A Parametric Memory Head for Continual Generative Retrieval

Kidist Amde Mekonnen
University of Amsterdam
Amsterdam, The Netherlands
k.a.mekonnen@uva.nl

Yubao Tang
University of Amsterdam
Amsterdam, The Netherlands
y.tang3@uva.nl

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
m.derijke@uva.nl

Abstract

Generative information retrieval (GenIR) consolidates retrieval into a single neural model that decodes document identifiers (docids) directly from queries. While this *model-as-index* paradigm offers architectural simplicity, it is poorly suited to dynamic document collections. Unlike modular systems, where indexes are easily updated, GenIR's knowledge is parametrically encoded in its weights; consequently, standard adaptation methods such as full and parameter-efficient fine-tuning can induce catastrophic forgetting. We show that sequential adaptation improves retrieval on newly added documents but substantially degrades performance on earlier slices, exposing a pronounced stability–plasticity trade-off. To address this, we propose *post-adaptation memory tuning* (PAMT), a memory-only stabilization stage that augments an adapted model with a modular parametric memory head (PMH). PAMT freezes the backbone and attaches a product-key memory with fixed addressing. During prefix-trie constrained decoding, decoder hidden states sparsely query PMH to produce residual corrections in hidden space; these corrections are mapped to score adjustments via the frozen output embedding matrix, computed only over trie-valid tokens. This guides docid generation while keeping routing and backbone parameters fixed. To limit cross-slice interference, PAMT updates only a fixed budget of memory values selected using decoding-time access statistics, prioritizing entries frequently activated by the current slice and rarely used in prior sessions. Experiments on MS MARCO and Natural Questions under sequential, disjoint corpus increments show that PAMT substantially improves retention on earlier slices with minimal impact on retrieval performance for newly added documents, while modifying only a sparse subset of memory values per session.

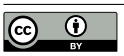
CCS Concepts

• **Information systems** → **Information retrieval**; **Retrieval models and ranking**; *Language models*.

Keywords

Generative information retrieval, Dynamic indexing, Continual learning, Catastrophic forgetting, Sparse memory

ACM Reference Format:
Kidist Amde Mekonnen, Yubao Tang, and Maarten de Rijke. 2026. A Parametric Memory Head for Continual Generative Retrieval. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26), July 20–24, 2026, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3805712.3809725



SIGIR '26, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-2599-9/2026/07
https://doi.org/10.1145/3805712.3809725

1 Introduction

Generative information retrieval (GenIR) reformulates document retrieval as an autoregressive generation task, mapping natural language queries directly to unique document identifiers (docids) [32, 45, 46]. By internalizing corpus knowledge in model parameters, GenIR enables end-to-end optimization, substantial index compression, and reuse of semantic priors from pretrained language models [31, 40, 48, 49, 53]. Training typically uses an encoder–decoder architecture with two complementary objectives: *indexing*, which associates document content with docids, and *retrieval*, which maps queries to docids. At inference time, GenIR employs constrained decoding (e.g., prefix tries) to restrict generation to valid identifiers.

Challenge: continual retrieval over dynamic corpora. Real-world collections are dynamic: new documents arrive continuously, and retrieval systems must integrate them without compromising performance on previously indexed data [30]. This is particularly challenging for GenIR. Unlike modular IR pipelines with explicit index structures, GenIR entangles query–docid mappings in model parameters, so updating the model to accommodate new documents can disrupt established retrieval behavior. Joint retraining (or rehearsal) on old and new data is the most reliable way to preserve performance, but it is often prohibitively expensive at scale. In contrast, adapting only on newly arrived documents is efficient, but it typically causes catastrophic forgetting [7, 29], leading to a stability–plasticity trade-off: gains on the new slice (*plasticity*) often come at the cost of degraded retrieval on earlier content (*stability*).

Existing approaches. Prior work on continual GenIR mitigates forgetting *during adaptation*, through replay or synthetic rehearsal, identifier design and index-side strategies, or parameter-efficient updates such as prompt tuning and adapters. While effective, these methods typically couple plasticity and retention in a single objective, making performance under long sequences of continual updates sensitive to gradient trade-offs and sampling ratios (see Section 2).

Empirical motivation. We analyze continual adaptation in GenIR by training on an initial corpus and sequentially incorporating disjoint document batches. We consider two representative docid designs: (i) semantic numeric identifiers such as product-quantized codes [5, 44, 51–53], and (ii) keyword-based identifiers such as URLs or titles [3, 21, 31, 34, 39]. We find that parametric adaptation methods, including full fine-tuning and parameter-efficient updates such as LoRA [10], reliably improve retrieval on newly added documents but substantially degrade performance on earlier slices. In contrast, when model parameters are frozen, expanding the constrained-decoding candidate set alone causes only modest retention drift,

while zero-shot transfer to future-slice identifiers remains limited. These controls indicate that the dominant source of degradation is update-induced interference in the learned query–docid mapping rather than search-space growth alone. They also indicate that adapting to each new slice remains necessary for plasticity, even though it harms stability. This motivates a post-adaptation mechanism that improves retention after learning new slices, without re-optimizing the full backbone.

Our approach. We propose *post-adaptation memory tuning* (PAMT), a two-stage framework that decouples learning new docid mappings from post-adaptation decoder-side calibration. In Stage 1, the GenIR backbone is adapted to the newly arrived slice using a standard adaptation method. In Stage 2, the adapted backbone is frozen and augmented with a *parametric memory head* (PMH), implemented as a product-key memory [20]. During prefix-trie constrained decoding, PMH produces sparse hidden-space corrections whose effect on next-token scores is computed only over trie-valid tokens. To limit cross-slice interference, Stage 2 updates only a fixed budget of memory values selected using decoding-time access statistics, while keeping the backbone and PMH routing components fixed during the stabilization stage. This yields a value-only post-adaptation calibration step targeted at the docid-decoding interface, rather than further backbone optimization. Importantly, PAMT does not attempt to restore pre-adaptation routing; instead, it performs calibration under the routing pattern induced by Stage 1.

Contributions. (i) We provide an empirical characterization of catastrophic forgetting in continual GenIR, quantifying the stability—plasticity trade-off across semantic numeric and keyword-based docid schemes under full and parameter-efficient adaptation. (ii) We introduce *post-adaptation memory tuning* (PAMT), an adapt-then-stabilize framework that improves retention after session-level adaptation through a memory-only stabilization stage, without replaying legacy supervision. (iii) We introduce a decoder-side *parametric memory head* (PMH) for continual GenIR that enables sparse post-adaptation value-only calibration under prefix-trie constrained decoding, allowing retention to be improved without further backbone updates or routing changes during Stage 2.

2 Related Work

Continual learning. Catastrophic forgetting [7, 29] remains a central challenge when training neural models on non-stationary data. Common mitigation strategies include experience replay [4, 36, 37], parameter regularization such as EWC [17], and knowledge distillation [22]. In GenIR, these strategies face additional constraints: replay requires storing queries and/or documents, raising privacy concerns [25] and scaling poorly over many slices [6, 13, 38]; regularization becomes harder to tune as the docid space expands; and distillation introduces additional objectives and hyperparameters that may be sensitive to slice-level distribution shift.

Continual learning in GenIR. Prior work adapts GenIR models to evolving corpora primarily through training-time updates. Replay-based approaches such as DSI++ [30] combine Sharpness-Aware Minimization with generative replay to mitigate forgetting. Other methods modify model capacity or interfaces: IncDSI [18] incrementally expands the output layer; PromptDSI [12] attaches learnable prompts; and MixLoRA-DSI [11] introduces a dynamically expandable mixture of LoRA experts driven by distribution shifts. CLEVER [5] combines incremental product quantization with memory-augmented learning and pseudo-query generation, while CorpusBrain++ [9] uses task-adaptive adapters with replay for continual pretraining. Complementary to backbone-adaptation methods, MDGR [50] emphasizes docid design to support search-space expansion under a fixed codebook, improving robustness to corpus growth without additional backbone training but potentially limiting plasticity when new query–docid mappings must be learned. In contrast to these training-time approaches, PAMT applies a separate post-adaptation stabilization stage through memory-only updates, without further modifying the backbone.

Memory-augmented language models and sparse memory updates. Memory-layer language models [1] augment transformers with large key–value memories accessed sparsely, including product-key top-k addressing, often as replacements or augmentations of feed-forward sublayers [16]. This complements analyses viewing transformer feed-forward layers as unnormalized key–value memories [8]. Sparse Memory Finetuning [24] reduces forgetting under distribution shift by ranking memory slots with access-based TF–IDF statistics and updating only the selected value slots while freezing the rest of the model. Our setting differs in both task and integration point. We target continual *docid decoding* in GenIR under prefix-trie constraints, and attach a modular parametric memory head queried during constrained decoding rather than inserting memory layers inside the backbone. PAMT uses sparse memory updates as a post-adaptation calibration mechanism: the top-p historically accessed rows are treated as protected legacy rows and excluded from updates, while a fixed budget of the remaining value rows is selected per session using current access and inverse historical access statistics. Unlike memory-layer finetuning inside transformer backbones, PAMT applies value-only calibration *after* backbone adaptation, with hidden-space corrections projected through the frozen output embedding only onto trie-valid next-token scores.

