Dynamic Retriever for In-Context Knowledge Editing via Policy Optimization

Mahmud Wasif Nafee[1,2*] **Maiqi Jiang**[3*] **Haipeng Chen**[3] **Yanfu Zhang**[3†]
[1]Rensselaer Polytechnic Institute, Troy, NY, USA
[2]Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
[3]College of William & Mary, Williamsburg, VA, USA
nafeem@rpi.edu mjiang04@wm.edu hchen23@wm.edu yzhang105@wm.edu

Abstract

Large language models (LLMs) excel at factual recall yet still propagate stale or incorrect knowledge. In-context knowledge editing offers a gradient-free remedy suitable for black-box APIs, but current editors rely on static demonstration sets chosen by surface-level similarity, leading to two persistent obstacles: (*i*) a quantity–quality trade-off, and (*ii*) lack of adaptivity to task difficulty. We address these issues by dynamically selecting *supporting demonstrations* according to their *utility* for the edit. We propose **D**ynamic **R**etriever for **I**n-Context **K**nowledge **E**diting (**DR-IKE**), a lightweight framework that (1) trains a BERT retriever with REINFORCE to rank demonstrations by editing the reward, and (2) employs a *learnable threshold* to prune low-value examples, shortening the prompt when the edit is easy and expanding it when the task is hard. DR-IKE performs editing without modifying model weights, relying solely on forward passes for compatibility with black-box LLMs. On the COUNTERFACT benchmark, it improves edit success by up to 17.1%, reduces latency by 41.6%, and preserves accuracy on unrelated queries, demonstrating scalable and adaptive knowledge editing.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in memorizing vast amounts of knowledge and applying it across various applications, such as question answering (Min et al., 2024), dialogue systems (Feng et al., 2023), and code generation (He et al., 2024). Nevertheless, because these models are trained on a fixed corpus (Touvron et al., 2023; Brown et al., 2020), their knowledge often lags behind real-world developments—either due to erroneous data in the training set (Cao et al., 2021a) or the inherent delay between data collection and model deployment (Dhingra et al., 2022; Gu et al., 2024b). For instance, during a conversation a user might ask, "*What is the newest iPhone right now?*", and a model trained before 2024 could confidently reply, "*The iPhone 15 is the latest model,*" even though Apple introduced the iPhone 16 in September 2024. Such mismatches propagate to end tasks that depend on up-to-date factual grounding, including temporal reasoning (Zhu et al., 2025), fact verification (Mousavi et al., 2024), and personalized recommendation (Bao et al., 2024). Retraining a full-scale LLM with fresh data would in principle close this gap, but the computational and financial costs are prohibitive (DeepSeek-AI et al., 2024); as a result, methods for **Knowledge Editing**—which aim to surgically modify the original model in order to incorporate new or corrected facts (Mitchell et al., 2022a)—have emerged as an attractive alternative due to their efficiency and targeted updates.

However, existing practical knowledge editing methods still face three major obstacles. (1) *Computation cost*. Gradient-based editors such as ROME (Meng et al., 2022), MEND (Mitchell et al., 2022a), MEMIT (Meng et al., 2023a), and SERAC (Cao et al., 2021b) first calculate the gradient with respect to the parameters that most influence the prediction, and then apply a low-rank update to the implicated weight matrices, so that the model produces the desired answer while leaving other behaviours unchanged. Although one edit on GPT-J-6B takes only seconds (Meng et al., 2023a), buffering intermediate activations for T5-11B already consumes ∼60 GB of GPU memory (Mitchell et al., 2022a), and both memory and compute scale linearly with model size and the number of edits, rendering thousands of real-time updates on 70-B models infeasible for most labs or start-ups. Moreover, these algorithms assume full read–write access to the weights, an assumption violated by the growing number of commercial LLM APIs. Consequently,

[*] Equal contribution.
[†] Corresponding author.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 16756–16769
November 4-9, 2025

in-context knowledge editing (IKE) (Zheng et al., 2023) has been proposed: it injects the corrected facts as demonstrations in the prompt, thereby achieving competitive success rates without any parameter updates or gradient computations. Because most production LLMs are exposed only through black-box APIs, we confine our study to this realistic black-box-only scenario and ask how far a purely in-context approach can be pushed.

Even for in-context-based editors (Madaan et al., 2022; Zheng et al., 2023; Zhong et al., 2023), two further challenges remain. (1) *Trade-off between example quantity and quality*. A series of recent in-context-learning studies reveals that accuracy rises sharply when the prompt includes the first few highly relevant examples, but the marginal benefit of each additional example soon saturates and may even reverse once redundancy or noise creeps in (Chen et al., 2023; Agarwal et al., 2024; Zhang et al., 2025; Liu et al., 2023). In knowledge editing, this fragility is amplified: irrelevant, conflicting, or poorly ordered supporting facts not only fail to help but also propagate errors to otherwise unrelated predictions (Yu et al., 2025), producing "ripple" side-effects (Cohen et al., 2024). Consequently, practical editors must retrieve and present a minimal set of maximally informative, non-overlapping demonstrations rather than indiscriminately lengthening the prompt.

(2) *Adaptivity to task difficulty*. Not all edits are equally tractable: recent work shows that success rates vary by knowledge type. Edits involving abstract categories (Ge et al., 2024a), commonsense reasoning (Wu et al., 2024), temporal references (Ge et al., 2024b), reasoning-heavy logic (Hua et al., 2024), or popular entities (Cohen et al., 2024) are consistently more error-prone. In addition, editing multiple related facts increases the risk of contradiction or unintended side-effects (Li et al., 2024). These findings highlight the need for adjust the editing methods based on the structure and complexity of each task.

To address the above challenges, we propose **D**ynamic **R**etriever for **I**n-Context **K**nowledge **E**diting (**DR-IKE**), an adaptive example retrieval framework optimized via policy gradients. Our method augments a frozen LLM with a trainable BERT-based retriever that selects and ranks auxiliary factual examples to best facilitate accurate edits. Rather than relying on heuristic retrieval or static prompts, the retriever learns to balance informativeness and factual consistency through interaction: it is rewarded for example selections that lead the LLM to produce the correct, updated output. This approach mitigates the cost and rigidity of gradient-based editors, while also adapting to the difficulty of individual edits through a learnable thresholding mechanism that filters harmful or redundant context. In doing so, our framework offers a scalable, model-agnostic alternative for real-time factual updates, especially in settings with limited access to model internals.

