MixLoRA-DSI: Dynamically Expandable Mixture-of-LoRA Experts for Rehearsal-Free Generative Retrieval over Dynamic Corpora

Tuan-Luc Huynh[1] **Thuy-Trang Vu**[1] **Weiqing Wang**[1]
Trung Le[1] **Dragan Gašević**[2] **Yuan-Fang Li**[1] **Thanh-Toan Do**[1]
[1] Department of Data Science & AI, Monash University, Australia
[2] Department of Human Centred Computing, Monash University, Australia
{tuan.huynh1,trang.vu1,teresa.wang,
trunglm,dragan.gasevic,yuanfang.li,toan.do}@monash.edu

Abstract

Continually updating model-based indexes in generative retrieval with new documents remains challenging, as full retraining is computationally expensive and impractical under resource constraints. We propose MixLoRA-DSI, a novel framework that combines an *expandable* mixture of Low-Rank Adaptation experts with a layer-wise out-of-distribution (OOD)-driven expansion strategy. Instead of allocating new experts for each new corpus, our proposed expansion strategy enables sublinear parameter growth by selectively introducing new experts only when significant number of OOD documents are detected. Experiments on NQ320k and MS MARCO Passage demonstrate that MixLoRA-DSI outperforms full-model update baselines, with minimal parameter overhead and substantially lower training costs.[1]

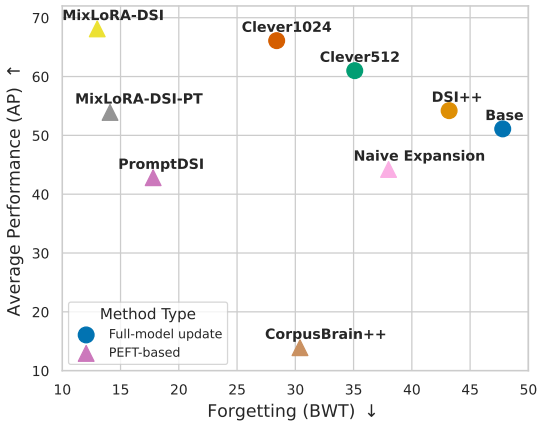

Figure 1: Continual learning performance of different Generative Retrieval methods on NQ320k (Recall@10). PEFT-based MixLoRA-DSI variants offer the best trade-off between average performance and forgetting, while being significantly more parameter-efficient than full-model updates. See Table 1 for more detailed results.

1 Introduction

Generative retrieval (GR) leverages pretrained Transformer models as model-based information retrieval (IR) index, also known as differential search index (DSI), to encode corpus information directly within the model parameters (Tay et al., 2022). However, GR methods primarily assume a static corpus, where the document set remains unchanged. This overlooks the real-world challenge of IR systems to handle dynamic corpora, where they need to continually integrate new documents.

Since GR methods perform indexing via model training, continually updating index may require expensive retraining. Recent works have explored continual learning (CL) approaches to mitigate catastrophic forgetting while efficiently updating the model without retraining from scratch (McCloskey and Cohen, 1989; Mehta et al., 2023), such as rehearsal-based CL (Mehta et al., 2023; Kishore et al., 2023; Chen et al., 2023). However, these methods either adopt atomic document identifiers (docids) (Mehta et al., 2023) or only evaluate on small-scale datasets, limiting their scalability on large-scale retrieval benchmarks (Bajaj et al., 2016). Additionally, rehearsal-based methods require access to previous documents, raising privacy concerns (Shokri and Shmatikov, 2015), making them impractical for real-world applications. Furthermore, dynamic corpora naturally align with task-agnostic (i.e., task-free) CL, characterized by blurry task boundaries (Aljundi et al., 2019), since new documents may share high semantic similarity to previous indexed ones. These challenges collectively indicate the need for a ***task-agnostic***, ***rehearsal-free***, and ***scalable*** solution.

In this work, we propose ***MixLoRA-DSI***, a parameter-efficient fine-tuning (PEFT)-based dynamically expandable framework for rehearsal-free GR over dynamic corpora, which outperforms various GR methods (Figure 1). Our approach builds on mixture-of-experts (MoE) (Shazeer et al., 2017),

[1]Our code is available at https://github.com/LouisDo2108/MixLoRA-DSI.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 381–397
November 4-9, 2025

where each expert is a low-rank adapter (Hu et al., 2022). MixLoRA-DSI replaces the standard top-k linear router with a top-k cosine classifier (Gidaris and Komodakis, 2018) and introduces a novel auxiliary loss. This improved router alleviates the recency bias in MoE routing of the original router, while the novel auxiliary loss encourages accurate token-to-expert alignment and preserves expert diversity. As a result, MixLoRA-DSI is inherently ***task-agnostic*** as the top-k routers dynamically select experts per token without relying on rigid task-specific parameters. The use of LoRA experts ensures efficient indexing while preserving knowledge from previously indexed documents, mitigating catastrophic forgetting and enabling ***rehearsal-free*** CL. We enhance ***scalability*** by opting for residual quantization (RQ)-based docids (Zeng et al., 2024a) and developing ***CL strategies tailored to RQ-based docids*** by leveraging their structural properties.

MixLoRA-DSI views GR over dynamic corpora through the lens of out-of-distribution (OOD) detection. Naively adding new experts for each corpus update leads to linear parameter growth without ensuring efficient capacity utilization. However, in practice, queries often share structural patterns and differ mainly in ***OOD keywords*** associated with newly indexed documents, thus new docids may frequently overlap in latent space. Building on this insight, we propose an ***layer-wise OOD-driven expansion strategy*** that selectively adds new experts when existing ones fail to capture significant novel information from new documents. This decision is governed by the ***layer-wise energy scores*** (Liu et al., 2020) from the routers, enabling sublinear parameter growth while maintaining retrieval effectiveness. Overall, we make the following contributions:

- We propose MixLoRA-DSI, the first OOD-driven dynamic expansion framework for rehearsal-free GR over dynamic corpora, achieving sublinear parameter growth.
- We improve MoE routing by replacing the standard router with a top-k cosine classifier, jointly optimized with a novel auxiliary loss to encourage expert specialization while ensuring balanced token-to-expert assignments.
- We propose continual learning strategies tailored to RQ-based document identifiers.
- Experiments on NQ320k (Kwiatkowski et al., 2019) and MS MARCO Passage dataset demonstrate that MixLoRA-DSI is significantly more parameter-efficient and robust to forgetting than full-model update baselines.

2 Preliminary

2.1 Generative Retrieval (GR)

Differentiable Search Index (DSI). DSI (Tay et al., 2022) is a representative method in generative retrieval (GR); we use DSI and GR interchangeably. Let f_θ denote the T5 (Raffel et al., 2020)-based DSI model. During indexing, DSI is trained to map each document $d_i \in D$ to a unique identifier $id_i \in I_D$. During the retrieval stage, given a set of query-docid pairs $Q_D^{\text{retrieval}} = \{(q_i, id_{q_i})\}$, DSI autoregressively generates a ranked list of top-k docids using constrained beam search, which are then mapped back to documents in D.

Document representations and document identifiers (docids) are two crucial design choices in DSI. At retrieval time, since DSI must process queries, which are typically much shorter and linguistically distinct from the training documents, it results in a distribution mismatch between indexing and retrieval. To address this issue, the current standard practice is to represent documents using pseudo-queries generated by off-the-shelf query generation models such as docT5query (Nogueira et al., 2019). These pseudo-queries are synthetic queries that simulate what users might ask to retrieve the corresponding documents. They serve as replacements for the original document content during indexing, helping to better align the training and inference distributions (Zhuang et al., 2022; Pradeep et al., 2023). We adopt Residual Quantization (RQ)-based docids, encoding documents as sequences of RQ codes (Babenko and Lempitsky, 2014), which is the first to show effectiveness on large-scale IR (Zeng et al., 2024a).

Dynamic corpora in GR. Dynamic corpora (Mehta et al., 2023) assumes there are $T+1$ corpora $\{D_0, \ldots, D_T\}$, each associated with a set of docids $\{I_0, \ldots, I_T\}$. D_0 is a large-scale corpus with annotated query-docid pairs, while $D_{>0} = \{D_1, \ldots, D_T\}$ contains newly added documents without annotations. Let θ_{t-1} denote the GR model parameters after indexing $\{D_0, \ldots, D_{t-1}\}$. At each timestep $t > 0$, the model must update to θ_t to incorporate D_t. Evaluation at timestep t is performed on retrieval queries $\{Q_{D_0}^{\text{retrieval}}, \ldots, Q_{D_t}^{\text{retrieval}}\}$ using θ_t.