3 Preliminaries

3.1 Generative information retrieval

Let $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\}$ be a document corpus and $\mathcal{Q} = \{q_1, \ldots, q_{|\mathcal{Q}|}\}$ a set of queries. Each document $d \in \mathcal{D}$ is assigned an identifier sequence $I_d = (t_1, \ldots, t_{|I_d|})$ whose tokens lie in a vocabulary $\mathcal{V}$. For semantic schemes (e.g., PQ codes), $\mathcal{V}$ is augmented with dedicated docid tokens; for keyword-based schemes (e.g., title+URL), identifiers are formed from the base tokenizer vocabulary. We denote the identifier set by $\mathcal{I}(\mathcal{D}) = \{I_d : d \in \mathcal{D}\}$.

GenIR uses an encoder–decoder Transformer with parameters θ to model the conditional distribution over docid sequences:

$$P(I \mid q; \theta) = \prod_{k=1}^{|I|} P(t_k \mid t_{<k}, q; \theta). \tag{1}$$

Training minimizes cross-entropy over supervision pairs $\mathcal{S}$:

$$\mathcal{L}(\theta) = -\sum_{(q,d)\in\mathcal{S}} \log P(I_d \mid q; \theta). \tag{2}$$

At inference, GenIR retrieves by *constrained docid decoding*: a prefix trie $\mathcal{T}(\mathcal{D})$ is constructed over $\mathcal{I}(\mathcal{D})$, and beam search is restricted to trie-valid tokens $A_k(\pi) \subseteq \mathcal{V}$ that extend the current prefix $\pi = t_{<k}$. This guarantees that every generated sequence corresponds to a valid document in $\mathcal{D}$. Documents are ranked by sequence log-probability (approximated via beam search), i.e., $\mathrm{Rel}(q, d) \triangleq \log P(I_d \mid q; \theta)$.

3.2 Continual adaptation task

We consider a corpus that evolves through a sequence of discrete sessions, with disjoint slices $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$ for $i \neq j$.

- *Session 0 ($t = 0$):* The base model θ_0 is optimized on the initial corpus $\mathcal{D}_0$ and query set $\mathcal{Q}_0$. A prefix trie $\mathcal{T}_0$ is constructed.
- *Subsequent sessions ($t \geq 1$):* A new document set $\mathcal{D}_t$ and corresponding query set $\mathcal{Q}_t$ are introduced. The cumulative corpus is $\mathcal{D}_{0:t} = \bigcup_{i=0}^{t} \mathcal{D}_i$, and the trie is expanded to $\mathcal{T}_{0:t} = \mathcal{T}(\mathcal{D}_{0:t})$.

At session t, the model must answer queries over $\mathcal{D}_{0:t}$. The challenge is to adapt to the new slice $\mathcal{D}_t$ (*plasticity*) while maintaining retrieval quality on earlier slices $\mathcal{D}_{0:t-1}$ (*stability*/retention) under an expanding trie and a shared identifier scheme. We distinguish two search-space protocols (Expanded vs. Fixed) later in § 5.3.

3.3 Document identifier schemes

We consider two representative types of docids.

Semantic product-quantized (SPQ) docids. Each document is embedded and quantized using product quantization [14]. The embedding is partitioned into M sub-vectors, each quantized via a codebook of K centroids learned on $\mathcal{D}_0$. The identifier I_d is a length-M sequence of centroid tokens, each corresponding to a dedicated docid token in $\mathcal{V}$. Codebooks are frozen after initial training; new documents in $\mathcal{D}_t$ are quantized using these existing centroids, inducing a shared semantic token set across slices.

Keyword-based (title+URL, TU) docids. We construct variable-length docids from document metadata by concatenating the title with normalized URL components and tokenizing with the base vocabulary to obtain I_d. This yields interpretable docids that naturally share prefixes between related documents.

4 Methodology

We propose *post-adaptation memory tuning* (PAMT) for continual GenIR. As shown in Figure 1, PAMT follows an *adapt–then–stabilize* procedure that separates learning new query–docid mappings from post-adaptation decoder-side calibration. For each session $t \geq 1$, Stage 1 adapts the GenIR backbone on the arriving slice $\mathcal{D}_t$ to ensure *plasticity*. Stage 2 then freezes the adapted backbone and applies a *memory-only calibration* step via the parametric memory head (PMH): it updates only a fixed-budget subset of PMH value rows using current-session supervision, while excluding historically high-usage rows to preserve legacy retrieval behavior. Stage 2 does not attempt to restore pre-adaptation routing; instead, it calibrates under the routing pattern induced by Stage 1, without rehearsal on legacy slices.

4.1 Parametric memory head (PMH)

PMH is an external decoding-time component, co-trained with the backbone on $\mathcal{D}_0$ and later used for post-adaptation calibration of docid generation. At each decoding step, it reads the current decoder hidden state, retrieves a small set of relevant memory rows, and converts them into an additive hidden-space correction that adjusts next-token scores under prefix-trie constraints. PMH does not alter the trie constraint; it only reweights trie-valid next-token scores.

Architecture. PMH is implemented as a parametric product-key memory (PKM) [20]. Given a decoder hidden state, a learned query projection maps it into H product-key query heads. These heads share a pair of product-key tables and a trainable value table:

- *Dual sub-key tables:* PMH stores two key tables $K^{(1)} = \{k_i^{(1)}\}_{i=1}^{S}$ and $K^{(2)} = \{k_j^{(2)}\}_{j=1}^{S}$, where $k_i^{(1)}, k_j^{(2)} \in \mathbb{R}^{d_{\text{key}}/2}$. For each query head, the query is split into two halves and scored against the two shared key tables. A memory row is addressed by a compositional pair (i, j), yielding a shared value address space of size S^2 while requiring only two S-way lookups per head. This product-key factorization provides large memory capacity with efficient retrieval.
- *Latent value table:* PMH has a trainable value matrix $V^{\text{hid}} \in \mathbb{R}^{S^2 \times d}$, whose rows store latent correction vectors in decoder hidden space. Each row corresponds to one flattened product-key pair (i, j). We initialize V^{hid} before initial co-training on $\mathcal{D}_0$; after $\mathcal{D}_0$ training, subsequent Stage 2 updates modify only selected rows of this value table. Retrieved rows are combined into a hidden-space residual whose contribution to next-token logits is computed through the output embedding matrix frozen during Stage 2.

During docid generation, the decoder produces a hidden state $h_k \in \mathbb{R}^d$ at decoding step k. Let $E \in \mathbb{R}^{|\mathcal{V}| \times d}$ denote the output embedding matrix of the post-adaptation backbone, which is frozen during Stage 2 and used to score next tokens.

Step 1: Product-key addressing. At each decoding step, PMH uses h_k to retrieve a small number of relevant value rows. A query module f_ϕ maps h_k to H query vectors $\{z_{k,h}\}_{h=1}^{H}$, with $z_{k,h} \in \mathbb{R}^{d_{\text{key}}}$. Each query is split into two halves, $z_{k,h}^{(1)}, z_{k,h}^{(2)} \in \mathbb{R}^{d_{\text{key}}/2}$, and matched against the product-key tables using dot products. A key pair (i, j) is scored as

$$u_{k,h,i,j} = \langle z_{k,h}^{(1)}, k_i^{(1)} \rangle + \langle z_{k,h}^{(2)}, k_j^{(2)} \rangle.$$

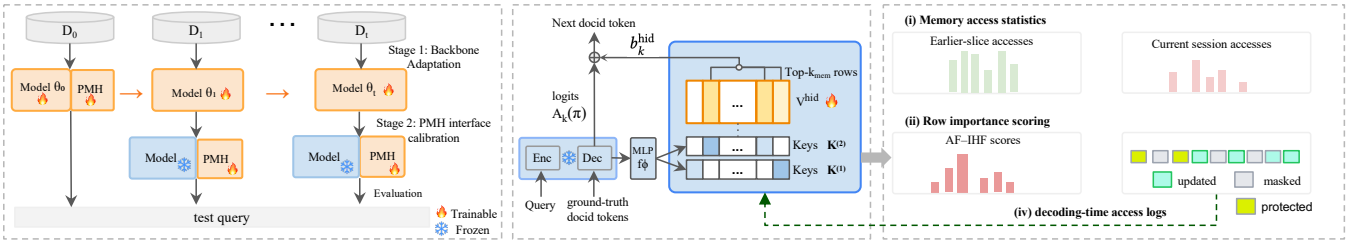
For each head h, we retrieve the top-K_{mem} rows, where K_{mem} denotes the number of retrieved value rows per head:

$$\mathcal{S}_{k,h} = \mathrm{TopK}\big(\{u_{k,h,i,j}\}_{i,j},\, K_{\text{mem}}\big), \qquad \mathcal{S}_k = \bigcup_{h=1}^{H} \mathcal{S}_{k,h}.$$

We flatten each key pair (i, j) into a unique row index $n \in \{1, \ldots, S^2\}$, for example $n = (i-1)S + j$, and compute TopK efficiently using factorized product-key lookup [20]. Equivalently, $u_{k,h,n}$ denotes the score for flattened row index n under head h.