Our main contributions are as follows:

- We design a BERT-based retriever trained with policy gradients that selects and ranks auxiliary facts *without touching model weights*. This light-weight, inference-time solution directly avoids the memory and compute overheads for gradient-based editors, making it viable in commercial API settings.
- By learning to surface only the most informative, non-overlapping examples, our method curbs prompt length and minimises redundancy, resolving the *quantity–quality trade-off* identified in in-context knowledge editing.
- We introduce a dynamic threshold that tightens or relaxes retrieval based on the predicted hardness of each edit, furnishing extra support for abstract, temporal, or popular-entity updates while avoiding over-prompting on easier cases—thereby addressing the need for *adaptivity to task difficulty*.
- On the COUNTERFACT benchmark, DR-IKE outperforms earlier in-context editors: it trims prompt length, cutting per-epoch latency by 41.6 % ; raises Edit-Success Rate by up to 17.1 % with fewer tokens; and, thanks to its learnable budget controller, delivers the largest gains on paraphrased and other difficult edits while preserving unrelated knowledge, demonstrating true *adaptivity to task difficulty*.

2 Related Work

2.1 In-context Learning for Knowledge Editing

Early work on knowledge editing relied on *gradient-based* parameter updates. Dai et al. (2022) modify FFN key–value pairs in KNOWLEDGE NEURON; Meng et al. (2023b) apply constrained least squares to the FFN matrix in ROME;

and Mitchell et al. (2022a) learn low-rank updates in MEND. Although effective, these methods are computationally heavy, can harm unrelated behaviour (Gu et al., 2024b), and are infeasible for commercial black-box LLMs (Zheng et al., 2023). This limitation has sparked interest in *in-context knowledge editing*, which injects the corrected facts as demonstrations in the prompt. Simple prompt engineering—e.g., prefixing with "*Imagine that ...*" (Cohen et al., 2024)—or chain-of-thought prompting in EditCoT (Wang et al., 2024a) can inject new facts without touching weights. Retrieval-augmented loops exploit past errors and user feedback: SERAC reroutes matched inputs to a counterfactual model (Madaan et al., 2022; Mitchell et al., 2022b), and multi-hop question formulations improve fidelity (Gu et al., 2024a; Zhong et al., 2023). Demonstration-diversity strategies—first shown by Si et al. (2023) and formalised in IKE via k-NN retrieval of COPY, UPDATE, and RETAIN examples (Zheng et al., 2023)—boost success rates further. Yet existing retrieval pipelines rely solely on embedding similarity, leaving room for *utility-aware* selection and pruning. There remains a need for dynamic methods that operate entirely under frozen weights, avoid backpropagation, and minimize both training and inference overhead.

2.2 Retrieval for In-context Learning

Retrieval for ICL traditionally uses static methods such as BM25 (Robertson and Zaragoza, 2009) or dense embeddings (Liu et al., 2022), which lack task-aware filtering. Fine-tuned retrievers (Lu et al., 2022; Li et al., 2023; Wang et al., 2024b) adapt to specific domains but still return a fixed number of demonstrations. Yu et al. (2025) model retrieval as a Markov Decision Process with a learnable threshold for long-tail QA. While their bag-of-words preselection suffices for that setting, it lacks the context sensitivity needed for knowledge editing. We adapt this framework by using embeddings-based preselection for richer semantics and editing-specific rewards to encourage efficacy, specificity, and stability.

3 Problem Statement

Definition 3.1 (Knowledge Editing)**.** Let $\mathcal{M}$ be a frozen large-language model and let $\mathcal{K}_c$ denote a single factual triple (or small set of triples) stored in the model's parametric memory. Knowledge editing seeks a post-edited model $\mathcal{M}' = f(\mathcal{M}, \mathcal{K}_c \to \mathcal{K}'_c)$ such that

1. for any query q whose answer depends solely on $\mathcal{K}_c$, the response of $\mathcal{M}'$ is consistent with the revised fact $\mathcal{K}'_c$, and
2. for any query that depends on unrelated knowledge $\mathcal{K}_s$ ($\mathcal{K}_s \cap \mathcal{K}_c = \varnothing$), the behavior of $\mathcal{M}'$ matches that of the original model $\mathcal{M}$.

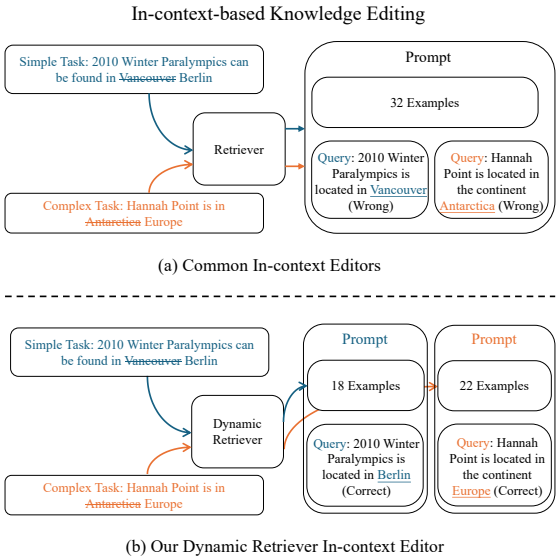

Figure 1: Motivating example for adaptive in-context editing. (a) A *fixed-shot* editor appends the same large set of demonstrations to every query; the oversized prompt diffuses model attention, hindering both a straightforward location swap (*Winter Paralympics*: Vancouver → Berlin) and a more concept-heavy update (*Hannah Point*: Antarctica → Europe), whose corpus evidence is dominated by Antarctic descriptions. (b) Our *dynamic* editor adjusts prompt length to task complexity, retaining a minimal context for the simple edit while retrieving additional, targeted facts for the harder one.

Definition 3.2 (In-context Knowledge Editing)**.** A language model may be modified either by changing its parameters or by altering the input prompt P. In-context Knowledge Editing keeps the parameters of $\mathcal{M}$ frozen and employs a retriever R to construct an augmented In-Context Learning (ICL) prompt $P^\star = P + \langle d_1, \ldots, d_m \rangle$, where d_j are natural-language *demonstrations*. Conventional editors choose a *fixed* number $m \geq 0$ of demonstrations (Zheng et al., 2023; Cohen et al., 2024). We generalize this paradigm by learning a threshold σ that allows the retriever to select a *variable* number $k_\sigma(q) = \big|\{j \mid \pi_j \geq \sigma\}\big|$ of demonstrations, where π_j is the retriever score of candidate j; see Fig. 1(b) for an illustration.

Definition 3.3 (Demonstration Categories (Zheng et al., 2023))**.** Each retrieved example is assigned one of three functional roles:

- **Copy**—explicitly restates the target fact to reinforce $\mathcal{K}'_c$.
  Example: "*The official language of Brazil is* $\rightarrow$ Spanish."

- **Update**—paraphrases the query before introducing the new fact.
  Example: "*Brazil's national language has been changed to* $\rightarrow$ Spanish."

- **Retain**—references a related context that *should not* change.
  Example: "*The official language of Canada is* $\rightarrow$ English."

While RETAIN examples increase specificity, an excess of low-utility RETAINS can bloat the prompt and push the LLM to fall back on outdated parametric memory. Therefore, our method dynamically selects only a subset of the RETAIN pool, while all three categories remain available for prompt construction.

