A Parametric Memory Head for Continual Generative Retrieval

Kidist Amde Mekonnen

Yubao Tang

Maarten de Rijke

IR Lab AMS

UNIVERSITY OF AMSTERDAM

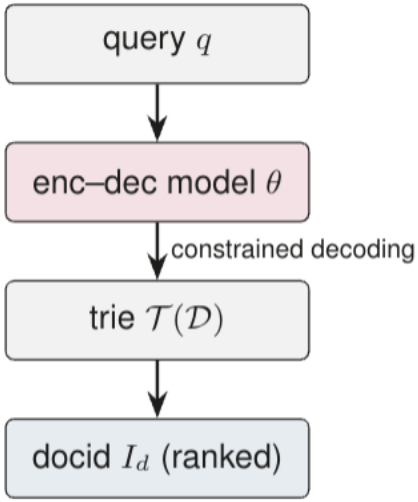
SIGIR2026
20-24 July
MELBOURNE | NAARM
AUSTRALIA

Motivation: Dynamic Collections Challenge Generative Retrieval

- Retrieval systems must learn new documents without forgetting old ones.
- **Traditional IR:** updates **an external index**
- **GenIR**: **must update the model** itself.
- This can make new documents retrievable, but it can also disrupt earlier **query → docid** mappings.

Key challenge: How can GenIR learn new documents without forgetting earlier query–docid mappings?

Recap: Generative Retrieval (GenIR)



GenIR treats retrieval as document-ID (docid) generation

- The model learns query → docid mappings.
- Given a query, it generates docids token by token.
- Inference: beam search is constrained to trie-valid next tokens, ensuring that only valid docids are generated.

Main bottleneck: corpus knowledge lives in the parameters , adding the docid to the trie is trivial, but **reliably retrieving new documents** requires model adaptation.

Stability–Plasticity Trade-off

- **Plasticity**: learning to retrieve newly added documents
- **Stability**: retaining retrieval performance on earlier slices

- **Sequential fine-tuning** buys plasticity at the cost of **catastrophic forgetting** on earlier slices; **joint retraining** preserves both but is **prohibitively expensive** at every update.

How severe is the forgetting, and what causes it?

- **We separate two effects: sequential model adaptation & search-space expansion.**

Sequential adaptation forgets:

- Full FT, SPQ, MS MARCO: $BWT^{\pm}$ = **−55.72**
- Holds for LoRA too (−48.7), fewer parameters ≠ less forgetting

Frozen-θ_0 controls isolate the cause:

- Trie expansion only: Hit@10 drops **≤ 3 pp**
- Zero-shot transfer to future slices: weak (5.6–20.3 Hit@10)

Setup: We first train on D_0, the initial 50% of the corpus. Then, we adapt the model through five sessions, D_1 to D_5, where each session adds 10% more documents. After every session, we evaluate on all document slices seen so far.

		MS MARCO	NQ
Adaptation	**Docid**	$BWT^{\pm}$ ↑	$BWT^{\pm}$ ↑
Full FT	SPQ	−55.72	−35.95
Full FT	TU	−38.43	−25.93
LoRA	SPQ	−48.70	−35.63
LoRA	TU	−31.55	−32.90

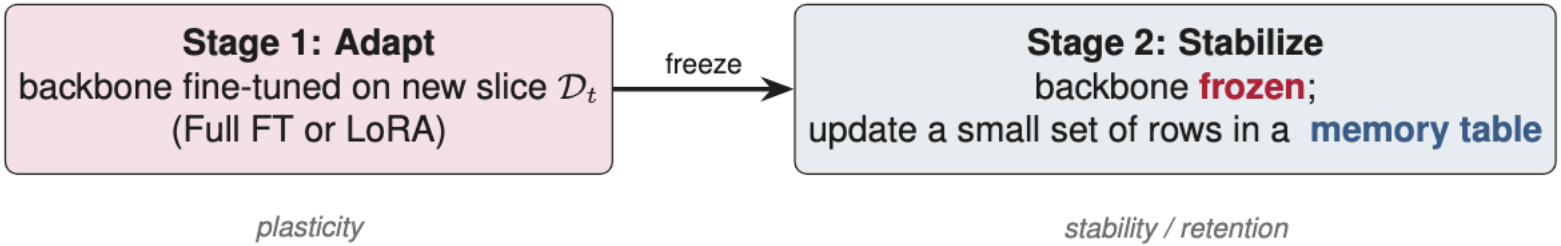
Expanded protocol; $BWT^{\pm}$ computed from MRR@10 (points).