Step 2: Latent correction construction. For each retrieved memory row $\ell \in \mathcal{S}_{k,h}$, PMH reads a latent correction vector $V^{\text{hid}}[\ell] \in \mathbb{R}^d$. We convert row scores into normalized weights using a softmax over the selected rows for each head:

$$\alpha_{k,h,\ell} = \frac{\exp(u_{k,h,\ell})}{\sum_{\ell' \in \mathcal{S}_{k,h}} \exp(u_{k,h,\ell'})}.$$



(a) Two-stage optimization pipeline (b) Parametric Memory Head (PMH) (c) Sparse PMH updates

Figure 1: Post-adaptation memory tuning (PAMT) for continual GenIR. (a) Adapt-then-stabilize pipeline: The parametric memory head (PMH) is attached and co-trained with the backbone on $\mathcal{D}_0$ to establish the initial addressing mechanism. At each session $t \geq 1$, the backbone is adapted on the new slice $\mathcal{D}_t$ (Stage 1), then frozen and refined via a memory-only stabilization stage (Stage 2). Stage 2 updates a sparse subset of PMH value rows to improve retention on legacy slices $\mathcal{D}_{0:t-1}$ *without gradient-based training on legacy relevance supervision.* (b) Parametric memory head: During prefix-trie constrained docid decoding, decoder hidden states are projected by f_ϕ to perform product-key top-K_{mem} retrieval from the key tables $(K^{(1)}, K^{(2)})$ and latent value table V^{hid}. Retrieved values are aggregated into a latent correction b_k^{hid} in decoder hidden space; its effect on token scores is computed via the frozen output embedding matrix over trie-valid tokens $A_k(\pi)$ only. (c) Sparse PMH updates: Decoding-time access logs from the current session $\mathcal{X}_t$ are contrasted with a historical access sketch to score rows using access frequency and inverse historical frequency (AF×IHF), forming an update set Ω_t. Sparsity is enforced by updating only $V^{\text{hid}}[n]$ for $n \in \Omega_t$ while blocking gradients on protected rows $\mathcal{P}_t$; the adapted backbone, f_ϕ, and keys remain fixed during Stage 2, keeping the Stage 2 routing function stable.

PMH then forms a correction vector by taking a weighted average of the retrieved values and summing across heads:

$$b_k^{\text{hid}} = \sum_{h=1}^{H} \sum_{\ell \in \mathcal{S}_{k,h}} \alpha_{k,h,\ell}\, V^{\text{hid}}[\ell], \qquad b_k^{\text{hid}} \in \mathbb{R}^d.$$

Intuitively, b_k^{hid} is a small additive adjustment in decoder hidden space that re-calibrates next-token scores under trie-constrained docid decoding.

Step 3: Bias-guided constrained decoding. At decoding step k, PMH adjusts next-token logits only over trie-valid options. Let $\ell_k[\tau]$ denote the backbone logit for token τ at step k, and let $A_k(\pi) \subseteq \mathcal{V}$ be the set of trie-valid next tokens given the current prefix $\pi = t_{<k}$. For each valid token $\tau \in A_k(\pi)$, we compute

$$\ell'_k[\tau] = \ell_k[\tau] + \langle b_k^{\text{hid}},\, E[\tau] \rangle, \quad \forall\, \tau \in A_k(\pi), \tag{3}$$

where $E[\tau]$ is the frozen output embedding row for token τ during Stage 2. Tokens outside $A_k(\pi)$ remain masked by the trie constraint, so PMH cannot make invalid docid continuations available. Restricting the projection to $A_k(\pi)$ keeps the additional computation proportional to the trie branching factor rather than the full vocabulary size.

4.2 Post-adaptation memory tuning (PAMT)

PAMT adds a *memory-only interface calibration* stage after session-level backbone adaptation. The goal is to incorporate new document mappings while preserving index stability, *without* replaying legacy relevance supervision and without further modifying the backbone.

Frozen backbone and addressing. In Stage 2, we freeze the adapted backbone, including any session-t adapters, the PMH query module f_ϕ, the product-key tables $(K^{(1)}, K^{(2)})$, and the output embedding matrix E. The only trainable component is the PMH value table V^{hid}, from which we update a fixed budget of m rows per session, corresponding to $m \cdot d$ trainable parameters. Because the correction is applied in decoder hidden space and projected to logits through the frozen output embedding, Stage 2 recalibrates the post-adaptation docid-scoring interface without further backbone updates or supervised training on legacy slices. Freezing the addressing function ensures that Stage 2 introduces no additional routing drift beyond that already induced by Stage 1 adaptation; the stage updates only the decoder-to-docid interface.

Access logs for sparse slot selection. Let $\mathcal{X}_t$ denote the query stream, i.e., the training queries observed in session t. To choose which memory rows to update *without* storing legacy supervision, we maintain a lightweight access log that tracks how often each memory row was retrieved in previous sessions. Concretely, we maintain a historical access-frequency counter $\text{AF}_t^{\text{hist}}(n)$ for every memory row $n \in \{1, \ldots, S^2\}$. We initialize $\text{AF}_1^{\text{hist}}$ using PMH accesses collected on $\mathcal{X}_0$ after initial training on $\mathcal{D}_0$. After finishing session $t-1$, we update this counter using a *binary* indicator per decoded docid sequence:

$$\text{AF}_t^{\text{hist}}(n) \leftarrow \text{AF}_{t-1}^{\text{hist}}(n) + \sum_{x \in \mathcal{X}_{t-1}} \mathbf{1}[\exists k :\ n \in \mathcal{S}_k(x)], \tag{4}$$

where $\mathcal{S}_k(x)$ is the set of PMH rows retrieved at step k when generating the docid for query x. This counts how many sequences used a row at least once, rather than overweighting repeated hits within a single sequence.

At the current session t, we compute an analogous access count on the new slice. Current-session accesses are collected after Stage 1 adaptation, so row selection reflects the routing pattern of the adapted model used during Stage 2:

$$\text{AF}_t(n) = \sum_{x \in \mathcal{X}_t} \mathbf{1}[\exists k :\ n \in \mathcal{S}_k(x)]. \tag{5}$$

To make row importance comparable within the session, we normalize access frequencies as:

$$\widehat{\mathrm{AF}}_t(n) \;=\; \frac{\mathrm{AF}_t(n)}{\sum_{n'=1}^{S^2} \mathrm{AF}_t(n')}. \tag{6}$$

We normalize current-session counts to ensure comparable row importance within the session; historical counts remain unnormalized because they serve only to downweight frequently used rows in the inverse historical frequency term. These logs store only scalar access counts, not query text or document identifiers, and do not receive gradients.

Protected set $\mathcal{P}_t$ (do-not-change legacy rows). Some memory rows are heavily used by earlier slices, so changing them is likely to hurt retention. We therefore define a *protected set* $\mathcal{P}_t$ whose rows are frozen during Stage 2; their gradients are always zero. We select $\mathcal{P}_t$ as the top-p fraction of rows by historical usage:

$$\mathcal{P}_t \;=\; \mathrm{TopP}\Big(\{\mathrm{AF}_t^{\mathrm{hist}}(n)\}_{n=1}^{S^2},\, p\Big), \tag{7}$$

where $\mathrm{TopP}(\cdot, p)$ returns the indices of the top p fraction of rows by score.

Budgeted update set Ω_t (update only m rows). Stage 2 updates only a fixed budget of m value rows. We choose these rows from the remaining capacity

$$C_t = \{1, \ldots, S^2\} \setminus \mathcal{P}_t,$$

favoring rows that are useful for the current slice but not heavily used in the past. Following the access-based selection principle of Lin et al. [24], we rank candidate rows using an AF×IHF score, where AF denotes current-session access frequency and IHF denotes inverse historical frequency:

$$w_t(n) \;=\; \widehat{\mathrm{AF}}_t(n) \cdot \log \frac{Z+1}{\mathrm{AF}_t^{\mathrm{hist}}(n)+1}, \qquad n \in C_t, \tag{8}$$

where $Z = |\mathcal{X}_{<t}|$ is the total number of queries accumulated in the historical log.[1] The second term acts as an *inverse historical frequency* (IHF),

$$\mathrm{IHF}_t(n) = \log \frac{Z+1}{\mathrm{AF}_t^{\mathrm{hist}}(n)+1}, \tag{9}$$

which downweights rows that were frequently accessed in prior sessions. We then update only the top-m rows:

$$\Omega_t \;=\; \mathrm{TopK}\Big(\{w_t(n)\}_{n \in C_t},\, m\Big). \tag{10}$$

Stage 2 optimization and gradient masking. Stage 2 trains *only* on current-session supervision from $\mathcal{D}_t$, equivalently queries $\mathcal{X}_t$, using the biased logits from Eq. (3). During Stage 2 training, logits are computed under teacher-forced gold prefixes. For each decoding step k, let y_k be the gold next token, and let $\mathcal{N}_k$ be negatives sampled from the trie-valid set $A_k(\pi) \setminus \{y_k\}$. We optimize the hinge ranking objective:

$$\mathcal{L}_{\mathrm{rank}} \;=\; \sum_k \sum_{\tau^- \in \mathcal{N}_k} \max\Big(0,\; \gamma - \ell'_k[y_k] + \ell'_k[\tau^-]\Big), \tag{11}$$

where $\gamma > 0$ is a margin hyperparameter.

To enforce sparse updates, we mask gradients so that only the selected rows in Ω_t can change:

$$\nabla V^{\mathrm{hid}}[n] \;\leftarrow\; \mathbf{1}[n \in \Omega_t] \cdot \nabla V^{\mathrm{hid}}[n], \qquad \forall n \in \{1, \ldots, S^2\}. \tag{12}$$

[1] Equivalently, $Z = \sum_{s<t} |\mathcal{X}_s|$.

