.202	0.240	81.6%
RI	-	+0%	+49.5%

Long-term Retrieval. As shown in Table 13, employing the LR objective results in a relative increase of 96.6% in the proportion of items within the output set that align with the user’s long-term interests and a relative increase of 36.4% in LR R@1000.

Table 13: Performance on Long-term Retrieval.

Objective	CPR R@1000	LR R@1000	Percent of Long-term Interest
CPR	0.263	0.209	32.6%
LR	0.149	0.285	64.1%
RI	-	+36.4%	+96.6%

Category Diversity. In some of the training data, we inject the number of categories in the target set into the objective as shown below.

Inputs: Please retrieve the top $\{K\}$ categories that users are most likely to be interested in.

Then, during the test stage, we observe that by adjusting the size of K in the objective, we could modify the category diversity in the output item set as shown in Table 14.

Table 14: Performance on category diversity.

K	Category Recall@1000	#Category
4	0.767	74.7
8	0.772	80.8
16	0.775	94.1
32	0.777	104.2
64	0.778	113.7
128	0.780	117.1
RI	+1.7%	+56.8%

C.4 More Experiments on Zero-shot Learning

Hybrid Objectives: Serendipity & Purchase Prediction Retrieval. As shown in Table 15, when the SR objective and PPR objective are combined, the set produced by the URM aligns better with the purchase objective, thereby improving PPR R@1000 compared to using the SR objective. At the same time, the percentage of new categories in the output set also increases compared to that of the PPR objective. This makes the output item set close to the distribution of items that the user has not clicked on before, but is highly likely to purchase next.

Table 15: The result of hybridizing the SR and PPR objectives.

Objective	PPR R@1000	Percent of New Category
PPR	0.581	25.6%
SR	0.411	46.2%
PPR × SR	0.510	45.1%

Hybrid Objectives: Long-tail Item Retrieval for Scene A. As shown in Table 16, when the RSA objective and LIR objective are combined, the output set is closer to the distribution of scenario A, thereby improving RSA R@1000 compared to using the LIR objective. Additionally, there is an increase in the proportion of long-tail items within the output set compared to the RSA objective alone. Consequently, the output item set becomes more representative of the long-tail items that are most likely to occur in scenario A.

Table 16: The result of hybridizing the RSA and LIR objectives.

Objective	RSA R@1000	Percent of Long-tail Items
RSA	0.530	72.6%
LIR	0.410	81.6%
RSA × LIR	0.483	81.5%

New Objective. Further, we validate the zero-shot task transfer performance from the CPR task to the RQ task. As shown in Figure 6, as the number of training samples increases, URM trained only on the CPR task can still learn the mapping between the query and target sets and reach 0.698 on RQ R@1000! The reason is that our input sequence includes both text and item ID tokens, which allows the URM to align items with the semantic space, even when no query-related tasks exist in the training samples. Of course, the performance of zero-shot task transfer still has a significant gap with supervised training on the corresponding task (MTL vs STL on CPR). However, this fully demonstrates that URM inherits the powerful task transfer capabilities of LLM.

New Context. By modifying the query in the RQ objective to a specific context, we can model the changes in user behavior over periods, such as during seasonal changes, festival celebrations, or

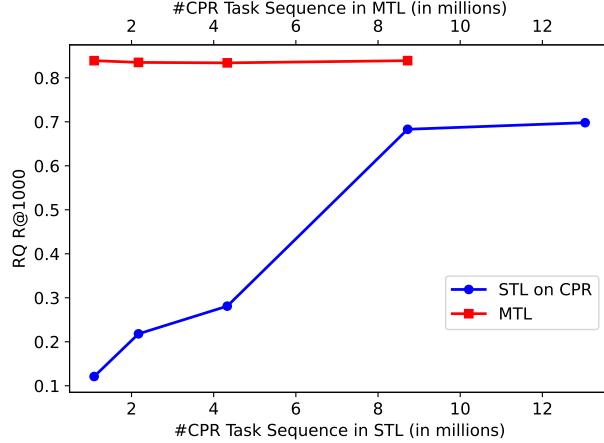


Figure 6: Zero-shot task transfer performance from the CPR task to the RQ task.

shopping events. Figure 7 gives two examples. The fundamental reason why URM can achieve this is that the training data for the RQ (Retrieval with Query) has established a comprehensive mapping between user behaviors, text constraints, and the target item set. This has two potential advantages.