2.2 Mixture of LoRA Experts

Mixture of Experts (MoE). An MoE layer (Shazeer et al., 2017) comprises of N feed-forward network (FFN) experts, denoted as $\{E_i\}_{i=1}^{N}$, and a router $R \in \mathbb{R}^{\text{dim}\times N}$. Given token x, the router computes logits $R(x) = R^\top x$. The gate value of the i-th expert is computed as:

$$p_i(x) = \frac{e^{R(x)_i}}{\sum_{j=1}^{N} e^{R(x)_j}},$$

where $p_i(x)$ represents the softmax-normalized probability for expert E_i. The token is routed to a subset of experts corresponding to the top-k gate logits. Let $Topk(\cdot)$ be a function that selects the top-k highest logits while setting the rest as zero. The output of a MoE layer is a weighted sum of the top-k experts, with the weights determined by the gate values:

$$\text{MoE}(x) = \sum_{i=1}^{N} Topk(p_i(x))E_i(x).$$

MoE is often trained with an auxiliary load balancing loss to mitigate the unbalanced load for experts when training (Fedus et al., 2022). Given N experts and a batch B with T tokens, the auxiliary loss $\mathcal{L}_{\text{aux}}$ is computed as following:

$$\begin{aligned} \mathcal{L}_{\text{aux}}^{\text{original}} &= a \cdot N \cdot \sum_{i=1}^{N} \mathcal{F}_i \cdot \mathcal{P}_i, \\ \mathcal{P}_i &= \frac{1}{T} \sum_{x\in B} R(x)_i, \\ \mathcal{F}_i &= \frac{1}{T} \sum_{x\in B} \mathbb{1}\{\arg\max_k R(x)_k = i\}, \end{aligned} \tag{1}$$

where $\mathcal{F}_i$ is the fraction of tokens dispatched to expert i, $\mathcal{P}_i$ is the fraction of the router probability allocated for expert i, and α stands for the weight coefficient of this auxiliary loss.

MixLoRA: Mixture of LoRA Experts. MixLoRA (Yu et al., 2024) employs LoRAs (Hu et al., 2022) as experts for parameter-efficient fine-tuning. Specifically, the pre-trained FFN remain frozen, and we apply a unique set of LoRAs to each FFN layer. For all experiments, we employ MixLoRA with top-2 router in specific T5's decoder blocks, since the decoder is responsible for autoregressive docid generation. Our MixLoRA implementation is detailed in Appendix D.

3 Method

3.1 MixLoRA-DSI

Figure 2 presents the overview of MixLoRA-DSI, a novel dynamic expansion framework for rehearsal-free GR over dynamic corpora, equipped with an OOD-driven dynamic expansion mechanism, IR-specific improved routers, and RQ-based docids CL strategies.

Expanding MixLoRA. MixLoRA can be expanded with new LoRAs while maintaining nearly constant computational cost. Each expansion adds a new LoRA to the existing set and appends a weight vector $w \in \mathbb{R}^{\text{dim}}$ to the router: $R' = \{R; w\}$. Naively adding one expert per corpus to each MixLoRA layer causes linear parameter growth without guaranteed efficiency. To address this, we seek a principled criterion to determine: ***When should we expand a specific MixLoRA layer?***

Energy-based OOD dynamic expansion. We observe that: (1) LoRA experts store knowledge, while the router selects which expert to activate, naturally making it responsible for deciding when expansion is needed. (2) In IR, queries often share structural similarities, with differences primarily in OOD keywords. Despite having unique docids, new documents frequently overlap in latent space. We thus propose a dynamic expansion strategy that detects OOD signals in the latent space relative to previously indexed documents, ensuring expert expansion only when necessary.

Among various OOD detection methods, we adopt the energy score (Liu et al., 2020), because of its two key advantages: (1) The router in each MixLoRA layer functions as a classifier, making the energy score a natural fit. (2) Unlike generative methods, it avoids computationally expensive and often unreliable density estimation (Lee et al., 2020; Ardywibowo et al., 2022; Nalisnick et al., 2019). Given an input token x to the router R of a MixLoRA layer, we treat the router as an N-class classifier. The energy score parameterized by the router R is computed as:

$$\mathbf{E}(x; R) = -T \cdot \log \sum_{i=1}^{N} \exp(\langle R_i, x\rangle / T), \tag{2}$$

where T is the temperature, typically set to 1.0 to ensure the energy score remains parameter-free,

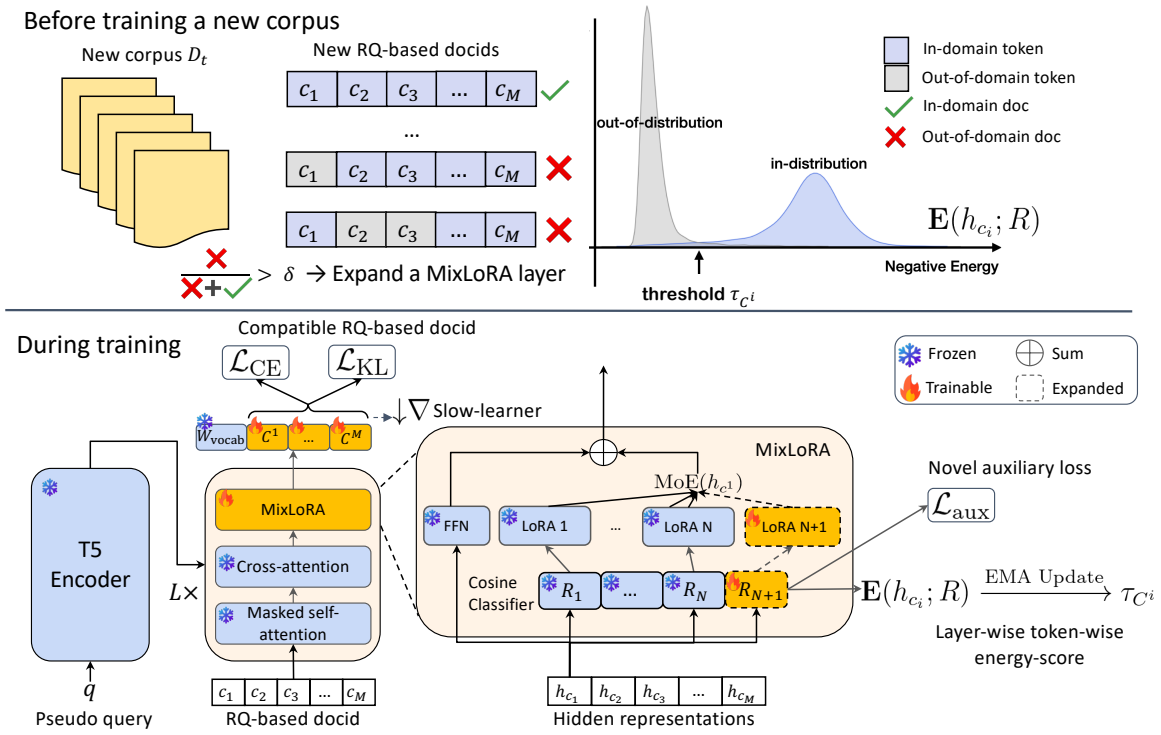

Figure 2: Overview of MixLoRA-DSI. Before training, the model scans the new corpus to to perform energy-based OOD dynamic expansion (Section 3.1). During training, the new router weights are optimized with the novel auxiliary loss $\mathcal{L}_{\text{aux}}$ (Eq. 3), and the energy score thresholds are updated via exponential moving average (EMA). New LoRA experts and RQ-based docid embeddings $\{C^1, \ldots, C^M\}$ are trained with the proposed RQ-based docids CL strategies (Section 3.2). Residual connections and layer normalizations are dropped without loss of generality.

and R_i represents the weight vector of the router corresponding to the i-th expert.

The energy-based OOD dynamic expansion process is illustrated in the top-half of Figure 2.We compute energy scores for each token at every MixLoRA layer. During indexing, we maintain exponential moving averages of in-distribution (ID) energy scores $\{\tau_{C^i}\}_{i=1}^{M}$ from previously indexed corpora from the router (i.e., the cosine classifier in a MixLoRA layer, bottom-half of Figure 2). Before indexing a new corpus, we iterate over its training pseudo-queries (i.e., the document representation) to detect OOD cases. Since energy scores are higher for OOD data, a token c_i is flagged as OOD if its energy score exceeds the corresponding threshold τ_{C^i} (gray c_i in Figure 2). A query is considered OOD if it contains at least one OOD token (Out-of-domain doc, marked with red X marks). If the number of OOD queries for a MixLoRA layer exceeds a predefined threshold δ, we trigger expert expansion for that layer (dashed LoRA and dashed cosine classifier weight in the MixLoRA layer, bottom-half of Figure 2).

Improving the router of MixLoRA. The original MoE design incorporates an auxiliary load balancing loss (Eq. 1) to evenly distribute tokens across experts. However, we observe recency bias in the router, where tokens are predominantly routed to newly added experts while ignoring older ones. This issue is exacerbated by the softmax competition in the router, where the unfrozen logits of new router weights dominate over frozen ones, skewing expert selection. To address this, we propose replacing the original softmax router, which has unnormalized router weights and takes unnormalized inputs, with a cosine classifier (Gidaris and Komodakis, 2018). The cosine classifier router is optimized via cosine embedding loss to better align layer-wise token hidden representations with router weights, i.e., the first term of Eq. (3). Additionally, to prevent router weights from becoming overly similar, we introduce another loss term that encourages new router weights to remain distinct from previous ones, i.e., the second term of Eq. (3).

