Model Editing for New Document Integration in Generative Information Retrieval

Zhen Zhang
Shandong University
Qingdao, China
zhen.zhang.sdu@gmail.com

Zihan Wang*
University of Amsterdam
Amsterdam, The Netherlands
zhw.cypher@gmail.com

Xinyu Ma
Baidu Inc.
Beijing, China
xinyuma2016@gmail.com

Shuaiqiang Wang
Baidu Inc.
Beijing, China
shqiang.wang@gmail.com

Dawei Yin
Baidu Inc.
Beijing, China
yindawei@acm.org

Xin Xin
Shandong University
Qingdao, China
xinxin@sdu.edu.cn

Pengjie Ren
Shandong University
Qingdao, China
renpengjie@sdu.edu.cn

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

Zhaochun Ren*
Leiden University
Leiden, The Netherlands
z.ren@liacs.leidenuniv.nl

Abstract

Generative retrieval (GR) reformulates the Information Retrieval (IR) task as the generation of document identifiers (docIDs). Despite its promise, existing GR models exhibit poor generalization to newly added documents, often failing to generate the correct docIDs. While incremental training offers a straightforward remedy, it is computationally expensive, resource-intensive, and prone to catastrophic forgetting, thereby limiting the scalability and practicality of GR.

In this paper, we identify the core bottleneck as the decoder's ability to map hidden states to the correct docIDs of newly added documents. Model editing, which enables targeted parameter modifications for docID mapping, represents a promising solution. However, applying model editing to current GR models is not trivial, which is severely hindered by indistinguishable edit vectors across queries, due to the high overlap of shared docIDs in retrieval results. To address this, we propose **DOME** (docID-oriented model editing), a novel method that effectively and efficiently adapts GR models to unseen documents. DOME comprises three stages: (1) identification of critical layers, (2) optimization of edit vectors, and (3) construction and application of updates. At its core, DOME employs a *hybrid-label adaptive training* strategy that learns discriminative edit vectors by combining soft labels, which preserve query-specific semantics for distinguishable updates, with hard labels that enforce precise mapping modifications. Experiments on widely used benchmarks, including NQ and MS MARCO, show that our method significantly improves retrieval performance on new documents while maintaining effectiveness on the original collection. Moreover, DOME achieves this with only about 60% of the training time required by incremental training, considerably reducing computational cost and enabling efficient, frequent model updates.[1]

*Corresponding author.

CCS Concepts

• **Information systems → Retrieval models and ranking**.

Keywords

Model editing, Generative retrieval

ACM Reference Format:
Zhen Zhang, Zihan Wang, Xinyu Ma, Shuaiqiang Wang, Dawei Yin, Xin Xin, Pengjie Ren, Maarten de Rijke, and Zhaochun Ren. 2026. Model Editing for New Document Integration in Generative Information Retrieval. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3774904.3792178

1 Introduction

Generative retrieval (GR) marks a paradigm shift in information retrieval. Instead of searching a pre-built index, GR models directly generate relevant document identifiers (docIDs) in response to a query [2, 38, 40]. GR is gaining increasing attention in the IR community as it enables end-to-end training and shows excellent retrieval performance [5, 50]. However, a critical limitation hinders the practical deployment of GR: poor generalization to new documents, with models often failing to retrieve items added after the initial training phase [16, 26, 51]. A prevalent strategy to address this problem is incremental training, in which the GR model is retrained on new additions when documents are added to the collection [11, 26]. While this enables the model to learn docIDs for new entries, it is computationally expensive, requiring substantial data and resources. Moreover, it is prone to catastrophic forgetting [16, 21, 26], where adapting to new documents diminishes performance on the initial set. These drawbacks significantly limit the scalability and practicality of current GR models.

[1]Our code is available at https://github.com/zhangzhen-research/DOME

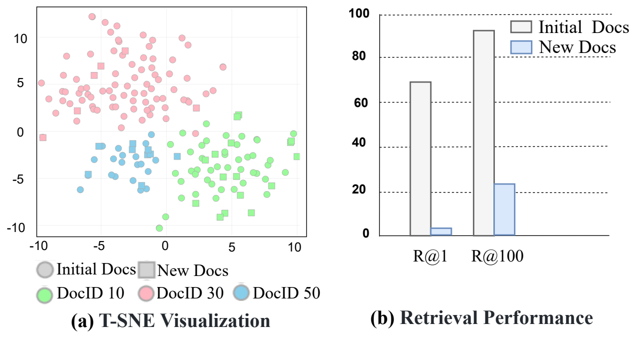


Figure 1: Behavioral analysis of the initial and new documents over the NQ dataset [18] using a hierarchical k-means-based docID GR model [38]. The model is trained on the initial document set, and docIDs for new documents are assigned using the same k-means clustering procedure. (a) T-SNE [24] visualization of intermediate representations, where different colors denote different docID prefixes (i.e., 10, 30, 50), with each prefix corresponding to a specific document type. Squares represent initial documents, and circles represent new documents. (b) Retrieval performance (Recall@1 and Recall@100) on the initial and new document sets.

In this work, we first conduct a detailed analysis to identify the root cause of the poor performance of GR models on newly added documents. Our findings reveal that the key bottleneck lies in the decoder's capacity to map hidden states to the correct docIDs of these documents. As shown in Figure 1(a), the encoder is able to position the newly added documents close to semantically related ones from the initial training set in the representation space. This indicates that the model has successfully captured the semantic content of the new documents. In contrast, Figure 1(b) shows a marked degradation in retrieval accuracy for the newly added documents. This discrepancy indicates that the mapping between semantic representations and specific docIDs is established only during initial training, but fails to generalize to new documents. Consequently, full retraining is unnecessary: The key is to update this mapping for new documents while maintaining retrieval effectiveness on the initial set.

This insight naturally motivates **model editing** as a promising solution, as it enables targeted parameter modifications for docID mapping while avoiding the high computational cost and catastrophic forgetting of incremental training [8, 29]. However, applying model editing to GR is undermined by a fundamental problem: the edit vectors for different queries lack discriminability. In typical editing of subject–relation–object triplets (denoted as $< s, r, o >$), the target objects are often unique. In contrast, GR generates docIDs autoregressively as a sequence of discrete tokens. At each step, the model predicts the next token (target docID) conditioned on the query and the previously generated tokens (prefix docID). The edit requests in GR take the form of <query, prefix docID, target docID> mappings. A key challenge arises from the limited docID vocabulary, which inevitably causes many different queries and docID prefixes to be mapped to the same target docID. Consequently, the edit vectors lose their discriminability for distinct GR contexts. This prevents the model from learning the precise mapping for each query that severely degrades editing accuracy.

To tackle the above challenges, we introduce **DOME** (docID-oriented model editing), a GR-specific editing method designed to efficiently adapt models to new documents without full retraining. DOME operates in three stages: First, the decoder FFN modules that encode docID mapping knowledge are identified through average patching analysis [33], ensuring that updates are confined to the most relevant parameters. Second, edit vectors for new docIDs are optimized using a *hybrid-label adaptive training* strategy that combines soft labels from the model's output distribution, which preserve query-specific semantic diversity, with hard ground-truth labels that enforce precise generation of the correct docID tokens. During training, the emphasis gradually moves from soft labels to hard labels, encouraging discriminative updates at the start and ensuring precise docID generation by the end. Finally, the optimized edit vectors are assembled into constrained updates and applied to the selected FFN modules, allowing the model to incorporate mappings for new documents while maintaining retrieval performance on the original collection. Extensive experiments on two standard benchmarks, Natural Questions (NQ) and MS-MARCO, demonstrate that our method significantly improves retrieval accuracy for new documents while exhibiting strong resilience to catastrophic forgetting. Moreover, DOME significantly reduces the adaptation time compared with conventional incremental training, enabling much faster deployment in real-world retrieval scenarios.