Rows outside Ω_t receive zero gradients. Thus, Stage 2 updates only a small, access-selected subset of memory values while keeping the backbone, addressing mechanism, and output embedding fixed.

By freezing the backbone and protecting high-usage legacy entries, PAMT concentrates session-specific corrections in less historically used memory rows, aiming to stabilize docid decoding at session t while limiting interference with earlier slices, without replaying legacy relevance supervision.

5 Experimental Setup

We describe benchmarks, continual splits, model configurations, adaptation methods, and evaluation. We vary four factors, docid scheme (SPQ, TU), adaptation method (Full FT, LoRA), search-space protocol (Expanded, Fixed), and benchmark (MS MARCO, NQ), to analyze the plasticity–stability trade-off in Section 6.

5.1 Datasets and continual splits

Benchmarks. We evaluate on *MS MARCO Document Ranking* [2] and *Natural Questions (NQ)* [19]. Following prior GenIR work [5, 39, 41–43, 50, 53], we use the commonly adopted 320K-document MS MARCO subset and NQ320K, with preprocessing from Wang et al. [47].

Continual benchmark construction. Each corpus is split into disjoint document slices $\mathcal{D}_0, \ldots, \mathcal{D}_n$ with $n = 5$: $\mathcal{D}_0$ contains 50% of documents, and $\mathcal{D}_1$–$\mathcal{D}_5$ each contain 10%. At session t, the cumulative corpus is $\mathcal{D}_{0:t} = \bigcup_{i=0}^{t} \mathcal{D}_i$. Test queries are partitioned into $\mathcal{Q}_{\mathrm{test},0}, \ldots, \mathcal{Q}_{\mathrm{test},n}$ such that each $q \in \mathcal{Q}_{\mathrm{test},t}$ has all relevant documents in $\mathcal{D}_t$ and none in $\mathcal{D}_{0:t-1}$; queries spanning multiple slices are discarded. At each session t, training and validation supervision is constructed only from the arriving slice $\mathcal{D}_t$. We denote the session-t training queries by $\mathcal{X}_t$.

5.2 Model architecture and document identifiers

GenIR backbone. We use T5-BASE.[2] For SPQ docids, the decoder vocabulary is expanded from 32,128 to 40,320 tokens by adding 8,192 dedicated docid tokens organized into $M = 32$ disjoint blocks.

Document identifier schemes. We use two docid schemes. *SPQ IDs* embed each document using SentenceTransformer E5-Mistral-7B-Instruct[3] over the title plus the first 50 sentences, apply ℓ_2 normalization, and PQ-code it with $M = 32$ and $K = 256$; the codebook is learned on $\mathcal{D}_0$ and reused for all sessions. *TU IDs* concatenate title tokens (up to 20) with reversed URL path segments plus the second-level domain, tokenize with the T5 tokenizer, and truncate to at most 100 subword tokens, with PAD/EOS removed.

Initial training on $\mathcal{D}_0$. Models are initialized from `t5-base` and trained on $\mathcal{D}_0$ using document-to-docid pairs, pseudo-query-to-docid pairs generated via `doc2query`[4] with 10 pseudo-queries per document, and real query-to-docid pairs. We optimize with AdamW (lr 1×10^{-3}) for 40 epochs.

Constrained decoding. Inference uses constrained beam search with beam size 10 over a prefix trie of valid docids. In *Expanded*, decoding at session t uses $\mathcal{T}(\mathcal{D}_{0:t})$ by inserting $\mathcal{I}(\mathcal{D}_t)$ into $\mathcal{T}(\mathcal{D}_{0:t-1})$.

[2] https://huggingface.co/google-t5/t5-base
[3] https://huggingface.co/intfloat/e5-mistral-7b-instruct
[4] https://huggingface.co/castorini/doc2query-t5-base-msmarco

In *Fixed*, decoding uses the static full-corpus trie $\mathcal{T}(\mathcal{D}_{0:n})$ at every session. Decoding is capped at $M = 32$ tokens for SPQ and 100 tokens for TU.

5.3 Evaluation protocol

Models are first trained on $\mathcal{D}_0$ and then sequentially adapted to $\mathcal{D}_1, \ldots, \mathcal{D}_n$.

Lower-triangular evaluation. After session $t \in \{0, \ldots, n\}$, we evaluate on all seen test query sets $\mathcal{Q}_{\text{test},s}$ for $s \le t$. Let $R^{\text{MRR}}_{t,s}$ denote MRR@10 on $\mathcal{Q}_{\text{test},s}$ using the model state after session t after PAMT when applicable. We also report $R^{\text{Hit}}_{t,s}$ as Hit@10 (%), but all aggregate continual-learning metrics are computed from $R^{\text{MRR}}_{t,s}$ unless stated otherwise.

Search-space protocols. We use two constrained-decoding protocols to separate model-induced forgetting from search-space expansion: *Expanded* constrains decoding at session t to identifiers in $\mathcal{D}_{0:t}$, while *Fixed* always constrains decoding to identifiers in the full corpus $\mathcal{D}_{0:n}$. Fixed is feasible because all docids can be pre-assigned without additional learning: TU from metadata and SPQ by reusing the $\mathcal{D}_0$ codebook.

Aggregate continual-learning metrics. Let $R_{t,s} \equiv R^{\text{MRR}}_{t,s}$. We report: (i) *Average Performance (AP):* $\text{AP}_n = \frac{1}{n+1}\sum_{s=0}^{n} R_{n,s}$; (ii) *Signed Backward Transfer (BWT):* $\text{BWT}^{\pm}_n = \frac{1}{n}\sum_{s=0}^{n-1}(R_{n,s} - R_{s,s})$ [26]; (iii) *Diagonal Forward Performance (*FWT_{diag}*):* $\text{FWT}_{\text{diag},n} = \frac{1}{n}\sum_{s=1}^{n} R_{s,s}$ [9, 11], measuring average performance on each newly arrived slice immediately after adapting to that slice.

Stage 1 adaptation. We consider two session-level adaptation methods. For Full FT, θ_t denotes the backbone after adapting on $\mathcal{D}_t$, initialized from θ_{t-1}; all backbone parameters are updated using target-docid negative log-likelihood with AdamW (lr 5×10^{-4}) for 3 epochs and global batch size 64. For LoRA [10], the backbone remains θ_0 and the session-t adapter is denoted ψ_t, so the adapted model is (θ_0, ψ_t). LoRA is applied to all linear layers using `peft` [28] with rank $r = 64$, $\alpha = 128$, and dropout 0.05; adapters are initialized from ψ_{t-1} and updated with AdamW (lr 1×10^{-3}), 10% warmup, 3 epochs, and global batch size 64. During evaluation, LoRA decoding uses (θ_0, ψ_t) without merging. When applying PAMT after LoRA, both θ_0 and ψ_t are frozen during Stage 2.

Post-adaptation memory tuning (PAMT). PMH retrieves $K_{\text{mem}} = 32$ value rows per head to construct a latent correction $b^{\text{hid}}_k \in \mathbb{R}^d$ (Section 4.1); this fixed retrieval-width hyperparameter keeps PMH access sparse while allowing multiple value rows to contribute to each decoding-step correction. Its logit effect is computed via the frozen output embedding only over trie-valid tokens $A_k(\pi)$ (Eq. 3). The value table is padded to a perfect square, yielding $N_{\text{mem}} = S^2$ rows for product-key addressing. Stage 2 protects the top $p = 10\%$ historically accessed rows and updates a fixed budget of $m = 10{,}000$ value rows per session using the AF×IHF selection rule for Ω_t. Stage 2 then optimizes the hinge ranking objective in Eq. (11) with margin $\gamma = 0.01$, $k_{\text{neg}} = 8$ trie-valid negatives, and SGD (lr 1×10^{-3}) for 2 epochs with 10% linear warmup and decay. We analyze sensitivity to the protected capacity p and update budget m in Section 6.5.

Trainable parameter footprint. Stage 1 Full FT updates 260.16M parameters for SPQ docids and 247.58M for TU docids, while LoRA updates 25.95M adapter parameters for both schemes. Stage 2 PAMT freezes the adapted backbone and PMH addressing mechanism, updating only $m = 10{,}000$ selected PMH value rows per session. With T5-base hidden size $d = 768$, this gives $m \cdot d = 7.68$M trainable parameters. The full PMH value table contains 160,000 rows on MS MARCO and 55,696 rows on NQ, so the default update budget modifies 6.25% and 17.95% of the value table, respectively.

PMH initialization. PMH is attached during initial training on $\mathcal{D}_0$ and trained with the standard docid negative log-likelihood loss; the hinge ranking objective is used only during Stage 2 PAMT. After session 0, the addressing mechanism, f_ϕ and product keys $(K^{(1)}, K^{(2)})$, is frozen, and all subsequent Stage 2 updates are restricted to V^{hid}. Thus, PAMT performs value-only calibration without introducing additional parameter-induced routing changes during Stage 2.