The auxiliary router loss for a MixLoRA layer with N experts is defined as:

$$\mathcal{L}_{\text{aux}}(h_{id}; R) = \frac{1}{M}\sum_{i=1}^{M}(1-\cos(R_N, h_{c_i})) + \sum_{j=1}^{N-1}\max(0, \cos(R_j, R_N)), \tag{3}$$

where $\cos(\cdot,\cdot)$ is the cosine function, $h_{id} = \{h_{c_1}, \ldots, h_{c_M}\}$ denotes the docids's hidden representations obtained from the previous Transformer decoder block. Each $h_{c_i} \in h_{id}$ and the column vectors of R are unit L_2-norm weight vectors.

3.2 RQ-based docids CL strategies

RQ-based docids. Mehta et al. (2023); Huynh et al. (2024) adopt atomic docids due to their robustness against forgetting compared to semantic docids. However, they scale poorly due to the required large softmax space. In this work, we challenge this observation and adopt scalable RQ-based docids. Following Zeng et al. (2024a), we train the document embeddings with M RQ codebooks $C^m := \{c_k\}_{k=1}^{K} \in \mathbb{R}^{\text{dim}\times K}$, each containing K centroids, to generate an RQ code of length M for each document. The learned RQ centroids are concatenated to the DSI's vocabulary weights W_{vocab} for subsequent optimization: $W_{\text{RQ}} = \{W_{\text{vocab}}; C^1; \ldots; C^M\}$, where $\{\cdot;\cdot\}$ is concatenation. Please refer to Appendix C for more details about RQ-based docids.

RQ-based docids mask. RQ-based docids have a structured design, where each token c_i is predicted from a specific segment of the extended vocabulary W_{RQ}, corresponding to its RQ codebook. To avoid requiring DSI to implicitly learn this structure, we apply a masking strategy that explicitly restricts the softmax computation to the relevant segment of the vocabulary at each decoding step, preventing unnecessary competition between logits. Specifically, for each token c_i at position i in the sequence, we define a binary mask $\mathbf{m}_i$ with the same shape as the output vocabulary, where:

$$\mathbf{m}_i[j] = \begin{cases} 1 & \text{if } j \in C^i, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Let $\mathbf{z}_i \in \mathbb{R}^{|W_{RQ}|}$ represent the logits produced at decoding step i. During each decoding step, we apply the corresponding mask $\mathbf{m}_i$ to invalidate irrelevant indices by replacing them with $-\infty$:

$$\mathbf{z}'_i[j] = \begin{cases} \mathbf{z}_i[j] & \text{if } \mathbf{m}_i[j] = 1, \\ -\infty & \text{if } \mathbf{m}_i[j] = 0. \end{cases} \tag{5}$$

Let us consider the docid $1{-}2$, which has length $M = 2$, and assume that there are $K = 4$ centroids per codebook. In the first decoding step, we mask the logits corresponding to the last 4 centroids in $W_{RQ} \in \mathbb{R}^{2\times 4}$ with $-\infty$ then apply softmax to generate the first code in the docid 1. Similarly, in the next decoding step, we mask the logits corresponding to the first 4 centroids in $W_{RQ} \in \mathbb{R}^{2\times 4}$ with $-\infty$ then apply softmax to generate the second code in the docid 2.

Learning compatible RQ-based docids embeddings. To enhance the CL performance of MixLoRA-DSI, we employ the slow-learner strategy (Zhang et al., 2023) by scaling down the gradient updates of the output vocabulary W_{RQ}, while keeping the original portion W_{vocab} frozen. Since we have access to the previous model parameters θ_{t-1} before indexing a new corpus, we regularize the updates to the RQ embeddings using KL divergence, loosely aligning the posterior predictive distribution of the current model with that of the previous model at each decoding step:

$$\mathcal{L}_{\text{KL}}(q, id; \theta_t, \theta_{t-1}) = \frac{1}{M}\sum_{i=1}^{M} D_{KL}(P(c_i|c_{<i}, q; \theta_{t-1}) \parallel P(c_i|c_{<i}, q; \theta_t)). \tag{6}$$

3.3 Optimization Objective

The optimization objective for MixLoRA-DSI is:

$$\mathcal{L}(Q_{D_t}^{\text{retrieval}}; \theta_t, \theta_{t-1}) = \sum_{i=1}^{|Q_{D_t}^{\text{retrieval}}|} \mathcal{L}_{\text{CE}}(q_i, id_{q_i}; \theta_t) + \alpha_1 \sum_{l\in L} \mathcal{L}_{\text{aux}}(h^{l-1}, R^l) + \alpha_2 \mathcal{L}_{\text{KL}}(q_i, id_{q_i}; \theta_t, \theta_{t-1}), \tag{7}$$

where $\mathcal{L}_{\text{CE}}$ is the log-softmax cross-entropy loss for seq2seq learning, L denotes the set of MixLoRA layers in DSI's decoder, h^{l-1} denotes the previous decoder block output, R^l denotes the router of l-th MixLoRA layer, and α_1, α_2 are hyperparameters. During the pre-training stage on D_0, we do not employ $\mathcal{L}_{\text{KL}}$. During continual indexing, if there is expansion, we optimize only the expanded router weights, new LoRAs, and the extended RQ docids

vocabulary weights; otherwise, we only optimize the extended RQ docids vocabulary weights (Section 3.2). All other parameters are frozen.

4 Experiments

4.1 Experimental setting

Datasets. We evaluate on Natural Questions (NQ320k) (Kwiatkowski et al., 2019) and MS-MARCO Passage (MSMARCO) (Bajaj et al., 2016). NQ320k is a small-scale GR benchmark (Sun et al., 2023; Kishore et al., 2023) with 320k query-document pairs from 108k documents, while MSMARCO is a large-scale IR benchmark with 8.8M passages and 503k queries, evaluated on its 6.9k-query development set. Following prior works (Kishore et al., 2023; Huynh et al., 2024), we simulate dynamic corpora by splitting each dataset into an initial corpus D_0 (90% of documents) and four incremental corpora D_1-D_4 (each 2.5%). Test queries are partitioned accordingly. Further details are in Appendix A.

Metrics. Following Zeng et al. (2024a,b), we use Recall@10 (R@10) and MRR@10 (M@10) to evaluate the retrieval performance on both datasets. Following previous works (Mehta et al., 2023; Huynh et al., 2024), we adopt Average Performance (AP); Backward Transfer (BWT, aka Forgetting) to measure the effect of indexing current corpus on previous indexed corpora; and Forward Transfer (FWT, aka Learning Performance) to measures the model's ability to index a new corpus. Let $P_{t,i}$ denote the retrieval performance of the model on corpus D_i after indexing corpus D_t. With $0 \leq i < t$, the CL metrics are defined as follows:

$$\text{AP}_t = \frac{1}{t}\sum_{i=0}^{t} P_{t,i}; \quad \text{FWT}_t = \frac{1}{t}\sum_{i=1}^{t} P_{i,i};$$

$$\text{BWT}_t = \frac{1}{t-1}\sum_{i=1}^{t-1} \max_{i' \in \{0,\ldots,t-1\}} (P_{i',i} - P_{t,i}).$$

Baselines and model variants.

- **Traditional IR models**: (1) **BM25** (Robertson et al., 2009) is a strong sparse retrieval baseline. (2) **DPR** (Karpukhin et al., 2020) is a popular BERT-based dual-encoder.
- **GR models**: **BASE** uses RQ-based docids (Zeng et al., 2024a) and is sequentially fine-tuned on new corpora. This setup is equivalent to RQ-based docid **Ultron** (Zhou et al., 2022).
- **Continual Learning Generative Retrieval (CLGR) models**: we compare CLGR models in terms of the non-rehearsal CL strategies which starts from a T5 checkpoint pre-trained on D_0. (1) **DSI++** (Mehta et al., 2023) continually fine-tunes DSI with Sharpness-Aware Minimization (Foret et al., 2020). (2) **CLEVER** (Chen et al., 2023) continually fine-tunes DSI with Elastic Weight Consolidation regularization (EWC) (Kirkpatrick et al., 2017), closely aligned with the CLEVER$^{\text{-MLE(d-)}}$ variant from the original work. (3) **PromptDSI** (Huynh et al., 2024) uses prefix-tuning (Li and Liang, 2021) and neural topic embeddings. (4) **CorpusBrain++** (Guo et al., 2024) continually fine-tunes a set of adapters (Houlsby et al., 2019) to adapt to new documents. (5) **Naive Expansion**: replaces selected T5's FFNs with 2-expert MixLoRAs, expanding by one LoRA per layer for each new corpus; uses the original MoE router and load balancing loss.
- **MixLoRA-DSI**: starts from a 2-expert MixLoRA architecture checkpoint pre-trained on D_0 with the proposed improved router, variants include: (1) **MixLoRA-DSI$^{\text{-PT}}$** starts from a normal T5 checkpoint pre-trained on D_0. (2) **MixLoRA-DSI$^{\text{-Expand}}$** does not expand any further during continual indexing. (3) **MixLoRA-DSI$^{\text{-OOD}}$** does not employ energy-based dynamic expansion.