Our contributions are summarized as follows:

- We identify that the core bottleneck in adapting GR to new documents lies in the model's inability to generate newly assigned docIDs, even when it correctly understands their content.
- We propose a hybrid-label adaptive training strategy to address the issue of insufficient discriminability in edit vectors that arises when applying model editing to GR.
- We develop DOME, a specialized and efficient model editing framework for GR, achieving strong retrieval performance on new documents while substantially mitigating catastrophic forgetting and reducing adaptation costs.
- Extensive experiments on NQ and MS-MARCO show that **DOME** consistently outperforms incremental training, while reducing adaptation time by over 40%.

2 Related Work

2.1 Generative retrieval

Generative retrieval (GR) formulates document search as sequence generation, where the model directly outputs a *docID* given a query, replacing index-based lookup. DocIDs can be *textual* identifiers such as titles or pseudo queries [2, 20, 53], or *numeric* codes composed of discrete tokens [36, 48, 54]. Textual IDs are flexible but incur costly string matching [41, 45]. Numeric IDs are typically derived via clustering or quantization, e.g., k-means [38], PQ [53], RQ [49], or atomic single-number IDs [17]. We focus on structured numeric docIDs (e.g., PQ or RQ), which offer semantic grouping and scalability over atomic IDs.

A key challenge in GR is generalization to unseen documents [16]. DSI++ [26] extends DSI to structured numeric docIDs and improves unseen-document generalization via mixed training and flattened loss. incDSI [17] enables real-time insertion through constrained parameter updates but supports only atomic IDs and requires codebook expansion, limiting applicability to structured docIDs.

2.2 Model editing

Model editing updates internal knowledge through small targeted parameter changes without full retraining [25]. Meta-learning approaches such as KE [4] and MEND [29] use hypernetworks to generate gradient-based edits, enabling fast updates but affecting many parameters. Locate–then–edit methods identify knowledge-bearing components and update only selected parameters [35, 47]. ROME [27] locates FFN neurons encoding specific facts and applies closed-form updates, while MEMIT [28] supports batch edits. AlphaEdit [9] further introduces orthogonal projections to reduce forgetting in lifelong editing.

In this work, we propose **DOME**, a GR adaptation framework that leverages model editing to incorporate new documents after initial training. Unlike incremental retraining [16, 22, 23, 26], DOME avoids expensive updates and mitigates forgetting through selective parameter edits. In contrast to existing editing methods [9, 27, 28], which struggle with GR due to indistinguishable edit vectors, DOME introduces a hybrid-label adaptive training strategy that combines soft-label semantic diversity with hard-label precision, gradually shifting to pure hard labels for exact docID mapping. This GR-specific design enables efficient integration of new identifiers while preserving retrieval performance on both original and newly added documents.

3 Preliminaries

3.1 Generative retrieval

Generative Retrieval (GR) formulates document retrieval as a sequence generation task. Given a query q, a GR model directly generates the document identifier (docID) of the most relevant document [5]. The docID is represented as a sequence of discrete tokens $\boldsymbol{y} = (y_1, \dots, y_T)$. These tokens are often assigned via structured schemes such as hierarchical k-means clustering [1].

The decoder generates the docID sequence $\boldsymbol{y}$ auto-regressively, predicting each token y_t conditioned on the query and the docID prefix $y_{<t}$ as follows:

$$p(\boldsymbol{y}|q) = \prod_{t=1}^{T} p(y_t|q, y_{<t}). \tag{1}$$

The training objective is to maximize this conditional likelihood, which corresponds to minimizing the negative log-likelihood loss for a query-document pair (q, d):

$$\mathcal{L}_{\text{GR}} = -\sum_{t=1}^{T} \log p(y_t|q, y_{<t}; \theta), \tag{2}$$

where θ represents the GR model parameters [37].

Adapting to new documents. Our goal is to adapt a pre-trained GR model, originally trained on a corpus C_{init}, to effectively retrieve documents from a newly added corpus C_{new} [26]. The model's parameters allow it to correctly generate docIDs for documents in C_{init}, but it often fails to produce the correct docID sequence for a query q whose relevant document lies in C_{new}. Instead of relying on incremental training, we formulate this adaptation as a model editing problem, where the model is updated with new docID mappings for C_{new} while preserving its performance on C_{init}.

3.2 Model editing

Model editing aimsg to efficiently update specific knowledge within a language model without full retraining, while preserving its original knowledge and performance [13, 15]. In typical settings, stored knowledge is represented as subject–relation–object triplets $< s, r, o >$ [6, 12, 42–44], and an editing request specifies a behavioral modification such that the model outputs the updated object o for a given (s, r) pair [52].

In transformer architectures, a feed-forward network (FFN) module can be interpreted as a *key–value memory* [10]: the *key vector* k encodes the subject–relation pair (s, r), while the *value vector* v encodes the object o. Specifically, given a hidden input state h, the input projection W_{in} followed by a non-linear activation σ produces a *key* vector:

$$k = \sigma(W_{\text{in}} h) \in \mathbb{R}^{d_0}, \tag{3}$$

where d_0 denotes the dimension of the FFN's intermediate layer. The output projection $W_{\text{out}} \in \mathbb{R}^{d_1 \times d_0}$ then maps k to a *value* vector:

$$m = W_{\text{out}} k \in \mathbb{R}^{d_1}, \tag{4}$$

where d_1 denotes the dimension of the FFN's output layer.

Model editing modifies the output projection W_{out} by adding an update matrix Δ. Suppose we aim to perform u edits to inject new knowledge. For each edit $< s_i, r_i, o_i >$, we obtain a new key–value pair (k_i, v_i) representing the desired knowledge modifications, where each k_i is computed by feeding the context (s_i, r_i) into W_{in} followed by the non-linear activation σ, and v_i is the desired value, namely the target output vector for o_i that the model should produce. Stacking all u keys and values yields $K_1 = [k_1 \mid k_2 \mid \cdots \mid k_u] \in \mathbb{R}^{d_0 \times u}$ and $V_1 = [v_1 \mid v_2 \mid \cdots \mid v_u] \in \mathbb{R}^{d_1 \times u}$, where K_1 denotes the keys and V_1 their desired values [39].

To obtain v_i, we first compute the model's original output

$$m_i = W_{\text{out}} k_i, \tag{5}$$

and then determine a minimal editing vector δ_i that shifts the prediction toward the correct object o_i:

$$\delta_i = \arg\min_{\delta} -\log p_{\theta, m_i+\delta}(o_i \mid s_i, r_i), \tag{6}$$

where $p_{\theta,m_i+\delta}$ denotes the model's output distribution when the FFN output is replaced with $m_i + \delta$. The updated value is then given by $v_i = m_i + \delta_i$, and stacking all updated keys and values yields the complete K_1 and V_1.