1. By changing the query to a certain context, we can inject external world knowledge into URM, allowing the results to get ready for potential upcoming events.
2. This also supports the combination of URM and Chain-of-Thought (CoT) technologies [77]. Specifically, the LLM generates intermediate results in the form of text through reasoning, and then injects this text as context into URM, thereby producing the final set.

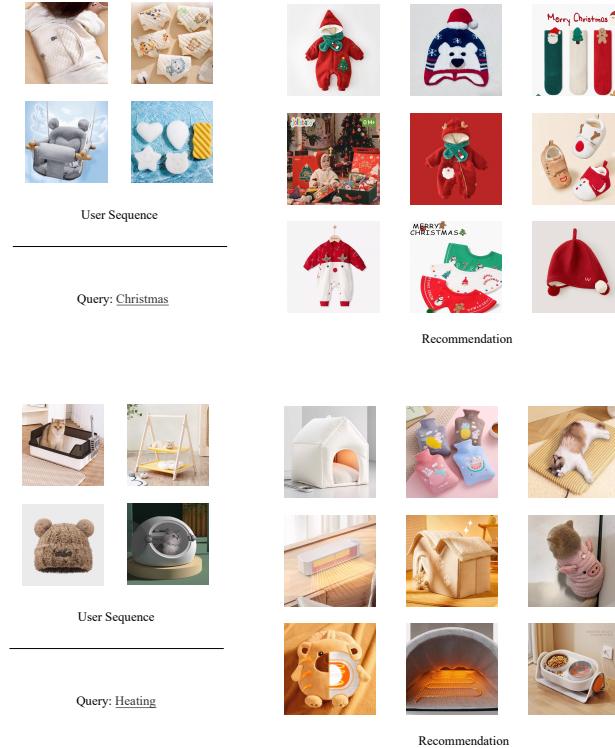


Figure 7: Zero-shot task transfer to a new context. Christmas is a festive celebration, and heating is a common need in winter. When these two contexts are injected as queries into the objective, the model can retrieve items based on the user’s historical behavior and the given context simultaneously.

C.5 More Experiments on Multi-Query Representation

Visualization of User Representations from Different Query Tokens. To further validate the role of query tokens, we categorize the output 1000 items from URM into the user representations that have the largest inner product with it. The items under 3 user representations are shown in Figure 8. It can be observed that different low-dimensional user representations have captured different user interests, which enhances the representation ability of URM for the target item set.

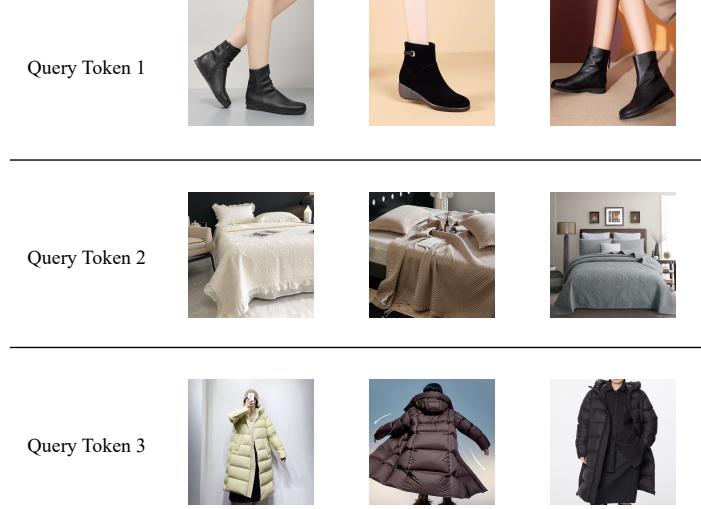


Figure 8: Visualization of user representations from different query tokens. Each token captures distinct facets of user interest, enabling LLM to jointly express different aspects of the target item set.