Baselines. We compare against index-based retrieval and continual GenIR baselines: (i) *Lexical:* BM25 [35] using LlamaIndex `BM25Retriever`;[5] (ii) *Dense:* DPR [15] and DPR-HN [27, 33] via Pyserini [23], with the document encoder pretrained on $\mathcal{D}_0$ and frozen, and the query encoder updated per session; (iii) *Continual GenIR:* DSI++ [30], CLEVER [5], CorpusBrain++ [9], PromptDSI [12], and MixLoRA-DSI [11]. We adapt public implementations when available; otherwise, we implement the method following the corresponding paper. For DSI++, we build on a public DSI codebase[6] and add Sharpness-Aware Minimization and generative replay following Mehta et al. [30].

6 Experimental Results

We evaluate continual GenIR under evolving corpora through four research questions: **(RQ1)** How do continual GenIR models trade off *plasticity* and *stability*, and how do adaptation method and docid design affect this trade-off? **(RQ2)** To what extent does post-adaptation memory tuning (PAMT) improve *stability* while preserving *plasticity* on newly added documents? **(RQ3)** How much degradation is driven by *task-induced* effects (e.g., search-space growth and identifier transferability) versus *model-induced* forgetting from parameter updates? **(RQ4)** How does PAMT compare with prior continual GenIR methods under the same protocol in terms of the stability–plasticity trade-off?

6.1 Plasticity–stability trade-off in continual GenIR

The Stage 1 results in Figures 2 and 3 show a pronounced *plasticity–stability trade-off* under corpus evolution (RQ1). Across both MS MARCO (MS) and Natural Questions (NQ), sequential adaptation provides plasticity on newly added slices but yields strongly negative $\text{BWT}^{\pm}$ on earlier ones. Under the Expanded protocol, Full FT with SPQ docids reaches $\text{BWT}^{\pm} = -55.72$ on MS and -35.95 on NQ, indicating severe forgetting in continual GenIR. This pattern holds across both adaptation regimes and both docid schemes, suggesting that forgetting is a structural challenge rather than an artifact of a particular configuration.

Effect of adaptation method (Full FT vs. LoRA). Full FT generally provides stronger plasticity than LoRA, yielding higher diagonal Hit@10 on newly added slices and higher FWT_{diag} under

[5] https://developers.llamaindex.ai/python/examples/retrievers/bm25_retriever/
[6] Unofficial implementation: https://github.com/ArvinZhuang/DSI-transformers.

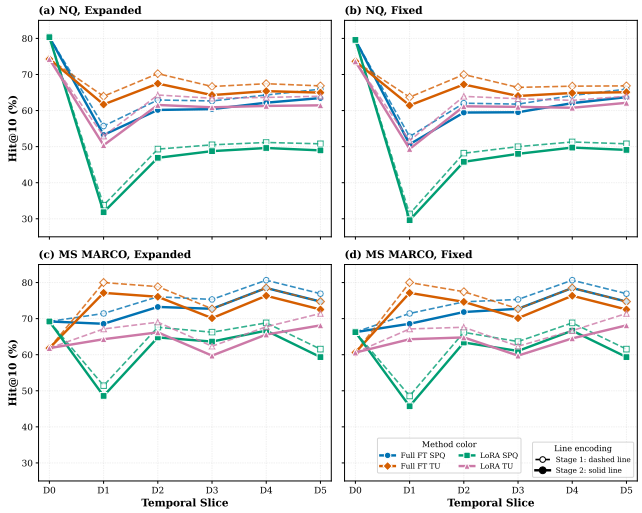


Figure 2: Stage 1 vs. Stage 2 across temporal slices. Hit@10 (%) over slices D_0–D_5 for NQ and MS MARCO under Expanded and Fixed protocols. Dashed lines denote Stage 1 adaptation before PAMT; solid lines denote Stage 2 after PAMT. The D_0 point corresponds to the initially trained model. Colors indicate method/docid combinations.

both search-space protocols. However, stronger plasticity does not consistently translate into better retention. On MS, LoRA reduces forgetting relative to Full FT under both docid schemes (Expanded $BWT^{\pm}$: SPQ −48.70 vs. −55.72; TU −31.55 vs. −38.43). On NQ, the pattern depends on the identifier scheme; for example, under Expanded with TU docids, Full FT is more stable than LoRA (−25.93 vs. −32.90). These results indicate that parameter count alone does not explain forgetting; rather, the disruption of learned query–docid mappings is central.

Effect of docid design (SPQ vs. TU). Docid design strongly affects the attainable stability–plasticity trade-off. Across both datasets and adaptation methods, keyword-based TU identifiers yield less negative $BWT^{\pm}$ than SPQ while maintaining competitive diagonal performance. Under Expanded with Full FT, switching from SPQ to TU improves $BWT^{\pm}$ from −55.72 to −38.43 on MS and from −35.95 to −25.93 on NQ, and substantially increases AP on NQ (15.38 → 31.59). This does not mean that TU is uniformly stronger: SPQ performs better on the initial slice $\mathcal{D}_0$, but TU yields a more favorable continual-learning profile after sequential updates. We return to this point in RQ3.

Expanded vs. Fixed search space. The same qualitative pattern holds under both search-space protocols. Although Fixed can be slightly harder in earlier sessions because decoding is performed over the full identifier set from the start, method ordering and forgetting severity remain similar. For example, on MS with Full FT–SPQ, $BWT^{\pm}$ is −55.72 under Expanded and −55.40 under Fixed. Thus, candidate-set growth alone is unlikely to explain the observed degradation. The central difficulty is preserving previously learned query–docid mappings during sequential parameter updates, rather than merely decoding over a larger identifier set.

Overall, Stage 1 adaptation provides plasticity on newly added slices but fails to preserve earlier ones. This motivates a second, post-adaptation stabilization step that targets retention without further modifying the adapted backbone.

6.2 Effectiveness of PAMT

The Stage 2 results in Figures 2 and 3 show that post-adaptation memory tuning (PAMT) substantially improves *stability* while preserving most of the *plasticity* gained in Stage 1 (RQ2). Across both datasets, Stage 2 consistently reduces the magnitude of signed $BWT^{\pm}$ relative to Stage 1. Under the Expanded protocol, Full FT–SPQ improves from −55.72 to −22.64 on MS MARCO and from −35.95 to −14.21 on NQ, while Full FT–TU improves from −38.43 to −15.37 on MS MARCO and from −25.93 to −9.16 on NQ. The same qualitative pattern holds under Fixed, indicating that the gains are not an artifact of search-space growth. This pattern is consistent across Full FT and LoRA, and across both SPQ and TU identifiers.

Retention gains with limited impact on new-slice performance. PAMT substantially increases AP while incurring only modest reductions in diagonal new-slice performance. Under Expanded, AP rises from 15.23 to 31.45 for Full FT–SPQ on MS MARCO and from 31.59 to 42.31 for Full FT–TU on NQ, with similarly strong gains across the remaining settings. The improvements are not confined to Full FT: under Expanded on MS MARCO, PAMT improves $BWT^{\pm}$ from −48.70 to −19.82 for LoRA–SPQ and from −31.55 to −11.23 for LoRA–TU. Thus, PAMT improves legacy retrieval while preserving most of the retrieval behavior acquired during Stage 1 adaptation. In other words, the main effect of Stage 2 is not to relearn the current slice, but to improve retention after Stage 1 while keeping the newly acquired behavior largely intact.

Interpreting the Stage 2 gains. Stage 2 freezes the adapted backbone and PMH routing components, and updates only a sparse subset of PMH value rows. The resulting hidden-space correction is projected through the frozen output embedding and applied only to trie-valid next tokens (Eq. 3). This structurally constrains what Stage 2 can change: it can re-rank among valid continuations at decoding time, but it does not alter the backbone representations, PMH routing, or the trie constraint itself. PAMT should therefore be understood as a value-only post-adaptation calibration step on the adapted decoding interface, rather than as a mechanism for restoring pre-adaptation routing. This interpretation is consistent with the observed pattern: earlier-slice retention improves markedly, while new-slice diagonal performance is largely maintained.

6.3 Disentangling sources of degradation

RQ3 asks how much continual GenIR degradation is attributable to (i) *search-space growth* from increased candidate competition, (ii) *identifier transferability* to previously unseen documents, and (iii) *parameter-induced forgetting* during sequential adaptation. To separate these factors, we fix the session-0 checkpoint θ_0 trained on $\mathcal{D}_0$ and evaluate two parameter-free controls: one that expands only the constrained-decoding trie, and one that tests zero-shot transfer to future slices under slice-only decoding. Table 1 also reports docid collision rates, which help explain differences between SPQ and TU. Together, these controls separate degradation due to a changing