To mitigate forgetting [7], we additionally construct a set of preserved key–value pairs (K_0, V_0) representing knowledge that should remain unchanged. Here, K_0 is obtained in the same way as K_1 but from original knowledge, and $V_0 = W_{\text{out}} K_0$ stores their original outputs. The update matrix Δ is then computed to align (K_1, V_1) while minimally disturbing (K_0, V_0):

$$\Delta = \arg\min_{\tilde{\Delta}} \ \|(W_{\text{out}} + \tilde{\Delta})K_1 - V_1\|_F^2 + \|\tilde{\Delta} K_0\|_F^2, \tag{7}$$

where $\|\cdot\|_F$ denotes the Frobenius norm [3]. This objective has the closed-form solution:

$$\Delta = (V_1 - W_{\text{out}} K_1) \left(K_1 K_1^\top + K_0 K_0^\top\right)^{-1} K_1^\top. \tag{8}$$

In practice, K_0 can be estimated from abundant auxiliary inputs in the original corpus to approximate the preserved knowledge subspace.

Editing for GR. In generative retrieval, the editing process targets the auto-regressive generation of document identifiers [38]. Each edit request can be represented as $< q, y_{<t}, y_t >$, where q is the input query, $y_{<t}$ is the generated docID prefix up to step $t-1$, and y_t is the desired next docID token. In this setting, the FFN key k is computed from the decoder hidden state conditioned on $(q, y_{<t})$, and the corresponding value v encodes the target docID token y_t [39]. By stacking multiple such keys and values into matrices $K_1 \in \mathbb{R}^{d_0 \times u}$ and $V_1 \in \mathbb{R}^{d_1 \times u}$, the editing framework can directly adjust the model's internal key–value representations to produce the intended outputs for the given query–prefix contexts.

4 Discussion

Before presenting our proposed method, we conduct a comprehensive analysis to identify the docID mapping bottleneck that limits the adaptation of current GR models to new documents (see §4.1). We further point out that existing model editing methods cannot fully resolve this issue, as their effectiveness is severely hindered by indistinguishable edit vectors across queries (see §4.2).

4.1 DocID mapping bottleneck

Recent mechanistic analyses of generative retrieval (GR) [19, 33] show that the encoder captures the semantic content of inputs, while the decoder generates the corresponding docIDs. The decoding process can be conceptually divided into three stages: (1) a priming stage that initializes the task representation, (2) a bridging stage where cross-attention extracts semantics from the encoder outputs, and (3) an interaction stage where FFN modules transform semantic features into final docID tokens [33].

Table 1: Comparison of Recall@10 and accuracy rates of the GR model [38] for initial and added documents on NQ [18].

		Accuracy Rate of the ith DocID		
	Recall@10	**DocID 1**	**DocID 2**	**DocID 3**
Initial Docs	0.862	0.774	0.772	0.757
New Docs	0.173	0.423	0.385	0.420

As Figure 1(a) shows, although the encoder produces meaningful representations for new documents, the decoder fails to generate their correct docIDs. To analyze this failure, we examine docID-level prediction accuracy. As shown in Table 1, the decoder achieves 42.3% accuracy on individual docID tokens but only 17.3% on complete docID sequences. This gap indicates that the model learns an approximate rather than an exact mapping from semantics to docIDs. It can predict partial tokens but not the precise sequence required for retrieval. Hence, the core challenge in adapting GR models to new documents lies in their inability to adjust this learned docID mapping. Therefore, instead of retraining the entire model, it is more effective to selectively update the decoder parameters responsible for docID mapping, thereby overcoming this bottleneck and improving retrieval for newly added documents.

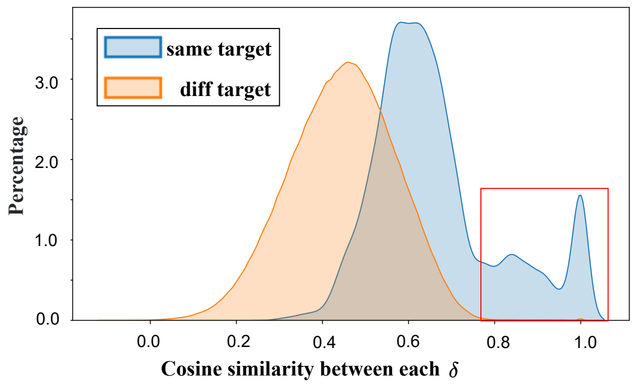


Figure 2: Percentage of pairwise cosine similarity of edit vector δ across GR edit requests applied to the GR method [38] on NQ [18]. Yellow denotes pairs with different target docID, blue denotes pairs with the same target docID.

4.2 Indistinguishable edit vectors in GR

As established in §4.1, GR models struggle to generate exact docIDs for new documents due to misaligned mappings between encoded semantics and discrete docID tokens. Model editing provides a promising solution by enabling targeted parameter updates without full retraining, but its effectiveness in GR is limited by indistinguishable edit vectors across queries. Specifically, as mentioned in §3.2, each GR editing request can be represented as a triple $\langle q, y_{<t}, y_t \rangle$, where $(q, y_{<t})$ is the input context and y_t is the target docID token. Because docID vocabularies are limited, many distinct contexts $(q, y_{<t})$ map to the same target token y_t. When optimized under a one-hot label objective (Eq. 10), these requests share identical supervision signals, leading to nearly identical edit vectors δ_i even for semantically distinct queries. This lack of discriminability prevents the model from learning context-specific query–docID mappings.

To investigate this, we construct GR editing requests for new documents, compute their edit vectors δ from the targeted FFN layers, and measure pairwise cosine similarity [46]. As shown in Figure 2, requests sharing the same target token exhibit high similarity between their edit vectors, confirming that one-hot hard-label optimization produces nearly indistinguishable updates, making it difficult for the model to learn precise docID mappings.

5 DOME: DocID-Oriented Model Editing for Generative Retrieval

DOME is a GR-specific model editing framework designed to inject new docID mappings into a pre-trained GR model while preserving its original retrieval ability. As shown in Figure 3, DOME consists of three stages: (a) Identification of critical layers: locating the decoder's critical FFN layers responsible for docID mapping via average patching technology; (b) Optimization of edit vectors: computing diverse and effective edit vectors using a *hybrid-label adaptive training* strategy, thereby mitigating the indistinguishable edit vectors problem. (c) Construction and application of updates: assembling these edit vectors into a constrained update matrix