4 Method

We now give a brief tour of **DR-IKE** (see Fig. 2 for a schematic). During training, a lightweight BERT retriever learns, via REINFORCE (Williams, 1992), to rank RETAIN candidates, while a learnable threshold σ simultaneously adjusts the number of examples admitted to the prompt. At inference time the same threshold truncates the ranked list once, producing a single compact prompt that is sent to the black-box LLM.

The rest of this section is organized as follows. Section 4.1 formalizes the retrieval process as a Markov Decision Process, specifying the state, action, policy, and reward. Section 4.2 details the policy-gradient training protocol. Finally, Section 4.3 introduces the dynamic budget controller that adapts σ to prevent prompt bloat.

4.1 Markov-Decision-Process Formulation

We cast RETAIN-example selection as an (Markov Decision Process) MDP $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$ so that the retriever can optimize for *utility* rather than raw similarity.

State. At step t the state is $s_t = (x, \mathcal{R}_t)$, where x is the query and $\mathcal{R}_t = \{e_1, \ldots, e_t\} \subseteq \mathcal{C}$ is the ordered set of RETAINS chosen so far. The candidate pool $\mathcal{C} = \{e^{(1)}, \ldots, e^{(n)}\}$ is produced off-line by KNN retrieval over sentence embeddings.

Action. The action space is $\mathcal{A} = \mathcal{C} \cup \{\textit{stop}\}$. Choosing $a_t = e^{(i)}$ appends that example to the growing prompt; choosing *stop* finalizes the prompt and terminates the episode.

State Transition. If $a_t \neq \textit{stop}$ the transition is deterministic:

$$s_{t+1} = (x, \mathcal{R}_t \cup \{a_t\}). \tag{1}$$

No successor state is generated after *stop*.

Reward. After each action a_t, the current prompt is fed to the frozen LLM and a step-wise reward is issued:

$$r_{t+1} \;=\; 2 \times \mathbf{1}\big[\hat{y}_{t+1} = y_{\text{new}}\big] - 1, \tag{2}$$

where $\hat{y}_{t+1}$ is the LLM's answer and $\mathbf{1}[\cdot]$ is the indicator. This step-wise feedback guides the retriever in estimating the marginal impact of each added example on editing success.

4.2 Policy-gradient Training of the Retriever

Fig. 2 (left) outlines the learning loop. At each epoch we optimize the retriever parameters while the LLM remains frozen.

Demonstration selection. For each edit instance (x, y_{new}), we use a pretrained 20M-parameter Sentence-Transformer (Reimers and Gurevych, 2020) to retrieve fixed COPY and UPDATE demonstrations, and to preselect a *candidate pool* for RETAIN examples.

Policy and action sampling. The retriever is a frozen 4-layer BERT encoder (29 M parameters) (Turc et al., 2020) followed by a trainable linear head S_θ whose parameters are denoted by θ. Each $e^{(i)} \in \mathcal{C}$ receives a score $z_i = S_\theta(x, e^{(i)})$. A softmax yields the categorical policy:

$$\pi_\theta(a_t = e^{(i)} \mid s) = \frac{\exp(z_i)}{\sum_{j=1}^{n} \exp(z_j)}. \tag{3}$$

At step t an index a_t is sampled from π_θ and the corresponding retain e_t is appended to the prompt *iff* $\pi_\theta(a_t \mid s_t) > \sigma$; otherwise the threshold mechanism (Section 4.3) emits the *stop* action and the episode terminates.

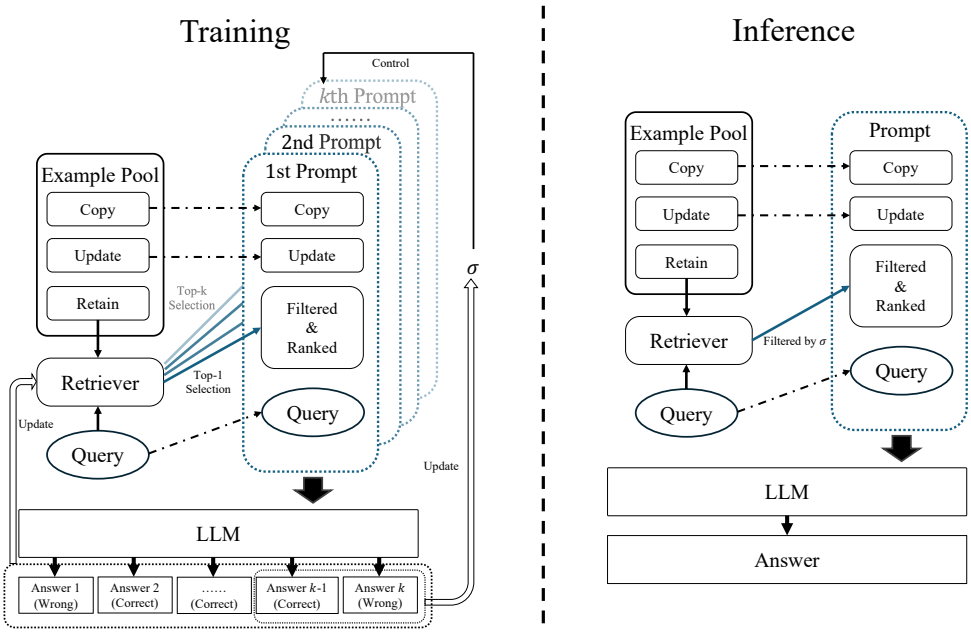

Figure 2: Overall architecture of the proposed framework. **Training stage (left).** Given an edit query, the retriever scores a pool of retained candidate demonstrations. A learnable threshold σ converts these scores into an integer k, the number of prompt variants to generate. The frozen LLM processes each variant; its outputs yield the reinforcement signal used to update the retriever via policy gradients. Whenever the aggregated outcome flips between success and failure, σ is adjusted to capture the revised difficulty estimate. **Inference stage (right).** At test time, the learned σ truncates the ranked list once, yielding a single concise prompt to the frozen LLM.

Policy Update. After each action the current prompt is queried and a binary reward $r_t \in \{+1, -1\}$ is observed Eq. 2. The REINFORCE (Williams, 1992) loss is

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} r_t \log \pi_\theta(a_t \mid s_t), \qquad (4)$$

where T is the (variable) episode length. Because gradients flow only through the linear head, updates are computationally light while still teaching the retriever *which* retains to keep and *when* to stop. A complete, line-by-line pseudocode listing appears in Appendix A.

4.3 Dynamic Budget Controller

To curb prompt bloat we endow the retriever with a *learnable threshold* σ called budget controller that caps the number of RETAIN examples. At the start of training we set $\sigma = 0$, guaranteeing that at least one RETAIN can be selected. During both training and inference a candidate $e^{(i)}$ is kept only if its policy probability satisfies $\pi_\theta(a_t = e^{(i)} \mid s) > \sigma$; candidates are considered in descending order of probability.