Further implementation details are in Appendix E.

4.2 Main Results

Table 1 (NQ320k) and Table 2 (MSMARCO) show that even a non-learning baseline like BM25 suffers from performance drift; while zero-shot retrieval with DPR exhibits slight forgetting. Both highlight that forgetting is unavoidable in dynamic corpora.

NQ320k. Both GR and non-PEFT CLGR models fine-tune all parameters, exhibiting strong recency bias (Smith et al., 2023), where models overfit recent corpora and suffer catastrophic forgetting on earlier ones. While CLEVER better preserves performance on new corpora, it incurs substantial memory costs (Section 4.3) and struggles to retain D_0 performance without rehearsal.

In contrast, PEFT-based CLGR methods prioritize retaining performance on the initial corpus D_0 by freezing the pre-trained backbone, often at the expense of adaptability to new corpora. For example, CorpusBrain++ achieves the lowest forgetting (lowest BWT_4), but lacks plasticity, result-

Method	Params.↓	NQ320k (Recall@10/MRR@10) $\mathbf{M_0}$	$\mathbf{D_0}$ ↑	$\mathbf{D_1}$ ↑	$\mathbf{D_2}$ ↑	$\mathbf{D_3}$ ↑	$\mathbf{D_4}$ ↑	AP_4 ↑	BWT_4 ↓	FWT_4 ↑
Traditional IR models (For reference)										
BM25	-	61.3/42.4	60.5/41.4	54.1/34.3	60.3/40.2	79.0/48.4	67.1/47.4	64.2/42.3	1.2/0.4	66.1/42.7
DPR	-	70.4/51.7	69.6/50.7	68.3/47.0	66.2/49.6	67.7/48.5	70.8/52.9	68.5/49.7	0.9/0.9	68.7/50.1
Generative Retrieval (GR) models										
BASE	235.4 M	83.0/70.4	18.3/2.5	33.3/5.6	45.6/11.2	67.7/40.8	90.6/83.7	51.1/28.8	47.7/66.3	91.0/84.9
Non PEFT-based Continual Learning Generative Retrieval (CLGR) models										
DSI++	235.4 M	83.0/70.4	29.7/5.6	41.7/10.1	48.5/17.5	45.9/66.1	85.6/80.3	50.3/35.9	45.6/55.2	87.9/82.7
CLEVER(n=512)	235.4 M	83.0/70.4	31.0/6.1	56.3/26.4	57.4/30.3	74.2/49.8	**85.9/81.6**	61.0/38.9	35.0/53.6	**90.6/84.8**
CLEVER(n=1024)	235.4 M	83.0/70.4	40.4/11.1	**67.2/46.3**	62.5/40.2	74.2/58.5	**85.9**/81.1	66.1/47.4	28.2/42.5	90.2/84.4
PEFT-based Continual Learning Generative Retrieval (CLGR) models										
PromptDSI	4.8 M	83.0/70.4	59.6/44.5	28.0/7.9	36.0/16.5	38.7/27.6	51.6/43.0	42.8/27.9	17.7/14.6	50.6/32.1
CorpusBrain++	0.8 M	83.0/70.4	6.6/1.9	8.7/3.2	14.7/8.5	9.7/8.9	29.6/23.1	13.9/9.1	**9.9/5.6**	29.6/23.1
Naive Expansion	1.6 M	83.0/70.4	48.7/21.0	26.3/6.5	33.1/12.0	43.5/24.6	69.3/62.1	44.2/25.2	38.0/50.0	72.5/64.0
MixLoRA-DSI$^{\text{-PT}}$	1.2 M	83.0/70.4	**68.8**/44.0	52.4/24.7	43.4/20.4	46.8/35.4	58.1/52.0	53.9/35.3	14.1/19.3	60.8/45.8
MixLoRA-DSI$^{\text{-Expand}}$	**0.6 M**	80.5/68.1	**68.8/51.2**	52.7/32.6	62.5/45.8	72.6/63.7	75.8/71.5	66.5/53.0	13.8/17.2	76.7/66.3
MixLoRA-DSI$^{\text{-OOD}}$	1.6 M	80.5/68.1	65.1/45.6	55.2/35.9	**72.1/55.6**	**75.8/70.7**	75.8/73.3	**68.8/56.2**	12.4/18.3	78.3/71.6
MixLoRA-DSI	0.9 M	80.5/68.1	66.1/47.2	54.1/33.8	68.4/52.7	**75.8**/69.1	76.2/73.0	68.1/55.2	13.0/18.1	78.0/70.0

Table 1: Models performance after indexing NQ320k's D_4, reported as {Recall@10}/{MRR@10}. Params. denotes the number of trainable parameters. For PEFT-based CLGR models, we report only the PEFT components, as RQ embeddings dominate the parameter count (i.e., 12.6 M). $\mathbf{M_0}$ shows D_0 performance of the initial checkpoint $P_{0,0}$. $\mathbf{D_i}$ represents $P_{4,i}$ on D_i's test queries. n denotes the number of samples per corpus used to approximate Fisher Information Matrix in CLEVER. Best/Second-best CLGR models are highlighted/underscored, except in $\mathbf{M_0}$.

ing in poor AP_4 and FWT_4. MixLoRA-DSI$^{\text{-PT}}$ outperforms Naive Expansion, PromptDSI, and CorpusBrain++ in terms of AP, demonstrating the effectiveness of MixLoRA layers with improved routers even without pre-training. Despite fine-tuning fewer than 2 M parameters, MixLoRA-DSI variants exceed CLEVER in AP_4 while suffering much less from forgetting. MixLoRA-DSI$^{\text{-Expand}}$ further highlights the importance of expansion for incorporating novel information. With layer-wise OOD-driven dynamic expansion, MixLoRA-DSI achieves sublinear parameter growth: using just 60% of MixLoRA-DSI$^{\text{-OOD}}$'s trainable parameters while maintaining over 98% of AP_4 and FWT_4 and better preserve performance on D_0.

MSMARCO. Scaling to MSMARCO presents greater challenges due to its massive size. However, the overall trend mirrors NQ320k: non-PEFT methods suffer from severe forgetting, with CLEVER reducing this at the cost of substantial memory overhead. Interestingly, CorpusBrain++ demonstrates significant robustness to forgetting. We hypothesize this is due to its continual fine-tuning of adapters with massive amounts of new data, which contrasts with MixLoRA-DSI's freeze-and-expand strategy. Our dynamic expansion strategy remains effective at controlling parameter growth. For MSMARCO, MixLoRA-DSI coincides with the non-expansion variant, outperforms PromptDSI and MixLoRA-DSI$^{\text{-OOD}}$ in terms of AP_4, ranking second only to CLEVER(n=1024), while exhibiting significantly lower forgetting. We attribute this gap to full-model updates providing greater capacity for memorizing large corpora.

4.3 Discussions

Memory comparison. In Table 3, we compare the memory usage of CLGR models. CLEVER incurs high memory costs in both storage and GPU usage due to EWC regularization, which scales with n (i.e., the number of prior samples used to estimate parameter importance via log-likelihood gradients). While a larger n helps mitigate forgetting, it significantly increases memory overhead. CLEVER must also store previous model checkpoints and their corresponding gradients. CorpusBrain++ is efficient during training, but its adapters require high memory during inference. In contrast, MixLoRA-DSI is slightly more memory-intensive during training but significantly more efficient during inference. It requires saving only the latest model and a temporary copy during training to compute $\mathcal{L}_{\text{KL}}$ (Eq. 6).