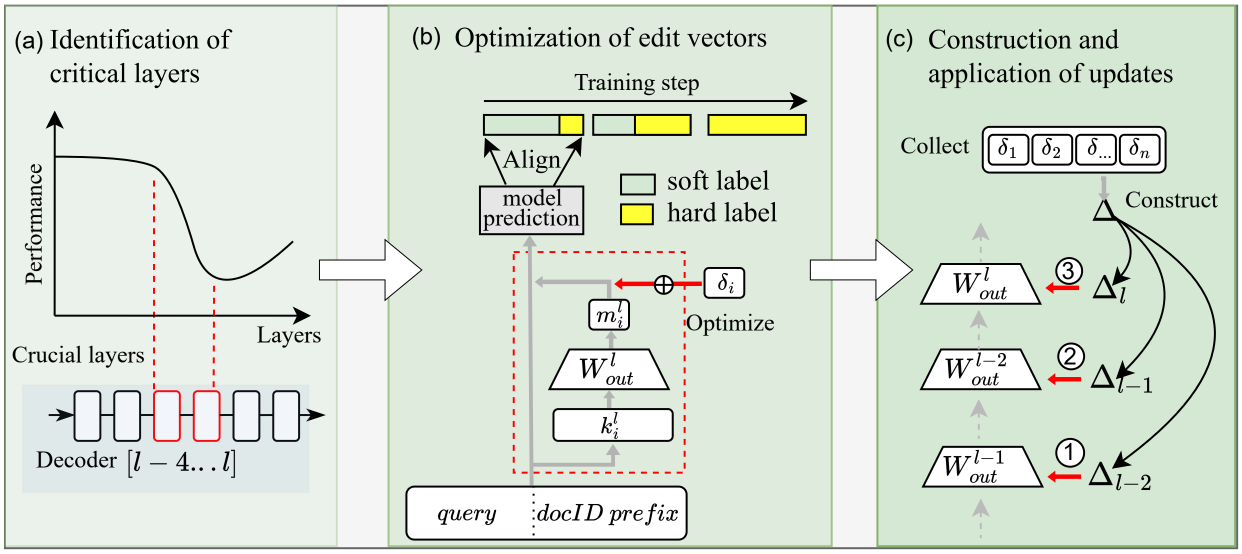

(a) Identification of critical layers
Performance
Layers
Crucial layers
Decoder $[l-4\dots l]$
(b) Optimization of edit vectors
Training step
Align
model prediction
soft label
hard label
δ_i
Optimize
m_i^l
W_{out}^l
k_i^l
query
docID prefix
(c) Construction and application of updates
Collect
δ_1 δ_2 $\delta_{\dots}$ δ_n
Construct
Δ
W_{out}^l ③ Δ_l
W_{out}^{l-2} ② Δ_{l-1}
W_{out}^{l-1} ① Δ_{l-2}

Figure 3: An overview of the proposed DOME framework. (a) A patching technique is used to diagnose the model and locate the critical layers responsible for docID mapping. (b) Our hybrid-label adaptive training strategy is employed to compute diverse and effective edit vectors (δ) for the new mappings. For clarity, the diagram retains only the crucial W_{out} component from the FFN layer. (c) These edit vectors are then used to construct an update matrix (Δ), which is applied to the crucial layer parameters (W_{out}) to inject the new docID mapping.

and sequentially applying it to the targeted layers to inject new mappings while retaining existing knowledge.

5.1 Identification of critical layers

Figure 3(a) illustrates the first stage of our workflow, which aims to locate the crucial layers that store docID-mapping knowledge. We adopt an *average patching* strategy to identify these critical layers [33]. Specifically, we first collect the output representations of all modules in the decoder's intermediate layers over a document set. Then, for another set of documents, we replace each layer's original output with the average representation computed from the training data. By measuring the retrieval performance drop after each replacement, we determine which five layers have the greatest impact when patched. We regard the five layers from $l-4$ to l as critical for docID mapping, as altering their outputs significantly degrades retrieval accuracy. Subsequent model-editing operations are performed exclusively on these identified layers to ensure targeted and efficient adaptation. Detailed analyses are provided in Appendix A.1.

5.2 Optimization of edit vectors

Figure 3(b) illustrates the second stage of DOME, which computes the edit vectors for injecting new docID mappings into the critical decoder layers identified in Section 5.1. We construct a series of GR editing requests from the new documents. For each new document, we generate pseudo-queries whose relevant target is this document, and pair them with its assigned docID sequence $(y_1, \dots, y_T)$ [32]. Following the request format defined in Section 3.2, each request is represented as $< query, docID\ prefix, target\ docID >$.

At the core of DOME is a **hybrid-label adaptive training** strategy, designed to address the low discriminability of edit vectors in GR identified in §4.2. Our strategy directly tackles the low discriminability of δ by replacing the one-hot hard label with a hybrid label that mixes the model's original output distribution (soft label) with the ground-truth token (hard label) [30]:

$$p_{\text{target}}^{(s)}(v \mid q, y_{<t}) = (1-\lambda_s)\, p_{\text{orig}}(v \mid q, y_{<t}) + \lambda_s\, \mathbf{1}[v = y_t], \quad (9)$$

where v is the docID vocabulary token, p_{orig} is the model's original output distribution, $\mathbf{1}[v = y_t]$ is the one-hot distribution of the correct token, and $\lambda_s \in [0, 1]$ gradually increases from a small initial value to 1.0 as training proceeds. The soft component preserves query-specific diversity across edit vectors, while the hard component enforces accurate prediction.

For each token-level editing request, we compute the edit vector δ_i at the final critical layer l. The hybrid label is then used to optimize the edit vector δ_i by modifying the optimization in Eq. 10 to:

$$\delta_i = \arg\min_{\delta_i} \left(-\sum_{v \in \mathcal{V}} p_{\text{target}}^{(s)}(v \mid q_i, y_{<t,i}) \log p_{\theta, m_i+\delta_i}(v \mid q_i, y_{<t,i}) \right), \quad (10)$$

where $p_{\theta, m_i+\delta_i}$ denotes the model's output distribution when the FFN output is patched with $m_i + \delta_i$. Since p_{orig} varies across queries, the resulting $p_{\text{target}}^{(s)}$ and optimal δ_i also differ, encouraging diverse update directions in the early stage. As λ_s approaches 1.0, training focuses increasingly on the hard label, ensuring that the learned mappings generate the exact docID tokens for new documents.

5.3 Construction and application of updates

Figure 3(c) illustrates the final stage of DOME, where the optimized edit vectors are integrated into the targeted decoder layers while avoiding catastrophic forgetting. For each identified FFN layer l, we collect two sets of key–value pairs: $K_1^{(l)}$ contains the key activations from the new GR editing requests, and $V_1^{(l)}$ stores the corresponding target values obtained by adding the optimized edit vectors δ to the FFN's original outputs. To preserve existing knowledge, we also gather $K_0^{(l)}$ and $V_0^{(l)} = W_{\text{out}}^{(l)} K_0^{(l)}$ from query–docID pairs in the original corpus.

Our objective is to align the output of the final identified layer l with V_1^l. Following the constrained closed-form solution in Eq. 8, we first compute the update matrix Δ for this final layer directly using K_0^l and K_1^l.

To further reduce the risk of knowledge forgetting, DOME distributes this update across multiple preceding layers, similar to strategies used in prior work. Specifically, for the five layers before the final target layer, the update matrix at layer j is scaled proportionally to the inverse of its distance from l_{f}:

$$\begin{aligned}\Delta^{(j)} = &\frac{1}{\text{dist}(j,l)}\left(V_1^{(l)} - W_{\text{out}}^{(l)} K_1^{(l)}\right) \\ &\left(K_0^{(j)} K_0^{(j)\top} + K_1^{(j)} K_1^{(j)\top}\right)^{-1} K_1^{(j)\top},\end{aligned} \tag{11}$$

where $\text{dist}(j,l) = |\, l - j + 1\, |$ denotes the absolute layer distance [28]. These updates are applied sequentially from the earliest of the targeted layers toward the final layer l. After applying Δ_j at a given layer, we re-run the forward pass to refresh the key activations $K_1^{(l)}$ for subsequent layers, ensuring that each update is based on up-to-date intermediate representations. This stage precisely injects the new docID mappings into the decoder, preserving established retrieval capabilities and completing DOME's adaptation to the newly added corpus.