Activation of Query Tokens on Different Tasks. Figure 9 shows the activation proportion for 8 different query tokens in the CPR and RQ tasks. A token is activated when the dot product of the user representation corresponding to that query token and the target item representation is used for the final output. We find that different query tokens exhibit variation across different tasks.

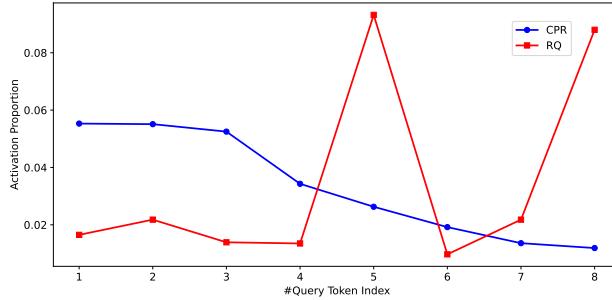


Figure 9: Activation proportion of different query tokens in CPR and RQ tasks.

The Necessity of Multi-Query Representation Format. URM employs multiple query tokens to generate several 128-dimensional output user representations, which are subsequently computed alongside the 128-dimensional item representation. To explore the effect of the multi-query representation format, we conduct experiments using higher embedding dimensions and a single token, with the results presented in Table 17. On one hand, when a single token is used to derive a 4096-dimensional user representation, which is then split into 32 separate 128-dimensional representations, the CPR recall drops significantly from 0.248 to 0.163. This highlights the advantage of generating multiple user representations with different query tokens, even when the formal dimension remains constant. On the other hand, increasing the item representation dimension from 128 to 4096 (resulting

in logits computed in a higher-dimensional space) does not enhance performance compared to the logits derived from multiple tokens, yielding a mere CPR recall of 0.203. These findings confirm that the multi-query representation design doesn't solely benefit from its higher dimensionality; rather, it demonstrates superior effectiveness in capturing the diverse interests of users within this specific format.

Table 17: The effect of multiple query tokens (metric: R@1000).

Methods	CPR	RQ
32 Tokens with 128-d Embeddings	0.248	0.835
1 Token with 4096-d User Representation	0.163	0.774
1 Token with 4096-d Item Representation	0.203	0.781

C.6 Visualization Comparison of Different Item Representation

To visualize different item representations, we use the nearest neighbor retrieval approach to find similar items to a given item under a specific representation. As shown in Figure 10,11,12, V_{trans} mainly captures similarity relationships, whereas V_{dis} are more centered on co-occurrence relationships. The fusion representations fall somewhere in between these two.

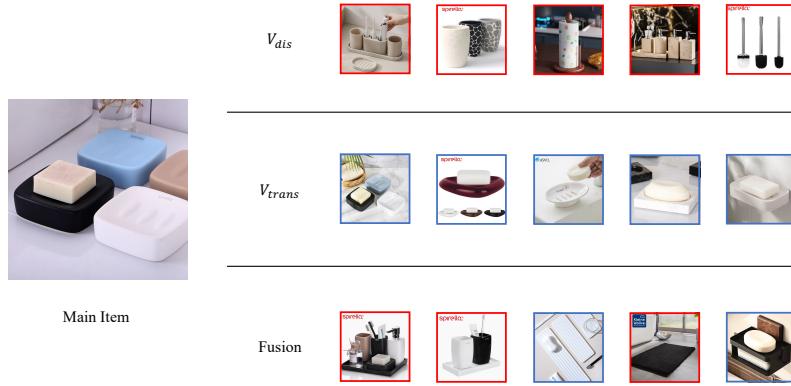


Figure 10: Visualization of different item representations (case 1).