Adaptive update. While constructing the prompt we monitor the binary reward $r \in \{+1, -1\}$ after each newly added example. If appending the $(j{+}1)$-th RETAIN turns a previously correct answer into an incorrect one, we tighten the budget by raising σ to the largest probability among the *remaining* candidates:

$$\sigma \leftarrow \max\Big(\sigma,\ \max_{i>j} \pi_\theta(a_t = e^{(i)} \mid s)\Big). \qquad (5)$$

Thus the bar for inclusion becomes progressively higher on difficult edits, whereas easy edits naturally terminate after a few high-utility examples. This single scalar threshold allows the system to learn *both* which RETAIN matter and *how many* are worth keeping, achieving compact yet effective prompts.

5 Experiments

5.1 Experimental Setup

Dataset We use the COUNTERFACT benchmark (Meng et al., 2023b), a widely adopted evaluation suite for factual knowledge editing in language models. It comprises 21,919 factual records. We follow Zheng et al. (2023) in using the first 2,000 records for the editable sample pool and the remainder for constructing ICL demonstrations.

Editing Method	Extra Params	S ↑	ESR ↑	PC ↑	RR ↑	ESM ↑	GSM ↑	Inference Time(s) ↓
Llama-3.1-8B-Instruct								
FactPrompt	0	0.434	0.61	0.34	0.43	0.21	-0.05	**1.99**
EditCoT	0	0.431	0.70	0.33	0.40	0.24	-0.06	2.02
IKE	20M	0.727	0.76	0.67	**0.76**	0.43	0.39	6.52
DR-IKE (Ours)	49M	**0.775**	**0.89**	**0.81**	0.66	**0.69**	**0.49**	3.81
Qwen 2.5-7B								
FactPrompt	0	0.335	0.48	0.26	0.33	0.54	0.45	2.42
EditCoT	0	0.424	0.53	0.43	0.35	-0.09	-0.15	**2.25**
IKE	20M	0.738	0.75	0.69	**0.78**	0.64	0.57	6.75
DR-IKE (Ours)	49M	**0.779**	**0.89**	**0.77**	0.70	**0.83**	**0.74**	4.21

Table 1: Editing performance across methods for two base models, including extra parameter counts, harmonic mean S, edit/generalization success margins (ESM/GSM), and measured inference time per iteration. Best scores are in **bold**, second-best are underlined.

Language Model We use Meta-Llama-3.1-8B-Instruct and Meta-Llama-3.2-3B-Instruct (Grattafiori et al., 2024), Mistral-7B-Instruct-v0.2 (Jiang et al., 2023), and both sizes of Qwen 2.5 (7B and 1.5B) (Yang et al., 2024) via HuggingFace's pipeline API. The model parameters are not updated at any point.

Training Configuration We train on 300 randomly selected samples and evaluate on 100, using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 1×10^{-4} for 5 epochs. Each training episode corresponds to a query, with batch size 1 for step-wise updates guided by policy-based optimization.

Compute Environment All training was conducted on Google Colab using an NVIDIA L4 GPU (24 GB VRAM), optimized for inference and lightweight training. We implement our pipeline in PyTorch and HuggingFace Transformers v4.39.3.

Baselines We compare our method with three representative in-context editing strategies: **FactPrompt** (Cohen et al., 2024), which steers the model by prepending a narrative prefix ("Imagine that…"); **EditCoT** (Wang et al., 2024a), which applies chain-of-thought prompting to guide step-by-step reasoning before editing; and **IKE** (Zheng et al., 2023), which iteratively selects COPY, UPDATE, and RETAIN examples to inject new facts while preserving existing knowledge.

Evaluation Metrics We evaluate editing performance using standard metrics introduced by Zheng et al. (2023). ***Edit Success Rate (ESR)*** measures how often the LLM outputs the correct edited object (`target_new`); ***Retention Rate (RR)*** measures whether the original object (`target_true`) is preserved on unrelated neighborhood prompts; and ***Paraphrase Consistency (PC)*** checks agreement on paraphrased in-scope prompts. We report their harmonic mean as ***Score (S)*** following Meng et al. (2023b). We also include ***Edit Success Magnitude (ESM)*** and ***Generalization Success Magnitude (GSM)***, which quantify log-probability shifts between `target_new` and `target_true` on edited and paraphrased prompts, respectively. ESR is tracked epoch-wise to monitor retriever learning.

5.2 Main Results

Tab. 1 shows the performance of knowledge editing in different methods using two different LLMs, Llama 3.1 Instruct-8b and Qwen 2.5-7b.

As expected, all methods yield comparable ESR scores, but diverge sharply on PC and RR. FactPrompt and EditCoT, for example, perform reasonably on ESR (0.61 and 0.70) but falter on PC and RR due to their limited prompt design. Without UPDATE or RETAIN examples, the LLM lacks guidance on when to generalize or preserve original knowledge. The IKE method improves on this by incorporating COPY, UPDATE, and RETAIN examples, enabling more structured control over generalization and specificity. However, its inclusion of all RETAINS can lead to noisy prompts that hinder effective editing.

Our method addresses this by using a retriever trained via policy-based optimization to rank RETAINS and a budget controller to keep only the most

helpful ones. As a result, it shows noticeable improvement over ESR and PC while remaining competitive on RR across both LLMs. We also observe consistent gains in *ESM* and *GSM*—which measure the model's confidence in the edited and generalized responses, respectively—indicating that DR-IKE not only produces correct outputs but does so with higher certainty.

Additionally, comparing the last column, our method matches or outperforms IKE across all but one metric while achieving lower inference time, thanks to the budget controller's adaptive prompt shortening.

Method	ESR ↑	PC ↑	RR ↑
Llama-3.1-8B-Instruct			
IKE-All	0.76	0.67	0.76
Rank-All	0.81	0.69	0.76
Rank-50%	0.86	0.71	**0.78**
DR-IKE	**0.89**	**0.81**	0.66
Qwen-2.5-7B-Instruct			
IKE-All	0.75	0.69	0.78
Rank-All	0.84	0.67	0.78
Rank-50%	0.84	0.76	**0.79**
DR-IKE	**0.89**	**0.77**	0.70

Table 2: Ablation of RETAIN-example selection: IKE-All (baseline), static ranking (Rank-All, Rank-50%), and our dynamic controller (DR-IKE).

5.3 Ablation Study of the Retain-Example Budget Controller

Tab. 2 contrasts four RETAIN-selection strategies. **IKE-All** uses all examples without filtering. **Rank-All** ranks and keeps all, i.e. no pruning of redundant RETAIN examples. **Rank-50%** ranks and keeps only the top half, adding a static budget. **DR-IKE** uses our dynamic controller to prune low-utility examples on the fly, yielding the best balance of efficiency and efficacy.