6 Experiments

We aim to answer the following research questions: (**RQ1**) Does DOME effectively and efficiently adapt generative retrieval (GR) models to newly added documents? (**RQ2**) Does DOME preserve retrieval performance on the original document set, thereby mitigating catastrophic forgetting? (**RQ3**) Does each core component of DOME contribute to its overall performance? (**RQ4**) Does DOME maintain stable performance as the number of newly added documents increases?[2]

6.1 Experimental setup

Baselines. We compare DOME with three categories of baselines: (1) **No training.** This category keeps all model parameters frozen after adding new documents, relying solely on the model's inherent generalization ability. It includes the sparse retriever **BM25** [34], the dense retriever **DPR** [14], and the generative retrievers **DSI** [38] and **Ultron** [53], all used without parameter updates. (2) **Incremental training.** These methods update model parameters through additional training to incorporate new document knowledge. They include **From Scratch**, which trains a GR model on the combined corpora; **New-Doc FT** [26], which fine-tunes the initial GR model using queries and pseudo-queries for new documents; and **DSI++** [26], which fine-tunes on both initial and new queries while promoting flatter loss basins to mitigate catastrophic forgetting. (3) **Model editing.** These methods directly inject new knowledge into targeted parameters without full retraining. We include **ROME** [27], **MEMIT** [28], and **AlphaEdit** [9], originally designed for knowledge editing but adapted here for GR by redefining edit requests as docID mappings and limiting updates to critical FFN modules. For fairness, all incremental training and model editing baselines start from the same GR model trained on the original corpus and adapt to the same set of newly added documents.

Datasets. We conduct experiments on two standard benchmarks: **Natural Questions (NQ)** [18] and **MS-MARCO** [31]. To simulate the addition of new documents in real-world settings, each corpus is partitioned into 90% for training the base GR model and 10% as the new document set. This setup allows us to evaluate the model's ability to adapt to new documents while retaining retrieval performance on the original collection.

Metrics. Retrieval performance is evaluated using Recall@K and Mean Reciprocal Rank (MRR). Following [26], we further compute the forgetting score (F_n) based on R@10 to quantify performance degradation on the initial corpus. Efficiency is measured by the update time per document (TPD), defined as the total training or editing time divided by the number of newly added documents.

Implementation Details. Experiments are conducted on four NVIDIA A100 GPUs. We adopt T5-large[3] as the base GR model, jointly optimizing query to docID, document to docID, and pseudo-query to docID generation tasks. Pseudo-queries for new documents are generated using the docTTTTTquery model [32]. For DOME, based on our layer importance analysis, editing is applied to the FFN modules in decoder layers 14 to 18. In the hybrid label adaptive training strategy, the mixing coefficient λ_s increases linearly from 0.3 to 1.0 over 50 optimization steps. We use residual quantization (RQ) for docID assignment in the main experiments, while alternative schemes, including BM25 based identifiers and product quantization (PQ), are examined in ablation studies. For baseline, the sparse retriever BM25 is implemented with the *bm25s*[4] library, and the dense retriever DPR with the *pyserini*[5] toolkit. Other GR baselines are reproduced from their official codebases for fair comparison. All model editing methods operate on the same pre-trained GR model trained on the original corpus, restricting edits to the decoder layers identified by our analysis.

6.2 Results on new documents (RQ1)

To answer RQ1, we evaluate the overall performance and adaptation efficiency of DOME.

Performance. We first assess the effectiveness of DOME in adapting the GR model to newly added documents. The results are presented in Table 2. We observe that: (1) Severe generalization failure without adaptation. Models such as DSI and Ultron show drastic performance degradation on new documents, with Recall@1 on NQ dropping to 0.071 and 0.064, respectively. These results clearly confirm the fundamental generalization bottleneck in GR models, as the decoder fails to generate docIDs unseen during initial training. (2) DOME demonstrates effective adaptation. On NQ, DOME attains a Recall@1 of 0.686 on new documents, nearly matching full retraining (0.693) and outperforming the strongest incremental training baseline, DSI++ (0.674). This result indicates that DOME

[2]Due to space constraints, the detailed analysis for RQ4 is provided in Appendix A.2.

[3]https://huggingface.co/google-t5/t5-large
[4]https://github.com/xhluca/bm25s
[5]https://github.com/castorini/pyserini

Table 2: Retrieval performance on the NQ [18] and MS-MARCO [31] datasets. Each cell reports score for *initial documents* / *new documents*. Bold and underlined denote the best and second-best results, respectively, for each metric. "From Scratch" is reported only as a theoretical upper bound and is not considered in ranking due to its high retraining cost. * Significant improvements against the best-performing baseline for each dataset are marked with * (t-test, $p < 0.05$).

Method	NQ				MS-MARCO		
	R@1	R@10	R@100	MRR@100	R@10	MRR@10	TPD (s)
No Training							
BM25	0.374 / 0.365	0.761 / 0.770	0.906 / 0.860	0.476 / 0.471	0.691 / 0.641	0.486 / 0.471	–
DPR	0.651 / 0.647	0.874 / 0.871	0.926 / <u>0.921</u>	0.706 / 0.710	0.767 / 0.754	0.526 / 0.515	–
DSI	0.655 / 0.071	0.866 / 0.193	0.901 / 0.250	0.705 / 0.105	0.541 / 0.056	0.392 / 0.032	–
Ultron	0.671 / 0.064	0.867 / 0.134	0.911 / 0.208	0.722 / 0.168	0.731 / 0.069	0.454 / 0.030	–
Incremental Training							
From Scratch	0.696 / 0.693	0.886 / 0.883	0.931 / 0.930	0.744 / 0.742	0.775 / 0.772	0.532 / 0.529	–
New-Doc FT	0.696 / 0.660	0.886 / 0.848	0.931 / 0.917	0.744 / 0.732	0.775 / 0.751	0.532 / 0.511	<u>3.42</u>
DSI++	0.696 / <u>0.674</u>	0.886 / <u>0.878</u>	0.931 / 0.920	0.744 / <u>0.735</u>	0.775 / <u>0.759</u>	0.532 / <u>0.517</u>	3.54
Model Editing							
ROME	0.696 / 0.201	0.886 / 0.276	0.931 / 0.356	0.744 / 0.326	0.775 / 0.351	0.532 / 0.251	20.23
MEMIT	0.696 / 0.609	0.886 / 0.687	0.931 / 0.756	0.744 / 0.665	0.775 / 0.673	0.532 / 0.495	12.62
AlphaEdit	0.696 / 0.616	0.886 / 0.677	0.931 / 0.745	0.744 / 0.664	0.775 / 0.661	0.532 / 0.490	12.12
DOME	0.696 / **0.686***	0.886 / **0.880***	0.931 / **0.927***	0.744 / **0.740***	0.775 / **0.764***	0.532 / **0.524***	**2.14***

effectively adapts GR models to new corpora while maintaining high retrieval accuracy. (3) DOME substantially surpasses existing model-editing methods. Compared with MEMIT and AlphaEdit, DOME delivers notably higher retrieval accuracy. Existing editing methods struggle to produce discriminative edit vectors when target docIDs overlap, whereas DOME's hybrid label adaptive training enables precise and stable docID mapping.