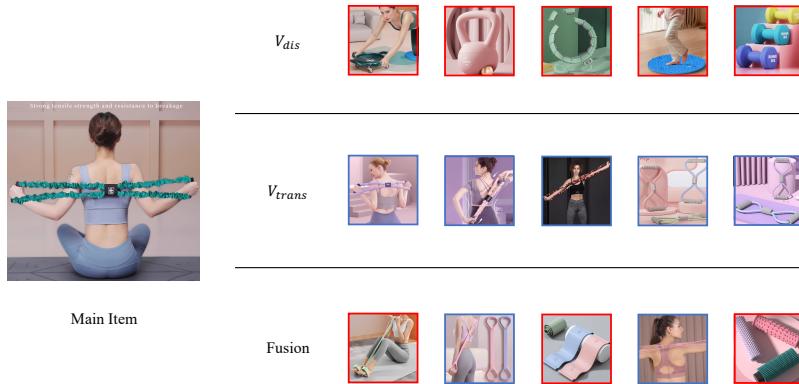


Figure 11: Visualization of different item representations (case 2).

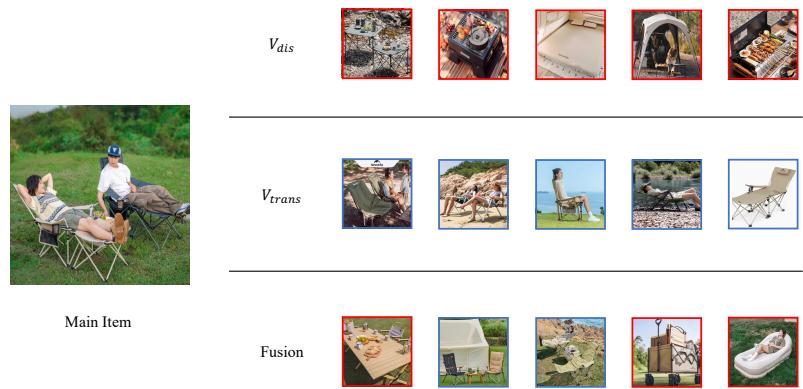


Figure 12: Visualization of different item representations (case 3).

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: The abstract and introduction clearly state our contributions – introducing the Universal Retrieval Model (URM), enhancing model expressiveness, learnability, and efficiency of generative retrieval, and achieving improvements in both offline and online metrics.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Due to space constraints, we have placed the limitations section in the Appendix A.1.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide detailed proof in Appendix A.3 for the assumption in Algorithm 1.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: In Algorithm 1, we provide a pseudo-code for our probabilistic sampling methods. In Section 5.1,5.2, Appendix B.1,B.2,B.3, we provide all the details to reproduce our experiment.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Upon acceptance, we will release the code related to the public datasets. Due to data security concerns, our code for the industrial online and offline experiments cannot be made public.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The experimental setting is presented Section 5.1 and 5.2. The full details can be found in Appendix B.2 and B.3.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Given the high computational cost, we did not include detailed statistical significance measures for our experiments. Nonetheless, our method consistently outperforms baseline methods across multiple tasks using both public and industrial datasets, showing marked improvements in online performance. This serves as compelling evidence of our approach’s effectiveness.

Guidelines:

- The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The computer resources for our experiment are given in Appendix B.3 and B.4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: Our manuscript adheres to the NeurIPS Code of Ethics by ensuring equitable treatment, safeguarding data privacy, and so on.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss potential positive societal impacts in Appendix A.2, and we foresee no negative societal impacts resulting from our research. There may be concerns regarding the use of sensitive user or item information; however, we address this by omitting such details and utilizing non-sensitive attributes in practice.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have cited all the original papers that produced the code package or dataset.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: Our paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [\[Yes\]](#)

Justification: Section 3.1 provides the inputs and outputs of the LLMs and a detailed explanation of the modifications we made to the existing LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.