Rank-All yields performance comparable to **IKE-All**, indicating that ordering alone is insufficient to improve editing outcomes. Introducing a static budget with **Rank-50%** which ranks and keeps only the top half of RETAIN examples enhances ESR, PC, and RR, confirming that an overabundance of redundant demonstrations can degrade efficacy, generalization, and specificity. Finally, **DR-IKE** uses our dynamic controller to prune low-utility examples on the fly, further boosting ESR and PC by adaptively selecting the most informative RETAINS; the modest drop in RR suggests that overly stringent pruning may occasionally discard useful context. Overall, dynamic example budgeting via DR-IKE proves essential for achieving the optimal balance of efficiency and effectiveness in knowledge editing. For empirical statistics on the number of RETAIN examples selected per prompt across 100 test instances, refer to Appendix B.

Effects on Training Efficiency Introducing our dynamic budget controller substantially reduces training time. By pruning lower-value RETAIN, we shorten prompts and cut LLM inference overhead. For Llama 3.1-8B, per-epoch runtime drops from 2080.1s to 1459.6s (29.8% saving), and for Qwen 2.5-7B from 1459.6s to 1174.6s (19.5%). This efficiency gain supports scalable deployment on larger models.

5.4 Case Study

To intuitively show the effectiveness of our proposed method, we show two knowledge editing case studies from the Counterfact dataset in Fig. 3. The edited fact, COPY and UPDATE examples remain the same for both methods. In IKE (blue columns), RETAIN examples are fixed, whereas in our method (green columns), they are dynamically selected using the retriever and budget controller. In the first case study, IKE's prompt construction correctly guides the LLM on the original query but fails when the prompt is paraphrased, whereas our method succeeds on both. This difference arises because our retriever identifies RETAIN examples whose contextual cues closely align with the factual update, regardless of surface variation, ensuring that even paraphrased queries receive the appropriate support. In the second case study, IKE fails on both the original and paraphrased queries—likely because its RETAIN examples mix language-based rather than location-based contexts—while our method retrieves only location-based RETAINS, resulting in correct answers in both cases and confirming the suitability of our example selection.

5.5 Base Model Performance

Editing performance often hinges on a model's capacity to integrate new information while preserving existing knowledge. Tab. 3 shows that smaller LLMs, e.g. **Llama 3.2 (3B)**, can match larger counterparts in ESR but suffer noticeable

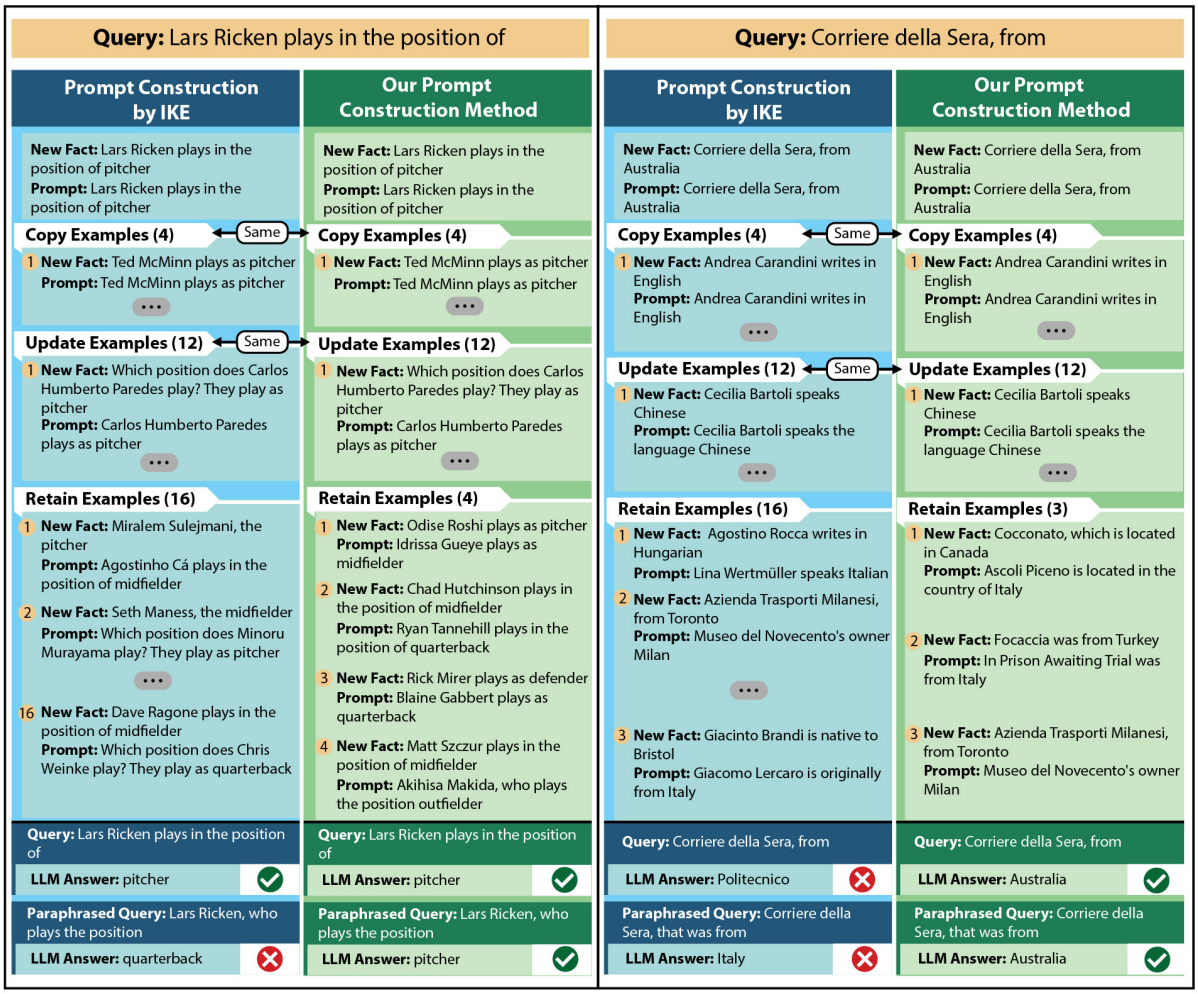

Figure 3: Case study illustrating prompt construction and example selection for challenging queries under IKE versus our method.

Model (Parameters)	ESR ↑	PC ↑	RR ↑
Llama 3.1 (8B)	**0.89**	**0.81**	0.66
Llama 3.2 (3B)	0.86	0.71	0.61
Mistral v0.2 (7B)	0.53	0.42	0.34
Qwen 2.5 (7B)	**0.89**	0.77	**0.70**
Qwen 2.5 (1.5B)	0.57	0.42	0.39
SmolLM2 (1.7B)	0.46	0.27	0.22

Table 3: Performance comparison of various LLMs.

drops in PC and RR, a pattern echoed by the tiny **Qwen 1.5B** and **SmolLM2 1.7B**. This underscores the role of scale in furnishing the nuanced representations needed for precise, stable edits. **Mistral v0.2 (7B)** underperforms across all metrics, likely because its limited 8k context window restricts the in-context learning needed for effective editing (Xu et al., 2024). Moreover, instruct-fine-tuned models like **Llama 3.1 (8B)** and **Qwen 2.5 (7B)** demonstrate particular resilience to paraphrased prompts, more reliably retaining injected facts under syntactic variation than their smaller or non-instruct counterparts.