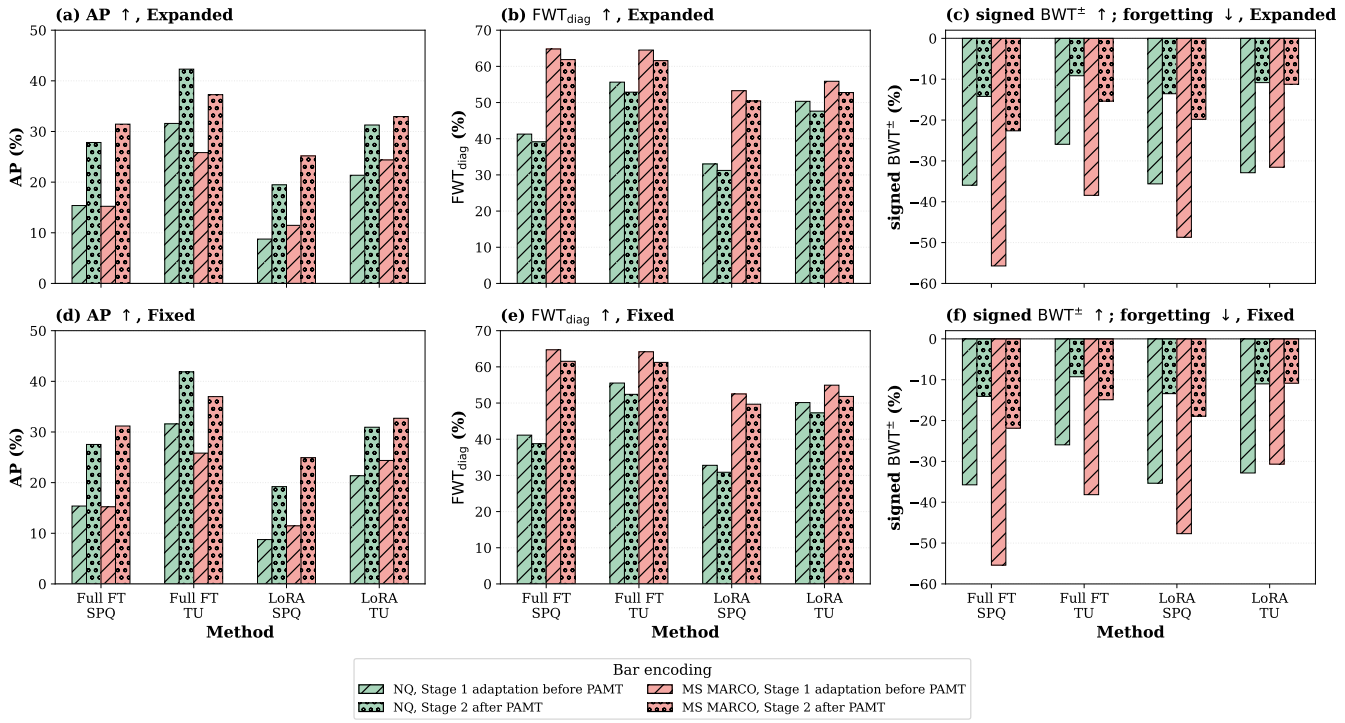


Figure 3: Stage 1 vs. Stage 2 aggregate continual-learning metrics. AP, $\mathrm{FWT}_{\mathrm{diag}}$, and signed $\mathrm{BWT}^{\pm}$ are computed from MRR@10 and reported in percentage points for NQ and MS MARCO under Expanded and Fixed protocols. Hatched bars denote Stage 1 adaptation before PAMT; dotted bars denote Stage 2 after PAMT.

Table 1: RQ3 controls using the frozen θ_0 checkpoint trained on $\mathcal{D}_0$. *D0 Hit@10* is performance on $Q_{\mathrm{test},0}$ with the $\mathcal{D}_0$ trie. *Frozen drop* is the absolute Hit@10 decrease on $Q_{\mathrm{test},0}$ when decoding expands from $\mathcal{I}(\mathcal{D}_0)$ to $\mathcal{I}(\mathcal{D}_{0:5})$ with no parameter updates. *ZS Avg* is average strict identifier-transfer Hit@10 on $Q_{\mathrm{test},t}$ for $t = 1, \ldots, 5$ under slice-only decoding over $\mathcal{I}(\mathcal{D}_t)$. *Coll. rate* is the docs-affected collision rate, i.e., the fraction of documents whose identifier is shared by at least one other document. Candidate counts denote unique identifier sequences.

Dataset	docid	D0 Hit@10	Frozen drop	Candidates	ZS Avg	Coll. rate
MS MARCO	SPQ	69.21	2.95	152K → 301K	5.64	7.45%
	TU	61.82	1.23	158K → 315K	14.46	2.37%
NQ	SPQ	80.34	0.76	54K → 108K	8.97	3.65%
	TU	74.28	0.66	55K → 110K	20.25	1.84%

search space, limitations of the identifier scheme, and sequential parameter updates.

Search-space growth alone causes only minor retention drift. The first control freezes θ_0 and expands only the decoding prefix trie by inserting docids from arriving slices. Retrieval is therefore performed over the cumulative identifier set $\mathcal{I}(\mathcal{D}_{0:t})$ without any parameter updates. As shown in Table 1, retention on $Q_{\mathrm{test},0}$ changes only modestly even as the searchable identifier set roughly doubles. On MS MARCO, Hit@10 drops by 2.95 pp for SPQ and 1.23 pp for TU; on NQ, the drift remains below 1 pp for both docid schemes. Thus, candidate competition alone explains only a small fraction of the retention loss observed under continual adaptation in RQ1.

Zero-shot transfer to future slices is limited and identifier-dependent. The second control evaluates the same frozen checkpoint on future-slice queries $Q_{\mathrm{test},t}$ under slice-only decoding over $\mathcal{I}(\mathcal{D}_t)$. This isolates whether a model trained only on $\mathcal{D}_0$ can decode useful identifiers for unseen documents *without* adaptation. Zero-shot transfer is limited across both datasets: average strict identifier-transfer Hit@10 over $\mathcal{D}_1$–$\mathcal{D}_5$ is only 5.64 for SPQ versus 14.46 for TU on MS MARCO, and 8.97 versus 20.25 on NQ. These results show that future-slice identifiers are not reliably decodable from θ_0 alone, so learning later slices generally requires parameter updates rather than candidate-set expansion alone. They also show that identifier design already constrains continual behavior before any sequential updating occurs.

Sequential parameter updates dominate the degradation. Compared with the two frozen controls above, continual adaptation produces substantially larger retention loss. For example, under Expanded in RQ1, Full FT–SPQ reaches $\mathrm{BWT}^{\pm} = -55.72$ on MS MARCO and −35.95 on NQ, whereas the frozen trie-expansion control produces only small Hit@10 drift. Although these quantities

Table 2: RQ4 (Expanded): Comparison to prior approaches. Diagonal entries report per-slice Hit@10 (%) after each session under the Expanded protocol, using constrained decoding over the cumulative corpus $\mathcal{D}_{0:t}$. AP, $\mathrm{FWT}_{\mathrm{diag}}$, and signed $\mathrm{BWT}^{\pm}$ are computed from the lower-triangular MRR@10 matrix (Section 5.3). Best values in each column are underlined.

	MS MARCO									Natural Questions								
Method	D0	D1	D2	D3	D4	D5	AP↑	FWT↑	BWT↑	D0	D1	D2	D3	D4	D5	AP↑	FWT↑	BWT↑
BM25	67.73	61.43	70.42	63.64	56.99	54.95	30.39	33.56	-5.06	73.78	71.57	73.52	68.55	69.06	67.89	44.87	47.58	-4.49
DPR	72.95	66.92	74.62	67.10	62.58	59.60	34.50	37.60	-4.80	77.21	72.79	75.01	69.78	69.88	68.50	38.00	40.75	-4.20
DPR-HN	74.35	68.42	75.82	68.50	63.98	61.10	35.50	38.60	-4.60	78.41	73.99	76.21	70.98	71.08	69.70	39.50	42.15	-4.00
Non-PEFT continual GenIR baselines																		
DSI++	64.53	45.71	57.75	54.55	56.99	51.65	14.82	35.64	-32.45	72.15	28.57	41.38	43.28	44.62	43.51	11.23	27.86	-28.74
CLEVER	66.01	54.29	63.38	61.04	63.44	58.24	19.56	44.82	-26.38	75.42	35.84	48.97	49.75	50.90	49.87	14.67	32.54	-24.16
PEFT-based continual GenIR baselines																		
CorpusBrain++	63.79	48.57	59.15	57.14	59.14	54.95	17.24	38.96	-29.82	70.58	31.17	44.14	45.27	46.15	45.33	12.84	29.43	-26.58
PromptDSI	61.33	42.86	54.93	51.95	53.76	48.35	13.47	32.18	-34.67	68.92	27.27	39.66	41.04	42.31	41.56	10.56	25.72	-30.21
MixLoRA-DSI	65.27	61.43	67.61	64.94	67.74	64.84	22.83	51.36	-22.94	77.89	48.57	58.62	57.96	59.23	58.31	24.56	41.87	-18.43
Stage 1: Base adaptation (ours)																		
LoRA–SPQ	69.21	51.43	67.61	66.23	68.82	61.54	11.47	53.27	-48.70	80.34	33.76	49.31	50.50	51.15	50.78	8.76	33.02	-35.63
LoRA–TU	61.82	67.14	69.01	62.34	67.74	71.43	24.38	55.88	-31.55	74.28	52.92	64.31	63.41	63.64	63.97	21.36	50.34	-32.90
Full FT–SPQ	69.21	71.43	76.06	75.32	80.65	76.92	15.23	64.80	-55.72	80.34	55.71	62.93	62.66	64.31	65.93	15.38	41.28	-35.95
Full FT–TU	61.82	80.00	78.87	72.73	78.49	74.73	25.83	64.51	-38.43	74.28	63.96	70.24	66.67	67.44	66.84	31.59	55.64	-25.93
Stage 2: Post-Adaptation Memory Tuning (ours)																		
PAMT–LoRA–SPQ	69.21	48.57	64.79	63.64	66.67	59.34	25.18	50.43	-19.82	80.34	31.82	46.90	48.75	49.62	48.96	19.47	31.24	-13.56
PAMT–LoRA–TU	61.82	64.29	66.20	59.74	65.59	68.13	32.94	52.76	-11.23	74.28	50.39	61.55	60.90	61.28	61.45	31.28	47.62	-10.84
PAMT–Full FT–SPQ	69.21	68.57	73.24	72.73	78.49	74.73	31.45	61.87	-22.64	80.34	53.25	60.17	60.40	62.18	63.46	27.83	39.15	-14.21
PAMT–Full FT–TU	61.82	77.14	76.06	70.13	76.34	72.53	37.26	61.58	-15.37	74.28	61.69	67.48	64.29	65.38	64.97	42.31	52.87	-9.16

use different aggregate metrics, the contrast indicates that search-space growth alone is insufficient to explain the retention loss observed after sequential adaptation. The primary source of degradation is therefore training-induced interference in the learned query–docid mapping.