How severe is the forgetting, and what causes it?

- Most of the drop comes from model updates disrupting earlier query–docid mappings, not from the larger search space.
- However, adaptation is still needed because new docids cannot be retrieved well without training.

✓ **Adapt for plasticity, then stabilize for retention.**

PAMT: Adapt, then Stabilize



- PAMT separates learning new query–docid mappings from post-adaptation stabilization.
- Stage 2 uses current-slice supervision only: no backbone re-optimization and no replay of legacy supervision.

- The memory is a trainable lookup table used to re-rank trie-valid docid tokens. It is co-trained with the backbone on D_0. In later sessions, Stage 2 freezes the backbone and memory addressing, and updates only selected value rows.

Parametric Memory Head: Decoder-Side Calibration

At each decoding step

1. The decoder hidden state queries a **product-key memory**.
2. The top-K_{mem} rows are retrieved for each memory head.
3. Retrieved values form a hidden-space correction b_k^{hid}.
4. The correction adjusts logits only for **trie-valid next tokens**.

$$\ell'_k[\tau] = \underbrace{\ell_k[\tau]}_{\text{backbone logit}} + \underbrace{\langle b_k^{\text{hid}}, E[\tau] \rangle}_{\text{memory-based correction}}, \qquad \tau \in A_k(\pi)$$

➔ The next question is: which memory rows should we allow to change, without disrupting the rows earlier slices depend on?

Which memory rows may change? Access statistics decide

Goal: calibrate for the new slice without touching what earlier slices rely on, and without storing legacy queries.

- **Access logs:** scalar counts of which rows each session accessed; no query text is stored.
- **Protected set P_t:** historically frequent rows remain frozen. (Gradients hard-masked to zero)
- **Update set Ω_t:** select m rows that are frequently accessed now but rarely accessed historically.

$$w_t(n) = \underbrace{\widehat{\mathrm{AF}}_t(n)}_{\text{used now}} \cdot \underbrace{\log \frac{Z+1}{\mathrm{AF}_t^{\mathrm{hist}}(n)+1}}_{\text{rarely used before}}$$

Stage 2 footprint

- $m = 10{,}000$ rows $\times\ 768 = 7.68$M trainable parameters
- Stage 1: approximately 260M for Full FT or 26M for LoRA
- Backbone, routing, keys, output embeddings, and protected rows remain frozen

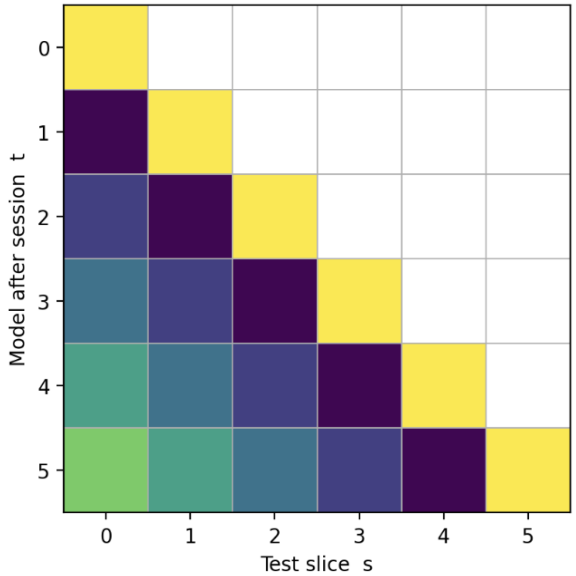
➢ PAMT uses currently useful but historically underused memory capacity.