5.6 DR-IKE under Black-Box API Settings

We additionally tested DR-IKE under black-box conditions using the Gemini-2.0-Flash API on Kaggle. As shown in Table 4, DR-IKE achieved substantial gains in Edit Success Rate (ESR) and Paraphrase Consistency (PC) compared to the in-context editing baseline (IKE). The improvements are not only considerable in absolute terms, but they also surpass the margins we observed on earlier experiments with smaller open-weight models such as LLAMA-3.1-8B or Mistral-7B. This suggests that larger LLMs may benefit even more from the dynamic retrieval strategy employed by DR-IKE. At the same time, we note a decrease in Retain Rate (RR), indicating a trade-off between successful ed-

its, generalization, and preservation of unrelated knowledge. Future work could explore techniques to further stabilize RR without compromising the strong gains in ESR and PC.

Method	ESR ↑	PC ↑	RR ↑
IKE	0.69	0.52	**0.57**
DR-IKE	**0.91**	**0.83**	0.46

Table 4: Black-box evaluation of DR-IKE against the in-context editing baseline (IKE) using the Gemini-2.0-Flash API on Kaggle.

5.7 Extended Evaluation on Benchmark Datasets

We further evaluated DR-IKE on two additional benchmarks: zsRE (Meng et al., 2023b) and WikiDataCounterFact (Zhong et al., 2023). Both datasets include a broader range of edit types and formats. The same experimental setup as in the main paper was used, with the LLAMA-3.1-8B model. Notably, the average similarity to k-nearest neighbors is substantially lower in zsRE (0.4042) and WikiDataCF (0.4507) than in CounterFact (0.5695), indicating that these datasets contain more diverse and less redundant examples. This demonstrates that DR-IKE maintains strong performance even in lower-similarity, low-resource settings.

As shown in Table 5, DR-IKE achieves consistently higher Edit Success Rate (ESR) and Paraphrase Consistency (PC) compared to the in-context editing baseline (IKE). Retain Rate (RR) is also reported alongside ESR and PC, providing a more complete picture of the trade-off between successful edits, generalization, and preservation of unrelated knowledge.

Method	ESR ↑	PC ↑	RR ↑
zsRE			
IKE	0.30	0.22	0.49
DR-IKE	**0.33**	**0.26**	**0.51**
WikiDataCounterFact			
IKE	0.39	0.40	0.63
DR-IKE	**0.42**	**0.43**	**0.64**

Table 5: Extended evaluation of DR-IKE on zsRE and WikiDataCounterFact.

6 Conclusion

In this work, to improve upon existing demonstration strategies and prompting paradigms for knowledge editing, we propose Dynamic Retriever for In-Context Knowledge Editing (DR-IKE), which uses a pre-trained BERT-based retriever (trained via policy gradient) to rank examples by their editing utility and a budget controller to prune lower-value cases from the prompt. In particular, we examine RETAIN examples that, while safeguarding auxiliary context, may compromise edit efficacy; our retriever instead prioritizes those that genuinely bolster both fact injection and knowledge preservation, as confirmed by our empirical results. DR-IKE's combination of learned ranking and adaptive budgeting yields slightly higher edit success and consistency than leading in-context editing methods, while also reducing inference and training time. These results suggest that DR-IKE can scale effectively to very large, black-box LMs. Future work could extend DR-IKE to better handle fact-type variation, low-overlap retrieval settings, and complex edits by incorporating fact-aware retrieval and more flexible retrieval sources.

Limitations

There are several limitations of our work. First, our evaluation is constrained by the COUNTERFACT benchmark, which does not categorize facts by type (e.g. historical, numerical, geographical, technical). As a result, we cannot assess how editing performance varies across different knowledge domains. Furthermore, DR-IKE relies on the presence of sufficiently similar paraphrases and neighborhood examples within a single dataset. In scenarios where such examples are sparse or absent, retrieval quality and editing efficacy may degrade, limiting the method's applicability in low-resource or highly specialized domains. Finally, although our budget controller mitigates prompt-length concerns, LLM context windows remain finite. In cases requiring a large number of demonstrations—such as nuanced multi-step edits or extensive domain coverage—dynamic budgeting alone may not suffice to fit all necessary examples within the model's maximum input length.

Acknowledgement

This work is supported in part by the National Science Foundation (NSF) grant IIS-2451436 and

Commonwealth Cyber Initiative grant HC-4Q24-059.

References

Rishabh Agarwal, Avi Singh, Lei M Zhang, Bernd Bohnet, Luis Rosias, Stephanie C.Y. Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, John D Co-Reyes, Eric Chu, Feryal Behbahani, Aleksandra Faust, and Hugo Larochelle. 2024. Many-shot in-context learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Keqin Bao, Ming Yang, Yang Zhang, Jizhi Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2024. Realtime personalization for llm-based recommendation with customized in-context learning. *ArXiv*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*.

Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. 2021a. Knowledgeable or educated guess? revisiting language models as knowledge bases. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021b. Editing factual knowledge in language models. In *Conference on Empirical Methods in Natural Language Processing*.

Jiuhai Chen, Lichang Chen, Chen Zhu, and Tianyi Zhou. 2023. How many demonstrations do you need for in-context learning? In *Conference on Empirical Methods in Natural Language Processing*.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bing-Li Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 179 others. 2024. Deepseek-v3 technical report. *ArXiv*.

Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, pages 257–273.

Yujie Feng, Zexin Lu, Bo Liu, Li-Ming Zhan, and Xiao-Ming Wu. 2023. Towards llm-driven dialogue state tracking. In *Conference on Empirical Methods in Natural Language Processing*.

Huaizhi Ge, Frank Rudzicz, and Zining Zhu. 2024a. How well can knowledge edit methods edit perplexing knowledge? *ArXiv*.

Xiou Ge, Ali Mousavi, Edouard Grave, Armand Joulin, Kun Qian, Benjamin Han, Mostafa Arefiyan, and Yunyao Li. 2024b. Time sensitive knowledge editing through efficient finetuning. In *Annual Meeting of the Association for Computational Linguistics*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and *et al.* 2024. The llama 3 herd of models. *CoRR*.

Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. 2024a. PokeMQA: Programmable knowledge editing for multi-hop question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024b. Model editing harms general abilities of large language models: Regularization to the rescue. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Runyu He, Anyu Ying, and Xiaoyu Hu. 2024. Improving opendevin: Boosting code generation llm through advanced memory management. *Applied and Computational Engineering*.