Why TU is more stable than SPQ under corpus evolution. The controls in Table 1 also clarify why TU yields a more favorable continual-learning profile than SPQ. Although SPQ is stronger on the initial slice $\mathcal{D}_0$ (MS MARCO: 69.21 vs. 61.82; NQ: 80.34 vs. 74.28), TU transfers much better to later slices. A likely reason is that TU reuses lexical subwords from the pretrained T5 vocabulary, whereas SPQ relies on dedicated docid tokens and a fixed PQ codebook learned on $\mathcal{D}_0$. Consistent with this, SPQ exhibits substantially higher docs-affected collision rates than TU (Table 1), which increases identifier ambiguity for later-slice documents and plausibly contributes to weaker transferability and less stable continual behavior.

Overall, RQ3 shows that continual GenIR degradation is driven primarily by update-induced interference, while docid design affects how transferable and stable the decoding space remains as the corpus evolves.

6.4 Comparison to prior approaches

Table 2 compares PAMT with index-based retrievers and prior continual GenIR baselines under the *Expanded* protocol. Index-based retrievers provide a stability reference because they maintain an explicit retrieval index rather than storing the corpus as generative docid mappings inside a single model. BM25 is non-parametric, while dense retrievers retain an explicit document index even when the query encoder is updated per session. This comparison places continual GenIR methods on a common spectrum, from retrieval architectures that largely decouple corpus growth from model-internal docid memorization to methods that must preserve legacy query–docid mappings while continuing to adapt.

Prior continual GenIR baselines still exhibit substantial forgetting. Under the Expanded protocol, prior continual GenIR methods incur strongly negative backward transfer, indicating that updates learned for later slices disrupt legacy query–docid mappings. For example, DSI++ reaches $\mathrm{BWT}^{\pm} = -32.45$ on MS MARCO and -28.74 on NQ, while CLEVER yields -26.38 and -24.16, respectively. MixLoRA-DSI is more stable than the other continual GenIR baselines, but still forgets substantially (-22.94 on MS MARCO and -18.43 on NQ). In contrast, index-based retrievers remain much closer to 0 (e.g., DPR-HN: -4.60 on MS MARCO and -4.00 on NQ), reflecting the stability advantage of maintaining an explicit retrieval index rather than a model-internal generative index.

PAMT improves the continual GenIR trade-off. Across both datasets, PAMT substantially improves stability while maintaining strong final-step effectiveness. Under Expanded, PAMT–LoRA–TU achieves $\mathrm{BWT}^{\pm} = -11.23$ on MS MARCO and -10.84 on NQ, improving on all prior continual GenIR baselines in terms of stability. PAMT–Full FT–TU attains the highest AP among continual GenIR

Table 3: Sensitivity of PAMT to protected-row ratio p and value-update budget m on MS MARCO with SPQ identifiers under the Expanded protocol. AP, $\mathrm{FWT}_{\mathrm{diag}}$, and signed $\mathrm{BWT}^{\pm}$ are computed from MRR@10; higher is better. † denotes the default setting.

p	m	AP↑	$\mathrm{FWT}_{\mathrm{diag}}$ ↑	$\mathrm{BWT}^{\pm}$ ↑
0%	2,000	27.80	59.92	-29.84
0%	10,000	29.76	62.44	-26.31
0%	50,000	29.98	**62.73**	-25.88
10%	2,000	29.34	59.46	-25.71
10%	10,000†	31.45	61.87	-22.64
10%	50,000	**31.66**	62.02	-22.31
30%	2,000	29.08	56.91	-23.96
30%	10,000	30.84	59.28	-20.87
30%	50,000	31.02	59.41	**-20.51**
50%	2,000	26.92	52.43	-24.68
50%	10,000	28.47	54.96	-21.93
50%	50,000	28.59	55.11	-21.61

methods, reaching 37.26 on MS MARCO and 42.31 on NQ, while also achieving strong stability on NQ ($\mathrm{BWT}^{\pm} = -9.16$). The gains are consistent across adaptation regimes and identifier schemes. These improvements are not obtained by simply sacrificing plasticity: PAMT retains strong diagonal performance on later slices while materially improving retention on earlier ones.

The comparison also reinforces the role of identifier design. Within our GenIR variants, TU consistently yields better retention and higher AP than SPQ, while PAMT improves stability under both identifier schemes. For example, under Expanded with PAMT–Full FT, TU improves AP from 31.45 to 37.26 on MS MARCO and from 27.83 to 42.31 on NQ relative to SPQ. Combined with the RQ3 controls, this pattern is consistent with TU providing a more transferable and stable decoding space under corpus evolution. Overall, PAMT materially narrows the stability gap between continual GenIR and classical retrievers through sparse post-adaptation calibration of the decoding interface, although index-based retrievers remain more stable overall.

6.5 Sensitivity to protected capacity and update budget

We ablate the protected-row ratio p and value-update budget m on MS MARCO with SPQ identifiers under the Expanded protocol as a representative sensitivity setting. The protected ratio p controls how many historically accessed PMH value rows are excluded from Stage 2 updates, while m controls how many remaining rows are updated. We vary $p \in \{0\%, 10\%, 30\%, 50\%\}$ and $m \in \{2{,}000, 10{,}000, 50{,}000\}$; the main experiments use $p = 10\%$ and $m = 10{,}000$ across datasets and identifier schemes without further tuning.

Table 3 shows a capacity–retention trade-off. Without hard protection ($p = 0\%$), AF×IHF selection yields worse $\mathrm{BWT}^{\pm}$ and lower AP despite marginally higher $\mathrm{FWT}_{\mathrm{diag}}$, indicating that explicit protection of historically accessed rows contributes to retention beyond AF×IHF alone. Increasing p improves retention up to $p = 30\%$, with the best $\mathrm{BWT}^{\pm}$ obtained at $p = 30\%$, $m = 50{,}000$ (−20.51), but this comes at the cost of lower AP and lower $\mathrm{FWT}_{\mathrm{diag}}$ than the best AP-oriented settings. Increasing protection further to $p = 50\%$ degrades all metrics relative to $p = 30\%$, suggesting that over-protection leaves too little capacity for current-slice calibration. Along the budget axis, gains largely saturate beyond $m = 10{,}000$: at $p = 10\%$, increasing m from 10,000 to 50,000 improves AP by only 0.21 and $\mathrm{BWT}^{\pm}$ by 0.33 while updating five times more rows. We therefore use $p = 10\%, m = 10{,}000$ as a balanced default, near the best AP while preserving update sparsity.

7 Conclusion

We studied continual generative information retrieval (GenIR) under evolving corpora on MS MARCO and Natural Questions using semantic product-quantized (SPQ) and title–URL (TU) identifiers. Across settings, sequential Stage 1 adaptation exhibits a clear stability–plasticity trade-off: models acquire retrieval behavior for newly added slices but incur strongly negative backward transfer on earlier ones. Additional controls show that this degradation is driven primarily by update-induced interference in the learned query–docid mapping, rather than by candidate-set growth alone. To address this, we introduced *Post-Adaptation Memory Tuning* (PAMT), an adapt-then-stabilize framework that freezes the adapted backbone and applies a memory-only stabilization stage through a modular parametric memory head. By updating only a sparse subset of memory values while keeping the backbone and PMH routing components fixed during Stage 2, PAMT improves retention without further backbone optimization or replay of legacy relevance supervision. Overall, our results indicate that a separate post-adaptation stabilization step can reduce forgetting in continual GenIR.

Limitations and future directions. PAMT leaves several directions for future work. First, PAMT relies on fixed design choices, notably the protected-row ratio p and update budget m. Our ablations suggest that $m = 10{,}000$ captures most Stage 2 gains, but future work could explore broader sensitivity, alternative row-selection rules, and adaptive budgets m_t driven by slice novelty or PMH access divergence. Second, PAMT evaluates the retrieval-level effect of Stage 1 adaptation via signed $\mathrm{BWT}^{\pm}$, but does not quantify or constrain the induced routing changes in PMH; routing-consistency diagnostics, anchor-based routing, or access regularization could clarify when value-only calibration suffices and when routing-level intervention is needed. Third, we do not report wall-clock adaptation cost or inference latency/throughput; although PMH projections are restricted to trie-valid tokens, a full systems-level efficiency analysis remains future work. Fourth, hard protection of high-usage rows could be replaced by soft protection that scales gradients by historical importance, better accommodating stale mappings or intentional forgetting. Finally, our experiments cover 320K-scale corpora, five sessions, and two identifier schemes; longer horizons, larger corpora and backbones, broader docid designs, and joint-retraining upper bounds remain important next steps.