Adaptation efficiency. We further examine the efficiency aspect of RQ1 by measuring the update time per document (TPD). The results are presented in Table 2, and the main conclusions are as follows: (1) DOME provides remarkably fast and scalable adaptation. DOME achieves a TPD of 2.14 seconds, which is significantly faster than incremental training methods such as DSI++ (3.54s), confirming its high efficiency in updating GR models. (2) DOME achieves substantial speedup over editing baselines. It is more than five times faster than MEMIT (12.62s) and AlphaEdit (12.12s), while maintaining superior adaptation performance. The improved efficiency stems from our carefully designed framework and training strategy, which are tailored to enable effective adaptation of generative retrieval models to new documents.

6.3 Results on initial documents (RQ2)

To address RQ2, we evaluate how well DOME preserves existing knowledge by measuring retrieval performance on the initial document corpus after adaptation. The results are shown in Table 3, and the main findings are as follows: (1) DOME effectively preserves prior knowledge. Its performance on the initial corpus remains nearly unchanged, with Recall@1 decreasing only slightly from 0.696 to 0.692. The forgetting score (F_n) of 0.003, the best among all methods, further demonstrates its strong knowledge retention capability. (2) Knowledge preservation results from constrained updates. As defined in Eq. (8), DOME introduces a constraint based on key-value pairs from the initial corpus (K_0, V_0). The term $K_0 K_0^\top$ ensures that the final update matrix Δ minimally affects outputs for previously learned documents. (3) Other model-editing methods exhibit severe forgetting. ROME collapses to a Recall@1 of 0.285, and both MEMIT and AlphaEdit perform notably worse than DOME. These editing strategies produce edit vectors with poor discriminability across different queries, which leads to interference with existing knowledge and ultimately degrades overall model performance. (4) DOME mitigates interference through hybrid-label adaptive training. By generating query-specific update vectors, DOME avoids conflicting updates and achieves substantially better knowledge retention than competing approaches.

Table 3: Retrieval performance on NQ, reported separately for initial documents. Bold denotes the best result for each metric, and * indicates significant improvement (t-test, $p < 0.05$).

Method	R@1	R@10	R@100	MRR@100	$F_n \downarrow$
Base Model	0.696	0.886	0.931	0.744	–
New-Doc FT	0.544	0.741	0.805	0.612	0.125
DSI++	0.674	0.879	0.926	0.721	0.007
ROME	0.285	0.569	0.774	0.355	0.317
MEMIT	0.654	0.866	0.904	0.705	0.020
AlphaEdit	0.647	0.852	0.891	0.696	0.034
DOME	**0.692***	**0.883***	**0.928***	**0.736***	**0.003***

6.4 Ablation study (RQ3)

To answer RQ3, we conduct an ablation study to analyze the contribution of DOME's key components: the number of pseudo queries, the docID assignment, the edited decoder layer range, and the hybrid-label training strategy. The results on the NQ new document set are shown in Table 4.

Table 4: Ablation study on NQ.

Variant	R@1	R@10	R@100	MRR@100
DOME (full)	**0.686**	**0.880**	**0.927**	**0.740**
Number of pseudo queries per new doc				
Pseudo = 1	0.506	0.760	0.875	0.592
Pseudo = 4	0.615	0.832	0.898	0.678
Pseudo = 7	0.652	0.866	0.918	0.698
DocID type				
BM25-based docIDs	0.683	0.860	0.921	0.722
PQ-based docIDs	0.675	0.877	0.933	0.719
Edited decoder layer range				
Layers 11–15	0.659	0.857	0.904	0.704
Layers 18–22	0.536	0.717	0.836	0.624
soft-to-hard label strategy				
w/o soft label	0.646	0.805	0.901	0.697
w/o hard label	0.325	0.556	0.711	0.436

Effect of pseudo query anount. The number of pseudo queries per document strongly influences adaptation. Using a single query yields poor performance (R@1 = 0.506), while increasing to four and seven queries progressively improves it to 0.652. More queries provide richer contextual cues, enhancing the mapping from semantic representations to docIDs.

Effect of docID assignment. DOME remains robust under different docID schemes. Replacing the default RQ-based identifiers with BM25- or PQ-based docIDs yields comparable performance, indicating that DOME's effectiveness does not rely on a specific docID structure and generalizes across GR systems.

Effect of edited layer range. Performance varies with the decoder layers selected for editing. Editing layers 14–18 achieves the best results, while shifting to earlier (11–15) or later (18–22) layers causes moderate (R@1 = 659) and sharp (R@1 = 0.536) drops, respectively. This confirms that docID mapping knowledge is concentrated in the middle-to-late decoder layers, demonstrating the importance of our layer localization strategy.

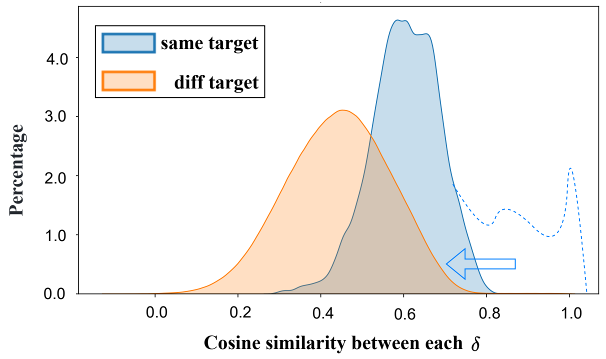


Figure 4: Percentage of pairwise cosine similarity of edit vector δ across GR edit requests on NQ. Yellow denotes pairs with different target docID, blue denotes pairs with the same target docID.

Effect of hybrid-label training strategy. The hybrid-label training is crucial for DOME's success, as removing either the soft or hard component significantly degrades performance. Without soft labels, one-hot supervision causes edit vectors for the same docID to collapse into similar directions. This considerably reduces adaptation flexibility and degrades retrieval performance. Without hard labels, the model lacks precise supervision, leading to severe degradation. As shown in Figure 4, the hybrid strategy reduces excessive similarity among edit vectors sharing the same target docID, yielding more discriminative, query-specific updates that enhance adaptation and retrieval quality.

7 Conclusion

In this work, we have identified that the decoder's failure to learn precise docID mappings is the key obstacle in adapting generative retrieval models to new documents. To address this, we have introduced **DOME**, a GR-specific model-editing framework with a hybrid-label adaptive training strategy that produces discriminative and precise updates to critical decoder layers. Experiments on NQ and MS-MARCO have shown that DOME markedly improves retrieval on new documents, mitigates catastrophic forgetting, and reduces adaptation time by over 40% versus incremental training, establishing it as an efficient and scalable solution.

While promising, our current approach updates only the decoder's FFN layers. In future work, we will focus on further improving the efficiency of the editing process and on exploring adaptation strategies for additional modules, including attention layers and encoder representations. We expect that incorporating more components will improve model adaptability and enhance retrieval performance on both original and new document sets.

Acknowledgments

This research was (partially) supported by the Natural Science Foundation of China (62272274, 62202271, 62472261, 62372275, 62522210, T2293773), the National Key R&D Program of China with grant No. 2024YFC3307300 and No. 2022YFC3303004, the Provincial Key R&D Program of Shandong Province with grant No. 2024CXGC010108, the Natural Science Foundation of Shandong Province with grant No. ZR2024QF203, the Technology Innovation Guidance Program of Shandong Province with grant No. YDZX2024088, 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 agreements No. 101070212 (FINDHR) and No. 101201510 (UNITE).

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

[1] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. 2020. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics* 9, 8 (2020), 1295.