Experimental Setup: Continual Retrieval over Six Corpus Slices

Factor	Levels
Benchmark	MS MARCO 320K / NQ320K ($\mathcal{D}_0$ = 50%, five 10% slices)
Docid scheme	**SPQ** (PQ codes, $M{=}32$, $K{=}256$) / **TU** (title+URL)
Stage-1 adaptation	**Full FT** / **LoRA** ($r{=}64$)
Search space	**Expanded** trie $\mathcal{T}(\mathcal{D}_{0:t})$ / **Fixed** full-corpus trie

CL evaluation metrics: capture the stability–plasticity trade-off

Lower-triangular evaluation: after each session t, evaluate every seen query set $Q_{test,s}$, $s \leq t$



Summary of CL metrics

- **AP ↑** (Average Performance) — *How good is the final model overall?*
- **FWT_diag ↑** (Forward transfer) — *Can it **learn new documents?***
- **BWT$^{\pm}$ ↑** (Signed backward transfer) — *Does it **retain earlier documents?***

Diagonal: learning current slice ($R_{t,t}$);
Below diagonal: retention on earlier slices ($R_{t,s}$, $s < t$)

PAMT Improves the Continual GenIR Trade-off

Config. (Expanded)		MS MARCO BWT$^{\pm}$ ↑	MS MARCO AP↑	NQ BWT$^{\pm}$ ↑	NQ AP↑
Full FT–SPQ	Stage 1	−55.72	15.23	−35.95	15.38
	+ **PAMT**	−22.64	**31.45**	−14.21	**27.83**
Full FT–TU	Stage 1	−38.43	25.83	−25.93	31.59
	+ **PAMT**	−15.37	**37.26**	−9.16	**42.31**
LoRA–SPQ	Stage 1	−48.70	11.47	−35.63	8.76
	+ **PAMT**	−19.82	**25.18**	−13.56	**19.47**
LoRA–TU	Stage 1	−31.55	24.38	−32.90	21.36
	+ **PAMT**	−11.23	**32.94**	−10.84	**31.28**

PAMT consistently makes signed BWT less negative and increases AP across both adaptation methods, docid schemes, and datasets.

- Forgetting is reduced by roughly 2–3× across all configurations , for example, from −55.7 to −22.6 on MS MARCO while Stage 2 updates only 7.7M parameters.
- TU is more stable than SPQ, but PAMT improves both docid schemes.

PAMT Improves over Prior Continual GenIR Methods

Method	MS MARCO		NQ	
	AP↑	BWT$^{\pm}$ ↑	AP↑	BWT$^{\pm}$ ↑
DSI++	14.82	−32.45	11.23	−28.74
CLEVER	19.56	−26.38	14.67	−24.16
CorpusBrain++	17.24	−29.82	12.84	−26.58
PromptDSI	13.47	−34.67	10.56	−30.21
MixLoRA-DSI	22.83	−22.94	24.56	−18.43
PAMT–Full FT–TU	**37.26**	−15.37	**42.31**	−9.16
PAMT–LoRA–TU	32.94	−11.23	31.28	−10.84

- Overall, PAMT gives a better balance between learning new documents and retaining earlier ones.

- PAMT improves both retention and overall performance.
- PAMT with LoRA gives the least forgetting on MS MARCO, while PAMT with full fine-tuning gives the highest AP on both datasets and the least forgetting on NQ.

Takeaways

- Most forgetting comes from **model updates**, not from expanding the search space alone.
- Adapt, then stabilize: **PAMT substantially improves retention** after adaptation, without replay and with only 7.7M trainable parameters in Stage 2.
- **Docid design matters:** TU identifiers are more stable than SPQ identifiers, but PAMT improves both.

Open directions: Adapt the memory budget automatically, make protection less rigid, and test PAMT over longer update sequences and larger models.

Thank You !

Questions?

IR Lab AMS

UNIVERSITY OF AMSTERDAM

SIGIR2026
20-24 July
MELBOURNE | NAARM
AUSTRALIA