Wenyue Hua, Jiang Guo, Mingwen Dong, He Zhu, Patrick Ng, and Zhiguo Wang. 2024. Propagation and pitfalls: Reasoning-based assessment of knowledge editing through counterfactual tasks. In *Annual Meeting of the Association for Computational Linguistics*.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L'elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *ArXiv*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023. Unified demonstration retriever for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. 2024. Unveiling the pitfalls of knowledge editing for large language models. In *The Twelfth International Conference on Learning Representations*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*.

Linyong Lu, Qingxiu Zhang, Xiang Li, and Jingjing Liu. 2022. Promptpg: Prompting for policy gradient training. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Neural Information Processing Systems*.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023a. Mass editing memory in a transformer. *The Eleventh International Conference on Learning Representations (ICLR)*.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2023b. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Dehai Min, Nan Hu, Rihui Jin, Nuo Lin, Jiaoyan Chen, Yongrui Chen, Yu Li, Guilin Qi, Yun Li, Nijun Li, and Qianren Wang. 2024. Exploring the impact of table-to-text methods on augmenting llm-based question answering with domain hybrid data. In *North American Chapter of the Association for Computational Linguistics*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022a. Fast model editing at scale. In *International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022b. Memory-based model editing at scale. In *International Conference on Machine Learning*.

Seyed Mahed Mousavi, Simone Alghisi, and Giuseppe Riccardi. 2024. Dyknow: Dynamically verifying time-sensitive factual knowledge in llms. In *Conference on Empirical Methods in Natural Language Processing*.

Nils Reimers and Iryna Gurevych. 2020. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Lee Boyd-Graber, and Lijuan Wang. 2023. Prompting GPT-3 to be reliable. In *The Eleventh International Conference on Learning Representations*.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Niko lay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cris tian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2020. Well-read students learn better: On the importance of pre-training compact models.

Changyue Wang, Weihang Su, Qingyao Ai, and Yiqun Liu. 2024a. Knowledge editing through chain-of-thought. *ArXiv*.

Liang Wang, Nan Yang, and Furu Wei. 2024b. Learning to retrieve in-context examples for large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.

Xiaobao Wu, Liangming Pan, William Yang Wang, and Anh Tuan Luu. 2024. Akew: Assessing knowledge editing in the wild. In *Conference on Empirical Methods in Natural Language Processing*.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Retrieval meets long context large language models. In *The Twelfth International Conference on Learning Representations*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 2024. Qwen2 technical report. *CoRR*.

Shuyang Yu, Runxue Bao, Parminder Bhatia, Taha Kass-Hout, Jiayu Zhou, and Cao Xiao. 2025. Dynamic uncertainty ranking: Enhancing retrieval-augmented in-context learning for long-tail knowledge in LLMs. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.

Xiaoqing Zhang, Ang Lv, Yuhan Liu, Flood Sung, Wei Liu, Shuo Shang, Xiuying Chen, and Rui Yan. 2025. More is not always better? enhancing many-shot in-context learning with differentiated and reweighting objectives. *arXiv preprint arXiv:2501.04070*.

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 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Chenghao Zhu, Nuo Chen, Yufei Gao, Yunyi Zhang, Prayag Tiwari, and Benyou Wang. 2025. Is your LLM outdated? a deep look at temporal generalization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*.

A Appendix: Training Procedure for the Dynamic Retriever

The overall training protocol for our dynamic retriever is illustrated in Algorithm 1. At the beginning of each epoch, we initialize the retriever's parameters θ and the budget threshold σ. For every training instance (x, y_{new}) in the dataset, we first construct the fixed COPY and UPDATE demonstrations based on semantic similarity using embeddings from a 20M-parameter Sentence Transformer (Reimers and Gurevych, 2020). These fixed components serve as the foundational part of every in-context prompt. Next, we select a candidate pool $\mathcal{C}$ of RETAIN examples—again using the same embedding space—to identify potentially helpful facts related to the input.

Once the RETAIN pool is constructed, the retriever's scoring function S_θ evaluates each candidate, producing scalar relevance scores which are normalized via softmax to obtain a probability distribution p over the pool. The retriever then determines how many examples to select, based on the current value of σ, by choosing the top candidates whose probabilities exceed the threshold. If none of the probabilities surpass σ, we ensure that at least one RETAIN is selected to prevent empty prompt construction. This mechanism enables the retriever to modulate prompt length dynamically, balancing retrieval confidence with context limitations.

We then proceed to construct the full in-context prompt incrementally by adding one RETAIN example at a time (top-j based on p). After each

Algorithm 1 Training Protocol for Dynamic Example Selection

Require: initial parameters θ_0; training set $\mathcal{T}$; max shots K; learning rate α
Ensure: trained retriever θ and budget threshold σ

$\theta \leftarrow \theta_0, \quad \sigma \leftarrow 0$
for epoch $= 1$ **to** N_{epochs} **do**
  for each $(x,\ y_{\text{new}}) \in \mathcal{T}$ **do**
    Fixed COPY / UPDATE demos for x
    $\mathcal{C} \leftarrow$ RETAIN candidates (pre-selected via embeddings)
    $z \leftarrow S_\theta(x, \mathcal{C})$ ▷ scalar scores
    $p \leftarrow \mathrm{softmax}(z)$ ▷ policy distribution
    $k \leftarrow \min\big(K,\ |\{i \mid p_i > \sigma\}|\big)$
    if $k = 0$ **then**
      $k \leftarrow 1$
    end if
    prev_r $\leftarrow \perp, \quad \mathcal{L} \leftarrow 0$
    for $j = 1$ **to** k **do**
      $\mathcal{R}_j \leftarrow$ top-j RETAINS from $\mathcal{C}$ by p
      Prompt with COPY, UPDATE, $\mathcal{R}_j$
      query LLM $\rightarrow \hat{y}$
      $r \leftarrow \begin{cases} +1, & \hat{y} = y_{\text{new}} \\ -1, & \text{otherwise} \end{cases}$
      $c \leftarrow (j > 1) \wedge (\textit{prev\_r} = +1) \wedge (r = -1)$
      if c **then**
        $\sigma \leftarrow \max\big(\sigma,\ \max_{i>j} p_i\big)$
      end if
      $\mathcal{L} \mathrel{+}= -r\ \log p_j$
      *prev*_r $\leftarrow r$
    end for
    $\theta \leftarrow \theta - \alpha\, \nabla_\theta \mathcal{L}$ ▷ REINFORCE
  end for
end for

addition, we query the language model and check if its output matches the desired edited response y_{new}. A binary reward $r \in \{+1, -1\}$ is assigned depending on whether the answer is correct. If adding a particular RETAIN causes a correct prediction to flip to incorrect, we treat it as a degradation in prompt quality and increase the threshold σ to exclude lower-probability examples in future steps. This allows the model to prune less useful RETAINS early and adapt the prompt construction strategy over time.