Acknowledgments

This research was (partially) supported by the Dutch Research Council (NWO), under project numbers 024.004.022, NWA.1389.20.183, and KICH3.LTP.20.006, and the European Union under grant agreement No. 101201510 (UNITE). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of their respective employers, funders and/or granting authorities.

References

[1] Vincent-Pierre Berges, Barlas Oğuz, Daniel Haziza, Wen-tau Yih, Luke Zettlemoyer, and Gargi Ghosh. 2024. Memory layers at scale. *arXiv preprint arXiv:2412.09764* (2024).

[2] Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, Li Deng, and Bhaskar Mitra. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv preprint arXiv:1611.09268* (2016).

[3] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive Entity Retrieval. In *Proceedings of EMNLP*.

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2019. Continual learning with tiny episodic memories. In *ICML*.

[5] 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 *CIKM 2023: 32nd ACM International Conference on Information and Knowledge Management*. ACM, 306–315.

[6] Matthias De Lange, Rahaf Aljundi, Marc Masana, et al. 2022. A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[7] Robert M. French. 1999. Catastrophic Forgetting in Connectionist Networks. *Trends in Cognitive Sciences* 3, 4 (1999), 128–135.

[8] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.). Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 5484–5495. https://doi.org/10.18653/v1/2021.emnlp-main.446

[9] 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* abs/2402.16767 (2024). https://api.semanticscholar.org/CorpusID:268032853

[10] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. LoRA: Low-rank Adaptation of Large Language Models.. In *ICLR*.

[11] Tuan-Luc Huynh, Thuy-Trang Vu, Weiqing Wang, Trung Le, Dragan Gašević, Yuan-Fang Li, and Thanh-Toan Do. 2025. MixLoRA-DSI: Dynamically Expandable Mixture-of-LoRA Experts for Rehearsal-Free Generative Retrieval over Dynamic Corpora. *arXiv preprint arXiv:2507.09924* (2025).

[12] 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* (2024).

[13] David Isele and Akansel Cosgun. 2018. Selective Experience Replay for Lifelong Learning. In *AAAI*.

[14] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2010), 117–128.

[15] 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)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 6769–6781. https://doi.org/10.18653/v1/2020.emnlp-main.550

[16] Gyuwan Kim and Tae Hwan Jung. 2020. Large Product Key Memory for Pretrained Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. 4060–4069.

[17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences* 114, 13 (2017), 3521–3526.

[18] Varsha Kishore, Chao Wan, Justin Lovelace, Yoav Artzi, and Kilian Q Weinberger. 2023. IncDSI: Incrementally Updatable Document Retrieval. In *International Conference on Machine Learning*. PMLR, 17122–17134.

[19] 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 (2019), 452–466. https://doi.org/10.1162/tacl_a_00276

[20] Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2019. Large Memory Layers with Product Keys. *Advances in Neural Information Processing Systems* 32 (2019).

[21] Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023. Multiview Identifiers Enhanced Generative Retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023)*. 6636–6648.

[22] Zhizhong Li and Derek Hoiem. 2017. Learning without Forgetting. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 40. IEEE, 2935–2947.

[23] 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 International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2356–2362. https://doi.org/10.1145/3404835.3463238

[24] Jessy Lin, Luke Zettlemoyer, Gargi Ghosh, Wen-Tau Yih, Aram Markosyan, Vincent-Pierre Berges, and Barlas Oğuz. 2025. Continual Learning via Sparse Memory Finetuning. *arXiv preprint arXiv:2510.15103* (2025).

[25] Bo Liu, Qiang Liu, and Peter Stone. 2022. Continual Learning and Private Unlearning. In *Conference on Lifelong Learning Agents*. PMLR, 243–254.

[26] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient Episodic Memory for Continual Learning. In *Advances in Neural Information Processing Systems*. 6467–6476.

[27] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, and Xueqi Cheng. 2022. Pretrain a Discriminative Text Encoder for Dense Retrieval via Contrastive Span Prediction. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 848–858. https://doi.org/10.1145/3477495.3531772

[28] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, Benjamin Bossan, and Marian Tietz. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. https://github.com/huggingface/peft.

[29] Michael McCloskey and Neal J Cohen. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Psychology of Learning and Motivation*. Vol. 24. Elsevier, 109–165.

[30] Sanket Vaibhav Mehta, Jai Gupta, Yi Tay, Mostafa Dehghani, Vinh Q. 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*. 8198–8213.

[31] Kidist Amde Mekonnen, Yubao Tang, and Maarten de Rijke. 2025. Lightweight and Direct Document Relevance Optimization for Generative Information Retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Padua, Italy) *(SIGIR '25)*. Association for Computing Machinery, New York, NY, USA, 1327–1338. https://doi.org/10.1145/3726302.3730023

[32] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking Search: Making Domain Experts Out of Dilettantes. *SIGIR Forum* 55, 1, Article 13 (July 2021), 27 pages. https://doi.org/10.1145/3476415.3476428

[33] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.). Association for Computational Linguistics, Online, 5835–5847. https://doi.org/10.18653/v1/2021.naacl-main.466

[34] Ruiyang Ren, Wayne Xin Zhao, Jing Liu, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. TOME: A Two-stage Approach for Model-based Retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 6102–6114. https://doi.org/10.18653/v1/2023.acl-long.336

[35] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*. National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, USA, 109–126. https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/

[36] Anthony V. Robins. 1995. Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connect. Sci.* 7 (1995), 123–146. https://api.semanticscholar.org/CorpusID:22882861

[37] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. 2019. *Experience Replay for Continual Learning*. Curran Associates Inc., Red Hook, NY, USA.

[38] 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*. 2410–2420.

[39] Yubao Tang, Ruqing Zhang, Jiafeng Guo, Jiangui Chen, Zuowei Zhu, Shuaiqiang Wang, Dawei Yin, and Xueqi Cheng. 2023. Semantic-enhanced Differentiable Search Index Inspired by Learning Strategies. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4904–4913.

[40] Yubao Tang, Ruqing Zhang, Jiafeng Guo, and Maarten de Rijke. 2023. Recent Advances in Generative Information Retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*. 294–297.

[41] Yubao Tang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, and Xueqi Cheng. 2024. Generative Retrieval Meets Multi-graded Relevance. *Advances in Neural Information Processing Systems* 37 (2024), 72790–72817.

[42] Yubao Tang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, and Xueqi Cheng. 2024. Listwise Generative Retrieval Models via a Sequential Learning Process. *ACM Transactions on Information Systems* 42, 5 (2024), 1–31.

[43] Yubao Tang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. 2024. Bootstrapped Pre-training with Dynamic Identifier Prediction for Generative Retrieval. In *Findings of the Association for Computational Linguistics ACL 2024*. 10303–10317.

[44] Yubao Tang, Ruqing Zhang, Zhaochun Ren, Jiafeng Guo, and Maarten de Rijke. 2024. Recent Advances in Generative Information Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval)*. Association for Computing Machinery, New York, NY, USA, 3005–3008. https://doi.org/10.1145/3626772.3661379

[45] Yubao Tang, Ruqing Zhang, Weiwei Sun, Jiafeng Guo, and Maarten de Rijke. 2024. Recent Advances in Generative Information Retrieval. In *Companion Proceedings of the ACM Web Conference 2024* (Singapore, Singapore) *(WWW '24)*. Association for Computing Machinery, New York, NY, USA, 1238–1241. https://doi.org/10.1145/3589335.3641239

[46] Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer Memory as A Differentiable Search Index. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 1587, 13 pages.

[47] Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Allen Sun, Weiwei Deng, Qi Zhang, and Mao Yang. 2022. A Neural Corpus Indexer for Document Retrieval. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 1856, 15 pages.

[48] Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. 2024. Scalable and Effective Generative Information Retrieval. In *Proceedings of the ACM Web Conference 2024* (Singapore, Singapore) *(WWW '24)*. Association for Computing Machinery, New York, NY, USA, 1441–1452. https://doi.org/10.1145/3589334.3645477

[49] Hansi Zeng, Chen Luo, and Hamed Zamani. 2024. 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* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 469–480. https://doi.org/10.1145/3626772.3657746

[50] Zhen Zhang, Xinyu Ma, Weiwei Sun, Pengjie Ren, Zhumin Chen, Shuaiqiang Wang, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. 2025. Replication and Exploration of Generative Retrieval over Dynamic Corpora. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Padua, Italy) *(SIGIR '25)*. Association for Computing Machinery, New York, NY, USA, 3325–3334. https://doi.org/10.1145/3726302.3730314

[51] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2023. Enhancing Generative Retrieval with Reinforcement Learning from Relevance Feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 12481–12490.

[52] Yujia Zhou, Jing Yao, Zhicheng Dou, Yiteng Tu, Ledell Wu, Tat-Seng Chua, and Ji-Rong Wen. 2024. ROGER: Ranking-Oriented Generative Retrieval. *ACM Trans. Inf. Syst.* 42, 6, Article 155 (Oct. 2024), 25 pages. https://doi.org/10.1145/3603167

[53] 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* (2022).