Ablation studies. Table 4 shows ablations of MixLoRA-DSI. The RQ-based docid mask provides a strong performance boost in AP_4 (+8.0 Recall@10) but does not mitigate forgetting. Incorporating RQ-based docid CL strategies further enhances retrieval (+8.4 Recall@10) while significantly reducing forgetting (-20.0 Recall@10). The

Method	Params.↓	MSMARCO (Recall@10/MRR@10) $\mathbf{M_0}$	$\mathbf{D_0}$ ↑	$\mathbf{D_1}$ ↑	$\mathbf{D_2}$ ↑	$\mathbf{D_3}$ ↑	$\mathbf{D_4}$ ↑	AP_4 ↑	BWT_4 ↓	FWT_4 ↑
Traditional IR models (For reference)										
BM25	-	39.8/19.1	38.4/18.2	39.1/19.0	38.8/19.4	32.8/13.5	40.3/19.3	37.9/17.9	1.0/0.5	38.4/18.1
DPR	-	53.2/26.0	51.6/24.6	49.5/26.2	54.1/25.4	48.3/22.3	49.4/24.5	50.6/24.6	0.5/0.7	50.4/24.9
Generative Retrieval models										
BASE	235.4 M	34.3/17.7	7.5/1.4	8.2/2.5	7.7/1.2	22.4/5.9	87.7/69.1	26.7/16.0	53.6/55.2	78.3/70.8
Non PEFT-based Continual Learning Generative Retrieval (CLGR) models										
DSI++	235.4 M	34.3/17.7	7.6/1.6	6.5/1.7	6.1/0.8	17.8/4.9	**90.3/68.0**	25.7/15.4	59.1/50.9	**82.6/65.7**
CLEVER(n=512)	235.4 M	34.3/17.7	14.4/4.1	25.5/9.9	19.9/7.6	27.6/12.6	81.2/67.6	33.7/20.4	45.1/43.2	78.6/64.2
CLEVER(n=1024)	235.4 M	34.3/17.7	18.7/6.2	**41.3/23.0**	**29.6/13.8**	31.6/13.0	70.1/60.2	**38.3/23.2**	33.1/36.0	72.4/60.6
PEFT-based Continual Learning Generative Retrieval (CLGR) models										
PromptDSI	101.5 M	34.3/17.7	17.6/7.5	18.5/8.6	12.8/5.1	14.4/5.1	39.0/34.3	20.4/12.1	32.0/31.3	48.9/42.0
Naive Expansion	1.6 M	34.3/17.7	24.5/10.4	20.7/8.5	17.3/6.5	24.7/10.7	65.6/56.0	30.6/18.4	28.5/30.8	58.2/49.4
CorpusBrain++	0.8 M	34.3/17.7	**28.9/15.4**	30.4/15.0	28.6/11.7	28.7/15.6	24.3/40.9	28.2/19.7	**3.8/4.4**	30.1/24.6
MixLoRA-DSI^{PT}	1.2 M	34.3/17.7	25.0/12.1	26.1/12.0	20.9/7.5	27.6/14.2	47.4/41.6	29.4/17.5	14.4/18.1	42.6/35.5
MixLoRA-DSI^{OOD}	1.6 M	34.0/17.8	27.9/13.6	29.9/13.4	25.5/10.8	33.3/17.7	58.4/44.5	35.0/20.0	16.6/19.8	51.9/40.3
MixLoRA-DSI	**0.6 M**	34.0/17.8	28.1/13.6	31.0/13.5	29.1/11.4	**34.5/18.2**	59.1/46.3	36.4/20.6	16.5/20.4	53.5/41.7

Table 2: Models performance after indexing MSMARCO's D_4. Refer to Table 1 for a detailed caption.

Method	EWC	Train	Inference	Storage
DSI++	-	10.2	21.1	**0.9**
CLEVER(n=512)	18.4	12.3	21.1	6.2
CLEVER(n=1024)	32.4	12.3	21.1	6.2
PromptDSI	-	**6.3**	18.4	**0.9**
CorpusBrain++	-	**6.3**	30.9	**0.9**
MixLoRA-DSI	-	8.1	**18.0**	**0.9**

Table 3: Memory footprints (GiB) during training and inference measured with a batch size of 128; and storage. CLEVER requires significant GPU memory to approximate Fisher Information Matrix due to EWC, and demands more storage.

Methods					AP_4 ↑		BWT_4 ↓	
Mask.	CL.	Rout.	PT.	OOD.	R@10	M@10	R@10	M@10
✗	✗	✗	✗	✗	36.2	20.7	33.1	38.7
✓	✗	✗	✗	✗	44.2	25.2	35.8	49.7
✓	✓	✗	✗	✗	52.5	32.8	15.8	21.1
✓	✓	✓	✗	✗	53.2	34.6	12.5	**15.3**
✓	✓	✓	✓	✗	**68.8**	**56.2**	**12.4**	18.3
✓	✓	✓	✓	✓	68.1	55.2	13.0	18.1

Table 4: Ablation results on NQ320k for MixLoRA-DSI's design: RQ-based docid mask (Mask.), RQ-based docid CL strategies (CL.), Improved router (Rout.), Pre-training on D_0 (PT.), and OOD-driven dynamic expansion (OOD.). Please refer to Appendix B.2 for ablation results on MSMARCO.

improved router further increases overall performance. Pre-training on D_0 has the most substantial impact on retrieval (+15.6 Recall@10), underscoring the importance of a strong initialization. OOD-driven expansion limits unnecessary growth while preserving performance. Overall, all components contribute to the strong stability-plasticity trade-off of MixLoRA-DSI.

Token routing analysis. Figure 3 analyzes the routing behavior of the original and improved routers. Naive Expansion starts with 2-expert MixLoRAs on D_1 and linearly adds new LoRAs per corpus. However, its softmax router, with unnormalized weights and inputs, induces competition, causing new tokens to be predominantly assigned to recently added experts. In contrast, our improved router, guided by a novel auxiliary loss, achieves a more balanced routing distribution while still prioritizing newly added LoRAs (e.g., the 5th for D_4). This demonstrates that our approach effectively preserves prior experts' knowledge while enabling new LoRAs to capture novel information.

Layer-wise OOD queries analysis. Table 5 presents the percentage of OOD queries. In the left-half, MixLoRA-DSI, with five 2-expert MixLoRA layers in the decoder, expands only once before D_1, resulting in total 15 LoRAs, as pre-training on D_0 provides broad general knowledge. In contrast, MixLoRA-DSI^{PT} (right-half) starts from a pre-trained T5 checkpoint, initializing all LoRAs and routers from scratch with no prior energy scores for OOD detection. Consequently, OOD estimation begins only after indexing D_1. As indexing progresses, energy scores stabilize, reducing OOD detections and slowing expansion, ensuring sublinear parameter growth.

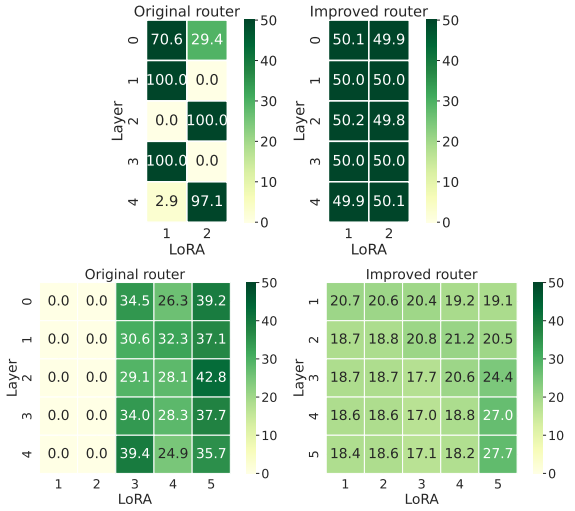

Figure 3: Token routing analysis of the original MoE routers (left) and our proposed improved routers (right) when training on D_1 (Top row) and D_4 (Bottom row). Each cell indicates the percentage of tokens being routed to a LoRA expert in a layer. Our improved routers achieve balanced routing while the original router suffers from recency bias.

Layer	MixLoRA-DSI				MixLoRA-DSI$^{\text{-PT}}$			
	D_1	D_2	D_3	D_4	D_1	D_2	D_3	D_4
1	3.6	0.0	0.0	0.0	-	81.2	55.8	22.0
2	1.6	0.0	0.0	0.0	-	43.1	19.3	5.6
3	1.2	0.0	0.0	0.0	-	33.0	6.9	7.1
4	1.3	0.0	0.0	0.0	-	35.8	5.9	5.3
5	2.0	0.0	0.0	0.0	-	47.2	14.6	2.8
# LoRA	10	15	15	15	10	15	18	19

Table 5: Percentage of layer-wise OOD queries for MixLoRA-DSI and MixLoRA-DSI$^{\text{-PT}}$ from Table 1 before training on each new corpus of NQ320k. '-' denotes cases where OOD detection are inapplicable.

5 Related Work

Generative Retrieval (GR). Docids remain a core research focus in GR (Bevilacqua et al., 2022; Li et al., 2023; Sun et al., 2023; Zhang et al., 2024; Zeng et al., 2024b; Liu et al., 2024). Recent works improve GR through learning-to-rank (Li et al., 2024; Zeng et al., 2024a) and multi-graded retrieval (Tang et al., 2024); however, these methods are not designed for dynamic corpora. GR over dynamic corpora is gaining traction (Mehta et al., 2023; Chen et al., 2023; Guo et al., 2024), with recent efforts addressing rehearsal-free CL (Huynh et al., 2024) and temporal IR (Kim et al., 2024). We note that related works such as IncDSI (Kishore et al., 2023), which differs from autoregressive GR, and L^{2}R (Cai et al., 2023), a rehearsal-based dense retrieval method, fall outside the scope of this work. MixLoRA-DSI pushes dynamic corpora GR towards task-agnostic, rehearsal-free, and scalable retrieval by leveraging RQ-based docids (Zeng et al., 2024a), introducing PEFT, and employing a dynamic expansion strategy. This challenges prior works that primarily adopt atomic docids and evaluate on small-scale datasets.