Throughout the episode, we accumulate REINFORCE-style loss (Williams, 1992) using the reward signal and the log-probability of each selected RETAIN. The resulting gradient is used to update only the retriever's scoring head via policy gradient. By repeating this process across training epochs, the retriever learns not just which RETAIN examples are most helpful for factual editing, but also how to dynamically adjust its selection budget to optimize both model performance and context efficiency.

B Appendix: Retain Budget Distribution Across Models

Our proposed framework, **DR-IKE**, dynamically selects the number of RETAIN examples per prompt based on their estimated contribution to edit success, rather than applying a fixed number across all edits. Fig. 4 and Fig. 5 illustrate how the number of RETAIN examples varies per edit instance for two different LLMs. In contrast to IKE, which statically includes 16 RETAINS regardless of their informativeness or utility, DR-IKE tailors the prompt length to the needs of each specific edit.

This adaptivity yields significant efficiency gains. On average, LLaMA 3.1-8B requires only $\mathbf{3.02 \pm 0.96}$ RETAIN examples per prompt, while Qwen 2.5-7B uses $\mathbf{3.72 \pm 1.15}$. These compact prompts reduce both the computational overhead and the risk of introducing irrelevant or distracting context. Notably, despite using fewer examples, DR-IKE maintains competitive or even superior edit success compared to IKE. This highlights the importance of selective inclusion: by pruning redundant or low-impact examples, DR-IKE not only saves prompt space but also enhances factual precision and generalization. The results underscore the utility of adaptive demonstration selection as a scalable solution for black-box knowledge editing.

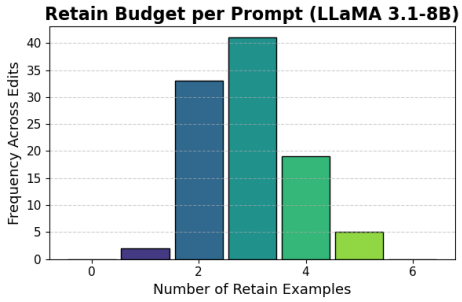

Figure 4: RETAIN budget distribution for LLaMA 3.1-8B under DR-IKE. Each bar shows the frequency of a given RETAIN count per prompt. Mean = $\mathbf{3.02}$, Std = $\mathbf{0.96}$.

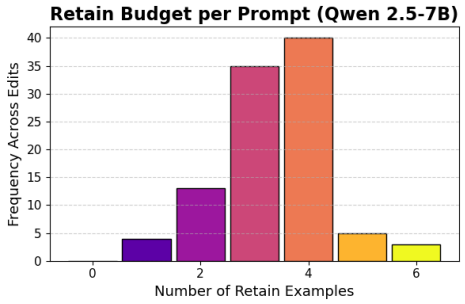

Figure 5: RETAIN budget distribution for Qwen 2.5-7B under DR-IKE. The model allocates more RETAIN examples for edits requiring stronger contextual support. Mean = $\mathbf{3.72}$, Std = $\mathbf{1.15}$.

C Appendix: Analysis of Budget Controller Update Strategies

We experimented with several alternative strategies for updating the budget controller, including semantic relevance modeling using BERT-based encoders, statistical modeling of the softmax score distribution, and hybrid approaches combining both. However, none of these alternatives consistently outperformed our current dynamic threshold update rule. In practice, we observed that the retriever dynamically adjusts its output probabilities over time. As σ increases, the retriever tends to assign higher scores to borderline examples, leading to a natural stabilization of the threshold. This feedback mechanism allows σ to converge without the need for complex or hand-crafted update rules.

Table 6 summarizes the different budget controller architectures we tested using the LLAMA-3.2-3B model, under the same train–test split as in our previous experiments. The **current method (baseline)** relies on a dynamic threshold update rule based on the retriever's evolving score distribution. The **MLP on softmax scores** variant uses

Architecture	ESR ↑	PC ↑
Current method (baseline)	**0.86**	0.71
MLP on softmax scores	0.81	**0.75**
MLP on softmax + query features	0.81	0.69
BERT + MLP (predict σ)	0.83	0.65
BERT (query + context) + MLP	0.80	0.67

Table 6: Comparison of alternative budget controller update strategies. The current method achieves the best overall balance of Edit Success Rate (ESR) and Paraphrase Consistency (PC), while other methods show improvements in isolated metrics but fail to consistently outperform the baseline.

a lightweight feed-forward network trained directly on the softmax probability distribution, while **MLP on softmax + query features** extends this by incorporating query-level features. In the **BERT + MLP (predict σ)** approach, the query is first encoded with a BERT encoder and the resulting representation is passed to an MLP to directly predict the threshold σ. Finally, **BERT (query + context) + MLP** encodes both the query and the in-context examples with BERT before passing them to the MLP for threshold prediction.

Overall, while some alternatives improve on a single dimension (e.g., higher PC with softmax-based MLPs), none consistently match the balanced performance of the baseline approach. This confirms the effectiveness of our dynamic threshold update rule, though confidence-based calibration remains a promising direction for future refinement.

D Appendix: Sensitivity of Reward Function

We tested the robustness of DR-IKE to input ordering and reward timing. Random shuffling or reordering of context examples left model outputs stable in the majority of cases. We also introduced random delays in the reward signal during training, and observed that Edit Success Rate (ESR) remained effectively unchanged, as summarized in Table 7. These results indicate that the framework trains robustly even when the LLM's binary reward signal is noisy or unstable.

Sigma update method	ESR ↑
No delay	0.7567
Randomly delayed	0.7533

Table 7: Sensitivity analysis of the reward function. Edit Success Rate (ESR) remains stable even with randomly delayed reward signals.

E Appendix: Additional Evaluation on Temporal and Numerical Facts

Beyond the main benchmarks, we also conducted preliminary evaluation of DR-IKE on two additional settings. First, on a temporal-edit dataset provided by (Zheng et al., 2023), DR-IKE achieved a small improvement in Edit Success Rate (ESR) compared to the in-context baseline (IKE). Paraphrase Consistency (PC) could not be assessed in this case, as no paraphrased prompts were available in the dataset. Second, we extracted a small subset of numerical examples from the COUNTERFACT dataset. On this subset, DR-IKE demonstrated consistently stronger performance in both ESR and PC relative to IKE. Since there is currently no dedicated dataset focused exclusively on numerical edits, and the few numerical cases in existing benchmarks are insufficient for a systematic study, we defer a more thorough investigation of numerical editing to future work.

Method	ESR ↑	PC ↑
Temporal Editing		
IKE	0.46	–
DR-IKE	**0.50**	–
Numerical Subset of CounterFact		
IKE	0.67	0.53
DR-IKE	**0.78**	**0.68**

Table 8: Preliminary evaluation of DR-IKE on temporal and numerical edits. Paraphrase Consistency (PC) could not be computed for the temporal dataset due to lack of paraphrases.