[2] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive Search Engines: Generating Substrings as Document Identifiers. In *NeurIPS*.

[3] Albrecht Böttcher and David Wenzel. 2008. The Frobenius norm and the commutator. *Linear algebra and its applications* 429, 8-9 (2008), 1864–1885.

[4] Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing Factual Knowledge in Language Models. In *EMNLP*. ACL, 6491–6506.

[5] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive Entity Retrieval. In *ICLR*. OpenReview.net.

[6] Siyuan Cheng, Ningyu Zhang, Bozhong Tian, Xi Chen, Qingbin Liu, and Huajun Chen. 2024. Editing language model-based knowledge graph embeddings. In *AAAI*, Vol. 38. 17835–17843.

[7] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge Neurons in Pretrained Transformers. In *ACL*. ACL, 8493–8502.

[8] Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating Factual Knowledge in Pretrained Language Models. In *EMNLP*. ACL, 5937–5947.

[9] Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. AlphaEdit: Null-Space Constrained Knowledge Editing for Language Models. In *ICLR*. OpenReview.net.

[10] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *EMNLP*. ACL, 5484–5495.

[11] 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. *ACM Transactions on Information Systems* (2024).

[12] Yongquan He, Zihan Wang, Peng Zhang, Zhaopeng Tu, and Zhaochun Ren. 2020. VN Network: Embedding Newly Emerging Entities with Virtual Neighbors. In *CIKM*. 505–514.

[13] Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Guojun Ma, Mingyang Wan, Xiang Wang, Xiangnan He, and Tat-seng Chua. 2025. Anyedit: Edit any knowledge encoded in language models. *arXiv preprint arXiv:2502.05628* (2025).

[14] 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 *EMNLP*. ACL, 6769–6781.

[15] Aditi Khandelwal, Harman Singh, Hengrui Gu, Tianlong Chen, and Kaixiong Zhou. 2024. Cross-Lingual Multi-Hop Knowledge Editing. In *EMNLP*. ACL, 11995–12015.

[16] Chaeeun Kim, Soyoung Yoon, Hyunji Lee, Joel Jang, Sohee Yang, and Minjoon Seo. 2024. Exploring the Practicality of Generative Retrieval on Dynamic Corpora. In *EMNLP*. ACL, 13616–13633.

[17] Varsha Kishore, Chao Wan, Justin Lovelace, Yoav Artzi, and Kilian Q. Weinberger. 2023. IncDSI: Incrementally Updatable Document Retrieval. In *ICML*, Vol. 202. PMLR, 17122–17134.

[18] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 452–466.

[19] Sangam Lee, Ryang Heo, SeongKu Kang, Susik Yoon, Jinyoung Yeo, and Dongha Lee. 2024. Why These Documents? Explainable Generative Retrieval with Hierarchical Category Paths. *arXiv preprint arXiv:2411.05572* (2024).

[20] Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023. Multiview Identifiers Enhanced Generative Retrieval. In *ACL*. ACL, 6636–6648.

[21] Yu-An Liu, Ruqing Zhang, Jiafeng Guo, Changjiang Zhou, Maarten de Rijke, and Xueqi Cheng. 2025. On the Robustness of Generative Information Retrieval Models: An Out-of-Distribution Perspective. In *ECIR*, Vol. 15573. Springer, 407–423.

[22] Yougang Lyu, Lingyong Yan, Shuaiqiang Wang, Haibo Shi, Dawei Yin, Pengjie Ren, Zhumin Chen, Maarten de Rijke, and Zhaochun Ren. 2024. KnowTuning: Knowledge-aware Fine-tuning for Large Language Models. In *EMNLP*. 14535–14556.

[23] Yougang Lyu, Lingyong Yan, Zihan Wang, Dawei Yin, Pengjie Ren, Maarten de Rijke, and Zhaochun Ren. 2025. MACPO: Weak-to-Strong Alignment via Multi-Agent Contrastive Preference Optimization. In *ICLR*.

[24] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[25] Vittorio Mazzia, Alessandro Pedrani, Andrea Caciolai, Kay Rottmann, and Davide Bernardi. 2024. A survey on knowledge editing of neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[26] 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 *EMNLP*. ACL, 8198–8213.

[27] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and Editing Factual Associations in GPT. In *NeurIPS*.

[28] Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-Editing Memory in a Transformer. In *ICLR*. OpenReview.net.

[29] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *ICML*. PMLR, 15817–15831.

[30] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. 2019. When does label smoothing help?. In *NeurIPS*. 4696–4705.

[31] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In *COCO@NeurIPS*, Vol. 1773. CEUR-WS.org.

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

[33] Anja Reusch and Yonatan Belinkov. 2025. Reverse-Engineering the Retrieval Process in GenIR Models. In *SIGIR*. ACM, 668–677.

[34] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[35] Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. 2021. Editing a classifier by rewriting its prediction rules. In *NeurIPS*. 23359–23373.

[36] Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. 2023. Learning to Tokenize for Generative Retrieval. In *NeurIPS*.

[37] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NeurIPS*. 3104–3112.

[38] Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Prakash Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer Memory as a Differentiable Search Index. In *NeurIPS*.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*. 5998–6008.

[40] Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Sun, Weiwei Deng, Qi Zhang, and Mao Yang. 2022. A Neural Corpus Indexer for Document Retrieval. In *NeurIPS*.

[41] Ye Wang, Xinrun Xu, Rui Xie, Wenxin Hu, and Wei Ye. 2024. Generative retrieval with large language models. *arXiv preprint arXiv:2402.17010* (2024).

[42] Zihan Wang, Zhaochun Ren, Chunyu He, Peng Zhang, and Yue Hu. 2019. Robust Embedding with Multi-Level Structures for Link Prediction. In *IJCAI*. 5240–5246.

[43] Zihan Wang, Kai Zhao, Yongquan He, Zhumin Chen, Pengjie Ren, Maarten de Rijke, and Zhaochun Ren. 2023. Iteratively Learning Representations for Unseen Entities with Inter-Rule Correlations. In *CIKM*. 2534–2543.

[44] Zihan Wang, Ziqi Zhao, Zhumin Chen, Pengjie Ren, Maarten de Rijke, and Zhaochun Ren. 2023. Generalizing Few-Shot Named Entity Recognizers to Unseen Domains with Type-Related Features. In *EMNLP*. 2228–2240.

[45] Zihan Wang, Yujia Zhou, Yiteng Tu, and Zhicheng Dou. 2023. NOVO: learnable and interpretable document identifiers for model-based IR. In *CIKM*. 2656–2665.

[46] Peipei Xia, Li Zhang, and Fanzhang Li. 2015. Learning similarity with cosine similarity ensemble. *Information sciences* 307 (2015), 39–52.

[47] Yang Xu, Yutai Hou, Wanxiang Che, and Min Zhang. 2022. Language anisotropic cross-lingual model editing. *arXiv preprint arXiv:2205.12677* (2022).

[48] Tianchi Yang, Minghui Song, Zihan Zhang, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. 2023. Auto Search Indexer for End-to-End Document Retrieval. In *EMNLP*. ACL, 6955–6970.

[49] Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. 2024. Scalable and Effective Generative Information Retrieval. In *WWW*. ACM, 1441–1452.

[50] Hansi Zeng, Chen Luo, and Hamed Zamani. 2024. Planning ahead in generative retrieval: Guiding autoregressive generation through simultaneous decoding. In *SIGIR*. ACM, 469–480.