Parameter-efficient Fine-tuning (PEFT). PEFT introduces lightweight trainable components to efficiently train large models. The three major methods are Adapters (Houlsby et al., 2019), Prompts (Lester et al., 2021; Li and Liang, 2021), and LoRA (Hu et al., 2022). LoRA is recently integrated into MoE (Shazeer et al., 2017; Fedus et al., 2022), results in MixLoRA and its variants in vision and NLP (Dou et al., 2024; Yu et al., 2024; Wu et al., 2024). However, its application in IR remains unexplored. We pioneer its use in dynamic corpora GR, introducing IR-specific enhancements: an improved router with novel auxiliary loss and an energy-based dynamic expansion strategy to address the challenges of dynamic corpora.

6 Conclusion

We introduce MixLoRA-DSI, the first dynamically expandable framework for rehearsal-free GR over dynamic corpora, which is task-agnostic, rehearsal-free, and scalable. Our layer-wise OOD-driven expansion strategy ensures sublinear parameter growth, and the proposed cosine classifier with novel auxiliary loss improves expert specialization and balance routing. Integrating with CL strategies for RQ-based docids, MixLoRA-DSI outperforms prior GR methods in efficiency and robustness.

Limitations

Our work does not incorporate recent ranking optimization techniques (Li et al., 2024; Zeng et al., 2024a), as state-of-the-art RQ-based docid methods often require multi-step training (Zeng et al., 2024a,b), which may exacerbate forgetting in dynamic corpora. Consequently, all GR models lag behind traditional IR on MSMARCO in this realistic setting. Due to resource constraints, we experiment only with T5-base. Although prior work (Pradeep et al., 2023) suggests that GR performance does not scale linearly with model size, exploring larger models is left for future work. We

also do not perform extensive hyperparameter tuning, yet observe consistent gains from each component. While we adopt a freeze-and-expand strategy, CorpusBrain++ shows that continually fine-tuning a fixed set of adapters yields strong robustness to forgetting. This raises an open question about the trade-off between dynamic expansion and parameter budget. We further acknowledge the need to assess generalization across diverse retrieval tasks (e.g., KILT (Petroni et al., 2021)) but defer this to future work due to space constraints. Finally, GR is not suitable for zero-shot retrieval, as indexing necessitates model updates. Addressing these challenges offers promising directions for future work.

Acknowledgments

This material is based on research sponsored by Defense Advanced Research Projects Agency (DARPA) under agreement number HR0011-22-2-0047. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government. This work is supported by the Australian Research Council Discovery Early Career Researcher Award DE250100032 and Monash eResearch capabilities, including M3. Trung Le was partly supported by the Air Force Office of Scientific Research under award number FA2386-23-1-4044.

References

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. 2019. Task-free continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11254–11263.

Randy Ardywibowo, Zepeng Huo, Zhangyang Wang, Bobak J Mortazavi, Shuai Huang, and Xiaoning Qian. 2022. Varigrow: Variational architecture growing for task-agnostic continual learning based on bayesian novelty. In *International Conference on Machine Learning*, pages 865–877. PMLR.

Jimmy Lei Ba. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Artem Babenko and Victor Lempitsky. 2014. Additive quantization for extreme vector compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 931–938.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, and 1 others. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings as document identifiers. *Advances in Neural Information Processing Systems*, 35:31668–31683.

Yinqiong Cai, Keping Bi, Yixing Fan, Jiafeng Guo, Wei Chen, and Xueqi Cheng. 2023. L2r: Lifelong learning for first-stage retrieval with backward-compatible representations. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 183–192.

Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, Yixing Fan, and Xueqi Cheng. 2023. Continual learning for generative retrieval over dynamic corpora. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 306–315.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, and 1 others. 2024. Loramoe: Alleviating world knowledge forgetting in large language models via moe-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Tobias Fink, Florina Piroi, Alaa El-Ebshihy, Jüri Keller, Romain Devaud, David Iommi, Petra Galuščáková, Gabriela Gonzalez-Saez, Philippe Mulhem, Lorraine Goeuriot, and et al. 2025. Longeval 2025 web retrieval collection.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*.

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP*

(RepL4NLP-2021), pages 316–321, Online. Association for Computational Linguistics.

Spyros Gidaris and Nikos Komodakis. 2018. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4367–4375.

Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*.

Jiafeng Guo, Changjiang Zhou, Ruqing Zhang, Jiangui Chen, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. 2024. Corpusbrain++: A continual generative pre-training framework for knowledge-intensive language tasks. *arXiv preprint arXiv:2402.16767*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Tuan-Luc Huynh, Thuy-Trang Vu, Weiqing Wang, Yinwei Wei, Trung Le, Dragan Gasevic, Yuan-Fang Li, and Thanh-Toan Do. 2024. Promptdsi: Prompt-based rehearsal-free instance-wise incremental learning for document retrieval. *arXiv preprint arXiv:2406.12593*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Chaeeun Kim, Soyoung Yoon, Hyunji Lee, Joel Jang, Sohee Yang, and Minjoon Seo. 2024. Exploring the practicality of generative retrieval on dynamic corpora. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Miami, Florida, USA. Association for Computational Linguistics.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Varsha Kishore, Chao Wan, Justin Lovelace, Yoav Artzi, and Kilian Q. Weinberger. 2023. Incdsi: Incrementally updatable document retrieval. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. 2020. A neural dirichlet process mixture model for task-free continual learning. In *International Conference on Learning Representations*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023. Multiview identifiers enhanced generative retrieval. *arXiv preprint arXiv:2305.16675*.

Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2024. Learning to rank in generative retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8716–8723.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pages 2356–2362.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475.

Yuxuan Liu, Tianchi Yang, Zihan Zhang, Minghui Song, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. 2024. Asi++: Towards distributionally balanced end-to-end generative retrieval. *arXiv preprint arXiv:2405.14280*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Sanket Mehta, Jai Gupta, Yi Tay, Mostafa Dehghani, Vinh Tran, Jinfeng Rao, Marc Najork, Emma Strubell, and Donald Metzler. 2023. DSI++: Updating transformer memory with new documents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8198–8213, Singapore. Association for Computational Linguistics.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. 2019. Do deep generative models know what they don't know? In *International Conference on Learning Representations*.

Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to doctttttquery. *Online preprint*, 6(2).

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

Ronak Pradeep, Kai Hui, Jai Gupta, Adam Lelkes, Honglei Zhuang, Jimmy Lin, Donald Metzler, and Vinh Tran. 2023. How does generative retrieval scale to millions of passages? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1305–1321.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321.

James Seale Smith, Junjiao Tian, Shaunak Halbe, Yen-Chang Hsu, and Zsolt Kira. 2023. A closer look at rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2409–2419.

Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten Rijke, and Zhaochun Ren. 2023. Learning to tokenize for generative retrieval. *Advances in Neural Information Processing Systems*, 36.

Yubao Tang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, and Xueqi Cheng. 2024. Generative retrieval meets multi-graded relevance. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, and 1 others. 2022. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35:21831–21843.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and 1 others. 2022. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 631–648. Springer.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of loRA experts. In *The Twelfth International Conference on Learning Representations*.

Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. 2024. Boosting continual learning of vision-language models via mixture-of-experts adapters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23219–23230.

Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. 2024a. Scalable and effective generative information retrieval. In *Proceedings of the ACM on Web Conference 2024*, pages 1441–1452.

Hansi Zeng, Chen Luo, and Hamed Zamani. 2024b. Planning ahead in generative retrieval: Guiding autoregressive generation through simultaneous decoding. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 469–480.

Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. 2023. Slca: Slow learner with classifier alignment for continual learning on a

pre-trained model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19148–19158.

Peitian Zhang, Zheng Liu, Yujia Zhou, Zhicheng Dou, Fangchao Liu, and Zhao Cao. 2024. Generative retrieval via term set generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 458–468.

Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen. 2022. Ultron: An ultimate retriever on corpus with a model-based indexer. *arXiv preprint arXiv:2208.09257*.

Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022. Bridging the gap between indexing and retrieval for differentiable search index with query generation. *arXiv preprint arXiv:2206.10128*.

A Datasets

The detailed statistics of NQ320k (Kwiatkowski et al., 2019) and MSMARCO (Bajaj et al., 2016) are shown in Table 6 and 7, respectively. For reproducibility, we adopt pseudo-queries provided by previous works: MSMARCO uses 10 per document (Zeng et al., 2024a), and NQ320k up to 15 (Kishore et al., 2023).

It is worth noting that MSMARCO contains 28 times more documents but only 2 times more annotated queries than NQ320k. To prevent data leakage in the dynamic corpora setting, we ensure that the splits (D_0-D_4) do not contain information from subsequent corpora, which results in fewer annotated queries, further impacting the quality of RQ-based docids construction.