[51] 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 *SIGIR*. ACM, 3325–3334.

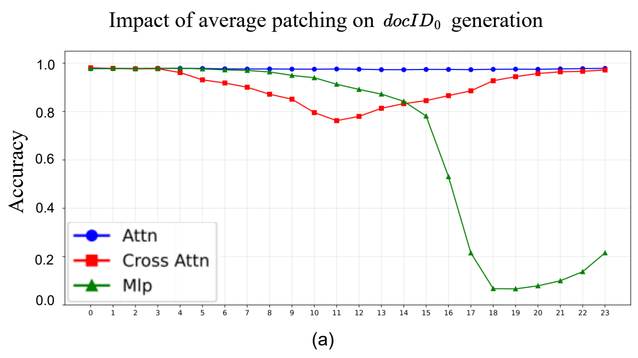
[52] Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can We Edit Factual Knowledge by In-Context Learning?. In *EMNLP*. ACL, 4862–4876.

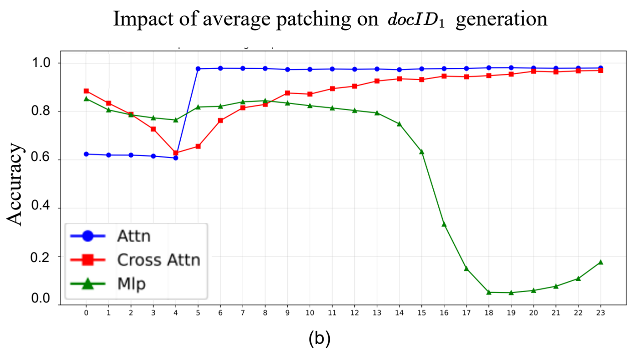
[53] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen. 2022. Ultron: An ultimate retriever on corpus with a model-based indexer. *arXiv preprint arXiv:2208.09257* (2022).

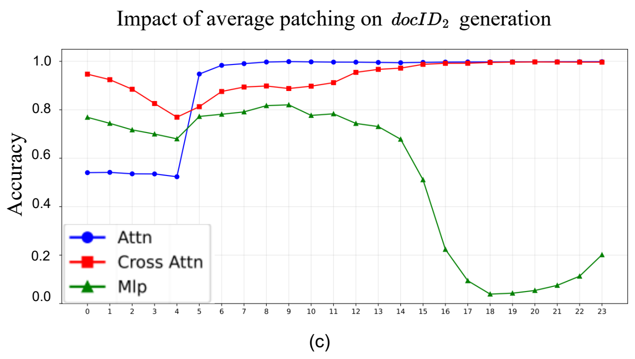
[54] Yu-Jia Zhou, Jing Yao, Zhi-Cheng Dou, Ledell Wu, and Ji-Rong Wen. 2023. Dynamicretriever: A pre-trained model-based ir system without an explicit index. *Machine Intelligence Research* 20, 2 (2023), 276–288.

A Appendix

A.1 Patching for locating critical layers







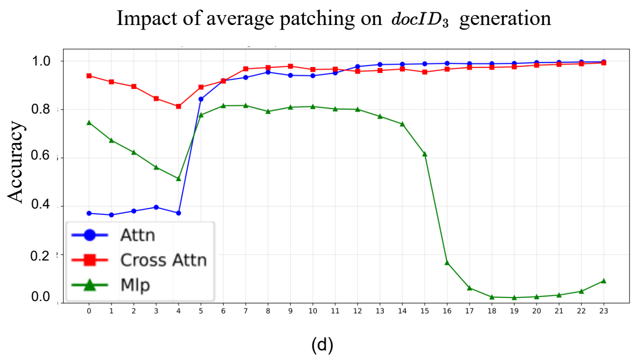


Figure 5: Accuracies of average patching across decoder layers for predicting docID tokens at positions 0–3 on NQ.

To investigate which decoder layers store the critical knowledge for docID mapping, we analyse the impact of replacing intermediate outputs in different modules (FFN, self-attention, cross-attention) during the prediction of each docID token in the generative process. We evaluate on four separate settings, where the target token is located at positions $t = 0$, $t = 1$, $t = 2$, and $t = 3$ in the docID sequence. The horizontal axis in Figure 5 denotes the decoder layer index, and the vertical axis denotes the probability of correctly predicting the target docID at that position.

Patching technique. Following the average patching approach [33], for a specific decoder layer l and module type M (FFN / Self-Attn / Cross-Attn), we first compute the average output representation $\bar{h}_{l,M}$ over the training set. Given an input (query,docID prefix) context x, we replace the original output $h_{l,M}(x)$ of module M in layer l with the average representation:

$$h'_{l,M}(x) = \bar{h}_{l,M}. \tag{12}$$

The patched representation propagates through the remaining layers, and we measure the probability

$$P_{\text{correct}}(t, l, M) = p_\theta \left(y_t^* \mid q, y_{<t}; h_{l,M} \leftarrow \bar{h}_{l,M}\right), \tag{13}$$

where y_t^* denotes the ground-truth docID token at position t. By applying this operation to each layer and module in turn, we obtain the performance curves shown in Figure 5.

Observations. Across all four target positions ($t = 0, 1, 2, 3$), patching FFN layers in the middle and final stages of the decoder leads to the most pronounced drop in P_{correct}, indicating that these layers store critical mapping knowledge from semantic representations to docID tokens. Patching self-attention modules has little to no effect for $t = 0$, while cross-attention patching causes a moderate drop in mid layers. For $t = 1, 2, 3$, both self-attention and cross-attention patching result in a noticeable drop in the early decoder layers. These consistent trends confirm the main finding in Section 5.1: middle and final FFN modules are the most crucial for docID mapping and should be the primary targets for model editing.

A.2 Generalization to varying numbers of documents (RQ4)

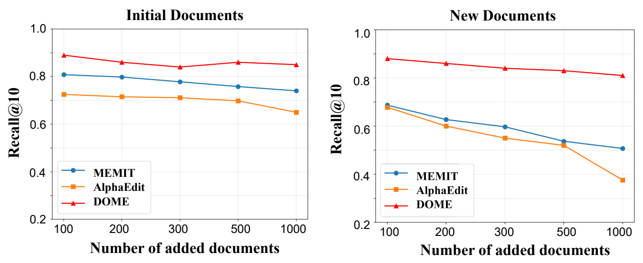


Figure 6: Performance over initial documents (left) and newly added documents (right) with a varying number of new documents on NQ.

To answer RQ4, we analyze DOME's generalization ability by varying the number of newly added documents from 100 to 1,000 and evaluating retrieval performance on both the initial and new collections. The results, illustrated in Figure 6, show that DOME maintains robust performance across different scales of document updates. We draw the following conclusions: (1) DOME effectively preserves existing knowledge. As shown in the left panel of Figure 6, retrieval performance on the initial collection remains highly stable, exhibiting almost no degradation even when 1,000 new documents are integrated. This demonstrates that the constrained update mechanism effectively prevents catastrophic forgetting regardless of update scale. (2) DOME demonstrates robust generalization to newly added documents. As shown in the right panel of Figure 6, performance on new documents remains strong, with only a slight and expected decline as the editing task becomes more complex. These results indicate that DOME can generalize effectively across varying numbers of new documents, integrating updates while ensuring that retrieval performance remains strong.