B Additional results

B.1 Ablation on the choice of using RQ-based document identifiers

While MixLoRA-DSI is compatible with any semantic docid (i.e., docids derived from quantization or hierarchical clustering over document embeddings (Zeng et al., 2024a,b)), generative retrieval generally depends on high-quality docids, which are typically fixed after construction. We provide additional experiments on NQ320k to justify the choice of RQ-based docid in Table 9.

We explored another popular semantic docid representation, Product Quantization (PQ)-based docids, used in prior work such as CLEVER (Chen et al., 2023) and Ultron (Zhou et al., 2022). OnD_0 , even with exact search via FAISS (Johnson et al., 2019), PQ-based docids achieved only 13.0% Recall@10 and 6.9% MRR@10, compared to 76.6% Recall@10 and 60.0% MRR@10 with RQ-based docids. As a result, MixLoRA-DSI-PT with PQ-based docids yields poor overall performance.

Another widely used docid in continual learning for generative retrieval is the atomic docid, as adopted by DSI++ (Mehta et al., 2023), IncDSI (Kishore et al., 2023), and PromptDSI (Huynh et al., 2024). While atomic docid is less prone to forgetting, they come with a substantial memory cost: the vocabulary embedding matrix grows linearly with the number of documents. For example, in MSMARCO, this would require an approximately 25 GiB extended vocabulary to store 8.8 million unique document embeddings. This makes atomic docid impractical for large-scale applications, especially given our goal of efficiency and sub-linear parameter growth. In contrast, both PQ- and RQ-based docids use a fixed-size extended vocabulary, which adds minimal memory overhead and still manages a good stability-plasticity trade-off.

B.2 Ablation studies on MSMARCO

We provide the ablation study on MS MARCO in Table 10. The results demonstrate that each proposed component contributes to the overall stability-plasticity tradeoff, leading to the improved performance of MixLoRA-DSI, similar to the findings on NQ320k in Table 4.

B.3 Ablation studies on the auxiliary router loss

Table 13 provides a breakdown of the contribution of each term in the auxiliary loss:

Regarding NQ320k, using both terms achieves significantly lower BWT (i.e., less forgetting) while still improving AP (Average Performance) compared to not using any. In contrast, using only one term improves AP at the cost of exacerbating BWT (i.e., more forgetting). For MSMARCO, using both terms also achieves significantly lower BWT with only a minor impact on AP. In contrast, using only one term either decreases AP or increases BWT. In conclusion, both loss terms in the proposed auxiliary loss contribute meaningfully to achieving a balance stability–plasticity trade-off.

NQ320k				
Split	Document	Annotated Queries	Train Pseudo Queries	Test Queries
D_0	9,8743	276,493	1,480,538	6,998
D_1	2,743	-	41,132	357
D_2	2,743	-	41,136	136
D_3	2,743	-	41,138	62
D_4	2,743	-	41,141	277

Table 6: The NQ320k dataset statistics used in our study.

MSMARCO				
Split	Document	Annotated Queries	Train Pseudo Queries	Test Queries
D_0	7,957,640	452,746	79,576,202	6,247
D_1	221,045	-	2,210,445	184
D_2	221,045	-	2,210,445	196
D_3	221,045	-	2,210,447	174
D_4	221,048	-	2,210,471	154

Table 7: The MSMARCO dataset statistics used in our study.

B.4 Ablation studies on optimization objective's hyperparameters

We conduct experiments to study the impact of different α_1 (i.e., the auxiliary router loss) and α_2 (i.e., KL divergence to regularize the updates to the RQ embeddings) combinations in Equation (7) on NQ320k.

In practical continual learning settings, preserving past knowledge (lower BWT) is often as critical as learning new knowledge (higher AP). The results show a trade-off between AP (Table 11) and BWT (Table 12): generally, a larger α_1 slightly boosts AP performance but lowers BWT, while a larger α_2 negatively impacts AP but reduces BWT. Our default choice of $\alpha_1 = 1.0$ and $\alpha_2 = 0.1$ strikes a balance between the two metrics, achieving the best AP while maintaining relatively low BWT on both Recall@10 and MRR@10.

B.5 Additional results on LongEval

As requested by the reviewers during the rebuttal and due to limited computational resources and time constraints, we conducted experiments on the LongEval Challenge (Fink et al., 2025) using 3 training epochs for selected methods. Following the DSI++ benchmark setup, we selected four timesteps (October 2022 to January 2023) to simulate a continual learning scenario. We could not use the latest LongEval 2025 test set, as it had not been released during the rebuttal period, nor could we access the 2024 test set.

Details of the dataset used in our experiments are summarized in Table 8. Each new split contains only newly added unique documents. The annotated queries are divided into an 80-20 train-test split, with only the annotated training queries from 2022-10 used to construct the RQ-based docids.

We compare our proposed methods against CLEVER, the strongest continual learning baseline for generative retrieval, in Table 14. Our findings on LongEval are consistent with trends observed on NQ320k and MSMARCO. Despite being rehearsal-free, MixLoRA-DSI$^{\text{PT}}$ achieves competitive performance relative to CLEVER (n = 512) with EWC regularization. Furthermore, pre-training the experts substantially boosts MixLoRA-DSI's performance, yielding the best results in this setup.

C More details about residual quantization-based docids

We adopt Residual Quantization (RQ)-based docid (Zeng et al., 2024a). RQ-based docid is the first docid that is effective for large-scale standard IR datasets (Bajaj et al., 2016). while most GR works have only shown competitive performance on smaller-scale benchmarks (Pradeep et al., 2023). For each document $d \in D$, we obtain its document embedding by treating the pre-trained backbone of DSI as a dense encoder:

$$e_d = \text{Decoder}(\text{<bos>}; \text{Encoder}(d)) \in \mathbb{R}^{\text{dim}}. \quad (8)$$

After obtaining the set of document embeddings $E_D = \{e_{d_1}, \ldots, e_{d_{|D|}}\}$, we train RQ with M codebooks, each containing K centroids (i.e., codewords), where the m-th codebook is defined as

LongEval				
Split	Document	Annotated Queries	Train Pseudo Queries	Test Queries
D_0 (2022-10)	2,137,584	9,005	8,790,469	1,003
D_1 (2022-11)	49,628	-	248,140	1,127
D_2 (2022-12)	468,940	-	2,344,700	1,128
D_3 (2023-01)	35,777	-	178,885	1,163

Table 8: The LongEval dataset statistics used in our study.

Docid variant	AP ↑		BWT ↓		Extended vocab size ↓
	Recall@10	MRR@10	Recall@10	MRR@10	MiB
Atomic	44.4	24.1	**0.0**	**0.0**	318
PQ-based	3.4	1.6	1.0	1.5	**48**
RQ-based	**53.2**	**34.6**	12.6	15.5	**48**

Table 9: MixLoRA-DSI$^{-\text{PT}}$ with different docid variants on D_0 of NQ320k.

Methods					AP_4 ↑		BWT_4 ↓	
Mask.	CL.	Rout.	PT.	OOD.	R@10	M@10	R@10	M@10
✗	✗	✗	✗	✗	29.4	17.5	**14.4**	18.1
✓	✗	✗	✗	✗	30.6	18.4	28.5	30.8
✓	✓	✗	✗	✗	34.3	20.0	17.3	20.1
✓	✓	✓	✗	✗	34.5	19.7	15.0	**17.9**
✓	✓	✓	✓	✗	35.7	20.2	16.9	20.3
✓	✓	✓	✓	✓	**36.4**	**20.6**	16.5	20.4

Table 10: Ablation results on MSMARCO for MixLoRA-DSI's design: RQ-based docid mask (Mask.), RQ-based docid CL strategies (CL.), Improved router (Rout.), Pre-training on D_0 (PT.), and, OOD-driven dynamic expansion (OOD.).

AP	$\alpha_2 = 0.05$		$\alpha_2 = 0.1$		$\alpha_2 = 0.5$	
Metrics	R@10	M@10	R@10	M@10	R@10	M@10
$\alpha_1 = 0.05$	65.2	50.8	66.7	52.3	59.5	40.6
$\alpha_1 = 0.1$	64.9	50.2	67.0	52.3	59.7	40.8
$\alpha_1 = 0.5$	66.4	51.8	**68.1**	**55.2**	61.9	45.3

Table 11: Average performance (AP) of MixLoRA-DSI on NQ320k under different combinations of α_1 and α_2.

$C^m := \{c_k\}_{k=1}^K \in \mathbb{R}^{\text{dim}\times K}$. We further denote the k-th centroid in the m-th codebook as $C^m[k] \in \mathbb{R}^{\text{dim}}$. The RQ codebooks are optimized to approximate the document embeddings:

$$e_d \approx \sum_{m=1}^{M} C^m[i_m], \quad i_m \in [1, K]. \tag{9}$$

The trained RQ codebooks is used to generate an RQ code of length M for each document d. The learned RQ centroids are concatenated to the DSI's vocabulary weights $W_{vocab} \in \mathbb{R}^{\text{dim}\times|V|}$ with length $|V|$ for subsequent optimization: $W_{\text{RQ}} = \{W_{\text{vocab}}; C^1; \ldots; C^M\} \in \mathbb{R}^{\text{dim}\times(|V|+M*K)}$, where $\{\cdot;\cdot\}$ is concatenation.

BWT	$\alpha_2 = 0.05$		$\alpha_2 = 0.1$		$\alpha_2 = 0.5$	
Metrics	R@10	M@10	R@10	M@10	R@10	M@10
$\alpha_1 = 0.05$	15.3	21.0	13.3	20.6	12.9	15.2
$\alpha_1 = 0.1$	18.3	26.2	12.8	17.7	**8.0**	**5.0**
$\alpha_1 = 0.5$	17.6	28.0	13.0	18.1	9.9	7.2

Table 12: Forgetting (BWT) of MixLoRA-DSI on NQ320k under different combinations of α_1 and α_2.

Methods		AP_4 ↑		BWT_4 ↓	
First term	Second term	R@10	M@10	R@10	M@10
✗	✗	36.2	20.7	33.1	38.7
✓	✗	44.2	25.2	35.8	49.7
✗	✓	52.5	32.8	15.8	21.1
✓	✓	68.1	55.2	13.0	18.1

Table 13: Ablation results on the contribution of each term in the proposed auxiliary router loss (Eq. 3).

D MixLoRA: Mixture of LoRA Experts

In this section, we elaborate the our implementation of MixLoRA in MixLoRA-DSI. We extend the MoE formulation in Section 2.2 by introducing LoRAs as experts, namely MixLoRA. We replace selected Feed-Forward Network (FFN) layers of DSI with MixLoRA. Consider a T5-based (Raffel et al., 2020) DSI model, the Feed-Forward Network (FFN) is a two-layer Multi-Layer Perceptron (MLP) that processes an input x. For simplicity, we omit dropout and activation functions without

Method	LongEval (Recall@10/MRR@10) $D_0 \uparrow$	$D_1 \uparrow$	$D_2 \uparrow$	$D_3 \uparrow$	$AP_3 \uparrow$	$BWT_3 \downarrow$	$FWT_3 \uparrow$
CLEVER(n=512)	5.7/3.2	6.8/4.2	6.2/4.5	9.9/8.5	7.1/5.1	2.0/4.9	8.9/8.3
CLEVER(n=1024)	6.4/5.0	8.6/7.4	6.8/5.9	9.5/9.1	7.8/6.8	1.3/2.7	9.1/8.5
MixLoRA-DSI^{PT}	6.4/6.7	8.4/8.5	7.0/7.5	9.3/9.4	7.3/6.0	**1.1/1.8**	8.2/6.4
MixLoRA-DSI	**10.5/11.5**	**11.1/11.5**	**10.2/10.9**	**11.7/11.6**	**10.9/11.4**	1.4/2.2	**11.8/12.4**

Table 14: Models' performance after indexing LongEval's D_3.

loss of generality:

$$\text{FFN}(x) = x + W_{\text{out}}(W_{\text{in}}(\text{LN}(x))), \tag{10}$$

where $\text{LN}(\cdot)$ denotes layer normalization (Ba, 2016); W_{in} and W_{out} are the input and output weight matrix, respectively. In MixLoRA, each layer of the FFN is adapted with a corresponding set of LoRAs[2], while keeping the original layer weight frozen. Let $\mathbb{L}_{\text{in}} = \{\Delta^i_{\text{in}}\}^N_{i=1}$ and $\mathbb{L}_{\text{out}} = \{\Delta^i_{\text{out}}\}^N_{i=1}$ denote the sets of N LoRAs with $N \geq 2$ for adapting W_{in} and W_{out}, respectively. The output of adapting W_{in} is:

$$x' = W_{\text{in}}(\text{LN}(x)) + \sum_{i=1}^{N} Topk(p_i(x))\Delta^i_{\text{in}}(x), \tag{11}$$

The final output of MixLoRA is computed as:

$$\text{MixLoRA}(x) = x + W_{\text{out}}(x') + \sum_{i=1}^{N} Topk(p_i(x))\Delta^i_{\text{out}}(x'). \tag{12}$$

By sharing the gate values across both MLP layers, we avoid the need for another router.

E Implementation details

All experiments are conducted on a single NVIDIA A100 80GB GPU, using the AdamW optimizer (Loshchilov and Hutter, 2017) with a weight decay of 0.01 and learning rate of $1e^{-3}$. We use linear learning rate schedule with 10% warm-up steps, and gradient norm clipping of 1.0. We use T5-base (Raffel et al., 2020) as the backbone for all generative retrieval (GR) models, initializing pre-trained checkpoints from Huggingface (Wolf et al., 2020). The maximum input sequence length is 256. For constrained beam search inference, we use a beam size of 10 and decode for up to 8 steps.

The initial checkpoint is prepared following the first two steps of RIPOR's implementation (Zeng et al., 2024a) docid initialization and Seq2Seq pretraining. We construct RQ-based docids using RQ codebooks with $M = 8$ and $K = 2048$ centroids per codebook. The initial checkpoint M_0 is trained with a batch size of 1024 and a learning rate of 0.001. Training runs for 20K steps on NQ320k and 250K steps on MSMARCO.

E.1 Traditional IR Models

- **BM25:** We use FAISS (Johnson et al., 2019) to index D_0 and retrieve documents using Pyserini's BM25 implementation (Lin et al., 2021).

- **DPR:** We train a BERT-based dual-encoder DPR (Devlin et al., 2019; Karpukhin et al., 2020) on D_0 and use it for indexing and retrieval on new corpora with FAISS. The implementation follows Gao et al. (2021), using the BERT-base checkpoint from Huggingface.

E.2 Generative Retrieval Models

We evaluate various GR models in our rehearsal-free dynamic corpus setting. During continual indexing, we train all models with with a learning rate of 0.001. We use a batch size of 128 and 512 for NQ320k and MSMARCO, respectively. Starting from M_0, GR and Non PEFT-based CLGR models are trained for 5 epochs on NQ320k and MSMARCO; PEFT-based CLGR methods and PEFT are trained for 10 epochs on NQ320k and 5 epochs on MSMARCO.

- **BASE:** We reproduce using the code from Zeng et al. (2024b,a).

- **DSI++:** We exclude generative replay, as it requires training an additional model on D_0, violating the rehearsal-free constraint. Since the source code is unavailable, we implement SAM (Foret et al., 2020) using an open-source repository[3].

[2]Compared to Yu et al. (2024), our formulation uses two unique sets of LoRAs for each FFN layer

[3]https://github.com/davda54/sam

- **CLEVER:** We experiment with previous sample sizes $n = 512$ and $n = 1024$ for EWC regularization. We implement EWC based on open-source materials[4].
- **CorpusBrain++:** We employ bottleneck adapters[5] to all encoder and decoder blocks of T5. The reduction factor is set to 96 to match the rank-8 LoRAs used in MixLoRA-DSI.
- **PromptDSI:** since the source code is not released, we implement it according to the paper. Neural topic embeddings are obtained from BERTopic (Grootendorst, 2022), using M_0 as the encoder to prevent data leakage during continual indexing.

E.3 MixLoRA-DSI

For continual indexing, we use LoRAs with rank 8, a dropout rate of 0.05, and a scaling factor of 16. We build on an open-source MoE implementation[6]. We find $\alpha_1 = 1, \alpha_2 = 0.1$ to be robust (Appendix B.4). Regarding slow-learner in the RQ-based docids embeddings, we scale down the gradient of concatenated RQ token weights by a factor of 100.

Following insights from prior PEFT studies (Wang et al., 2022; Huynh et al., 2024), we apply MixLoRA to the first five layers of the decoder, as it is responsible for autoregressive decoding. By default, MixLoRA employs a top-2 router, initializing with two unique sets of LoRAs, discussed in Appendix D. The token-wise energy score OOD thresholds τ are set to the average exponential moving average of energy scores.

For NQ320k, all pre-trained MixLoRA-DSI variants use a batch size of 32, while other variants use a batch size of 128. All models are trained for 10 epochs. The layer-wise OOD query threshold δ is set to 1% for pre-trained variants and 5% for MixLoRA-DSI$^{\text{PT}}$.

For MSMARCO, MixLoRA-DSI variants use a batch size of 512 and are trained for 5 epochs. The layer-wise OOD query threshold δ is set to 0.01% for pre-trained variants and 10% for MixLoRA-DSI$^{\text{PT}}$.

The naive expansion baseline follows the same setup as MixLoRA-DSI but without pre-training on D_0, expanding naively, and using the original MoE router with original load balancing loss.

[4] https://github.com/ContinualAI/colab/blob/master/notebooks/intro_to_continual_learning.ipynb

[5] https://docs.adapterhub.ml/methods.html#bottleneck-adapters

[6] https://github.com/lucidrains/st-moe-pytorch