MoPE: A Massive Mixture of Passage-Level Experts for Knowledge Editing in Open-Domain Question Answering

Anonymous ACL submission

Abstract

As world knowledge continues to evolve, adapting LLMs to new knowledge is crucial, however, it poses significant challenges, as naively fine-tuning the entire model often leads to catastrophic forgetting and high computational costs. While RAG and model editing have been increasingly studied for knowledge adaptation, this paper moves beyond the 'RAG vs. fine-tuning' discussion to explore the '*RAG vs. model editing*' issue, and propose a "Massive Mixture of Experts (MMoE)" approach for model editing, referred to as **MoPE**, i.e., Massive Mixture of Passage-Level Experts, which consists of the key components at training and inference stages: (1) **Massive passage-level editing with MMoE**, where a large set of passage-level experts is created using automatically generated question-answer pairs for each passage, and (2) **Retrieval-based routing with MMoE**, which employs dense retrieval to select the top-k passage-level experts without requiring additional training. Experimental results demonstrate that MoPE outperforms a naively designed variant of RAG, i.e., direct RAG, and when combined with direct RAG, it surpasses an advanced variant of RAG, significantly improving over LoRA-based parameter-efficient tuning methods. Our data and code will be available at `https://github.com/XXX/XXX`.

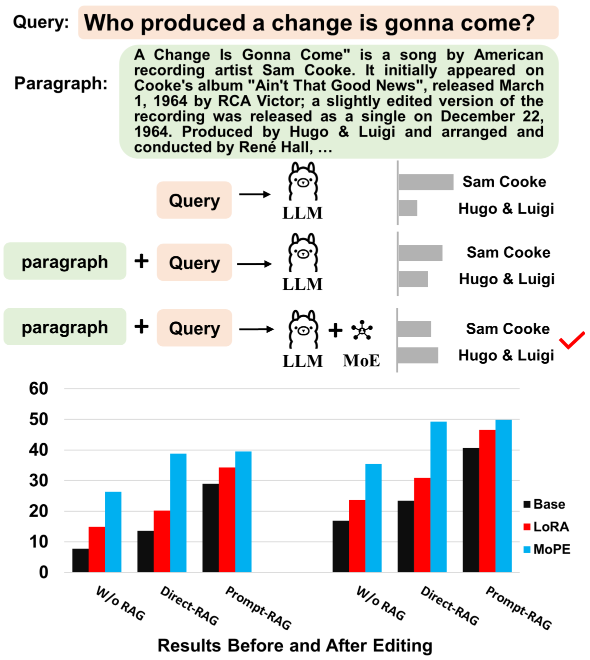

Figure 1: An illustration of context-aware attention: mitigating content focus limitations in RAG through paragraph editing, and a brief overview of the main results of MoPE comparing to the base model with our without RAG, on the NQ dataset. **Left:** EM metric, **Right:** F1 Score. MoPE outperforms direct RAG, which relies on a naively designed prompt for retrieval. When combined with direct RAG, MoPE surpasses prompt-RAG, which employs a more refined prompt variant.

1 Introduction

Large language models (LLMs) have shown remarkable performances on various natural language processing (NLP) tasks, particularly effectively performing knowledge-intensive tasks by their enormously large pretrained knowledge (Zhao et al., 2023; Hadi et al., 2023). Given that world knowledge is continually updated, adapting LLMs to new information and knowledge is crucial, however, naively finetuning the entire model of LLMs meets the key challenging problems: 1) catastrophic forgetting, which may lead to disruption of the old knowledge and performance graduation on the pre-acquired tasks, and 2) nontrivial updating costs due to the large scale of parameters, which requires high computation and memory resource required for updating new knowledge. Previous adaptation approaches can be broadly categorized into two main methods: (1) **Retrieval-Augmented Generation (RAG)** (Lewis et al., 2020; Pan et al., 2024), which leverages the in-context learning (ICL) capabilities of large language models (LLMs) by augmenting input prompts with "retrieved" passages; and (2) **Knowledge Injection**, which can be further divided into *parameter-efficient tuning*(Hu et al., 2021) – which adjusts a small set of trainable parameters while keeping the original ones intact – and *model editing* (De Cao et al., 2021; Meng et al., 2022a), which aims to balance the incorporation of new knowledge updates while preserving existing knowledge.

Among these approaches, unlike RAG and parameter-efficient tuning, which have demonstrated improvements across various NLP tasks, Model editing has been explored primarily in standard 'editing'-specific tasks, rather than in practical knowledge-intensive NLP tasks, leaving its usefulness and impact in real-world applications unclear. Going beyond traditional editing-specific tasks, we aim to explore model editing within open-domain QA as a representative knowledge-intensive NLP task to assess its value and impact in achieving improvements over RAG, as open-domain QA has been widely used. Related to this issue, existing works on fine-tuning methods have explored the debate of 'RAG vs. fine-tuning' (Ovadia et al., 2023; Alghisi et al., 2024; Gupta et al., 2024), showing that parameter-efficient tuning and fine-tuning generally underperform compared to RAG, and their combination with RAG has been found to be advantageous, etc. Advancing beyond previous works, we extensively explore "model editing," rather than fine-tuning or parameter-efficient tuning, so addressing '*RAG vs. model editing*', leading to our key research question: "*Are specific model editing methods effective in achieving improvements over RAG on standard open-domain QA tasks, compared to results obtained using fine-tuning or parameter-efficient tuning?*"

Unlike editing-specific tasks, applying model editing to open-domain QA involves handling a set of *passage-level edits* within a collection, thereby requiring the editor to effectively inject the required passages into LLMs, treating each passage as an edit request. To address this passage-level editing problem, inspired by the remarkable adaptability of the mixture of experts (MoE) on knowledge-intensive and editing tasks (Wang and Li, 2024b,a), we propose the use of a massive MoE (MMoE) for model editing to inject all required passages into a set of experts, referred to as MoPE, i.e., massive mixture of passage-level experts, by assigning a dedicated passage-level expert to each individual passage and integrating these passage-level experts into LLMs, using the "retrieval"-based router, without modifying the original parameters. MoPE consists of the key components for editing and inference stages, as follows:

1. **Massive passage-level editing with MMoE**. In the editing stage, all target passages are considered as massive edit requests. Similar to most MoE approaches that utilize feed-forward network (FFN) layers (Fedus et al., 2022; Du et al., 2022), we augment an FFN layer in the Transformer with an additional expertized FFN module, referred to as a passage-level expert, specialized for each individual passage; Given a passage, we first generate a set of question-answer pairs using a simple prompting method, and then train the passage-level expert based on the loss function used in the machine reading comprehension (MRC) task. During the editing stage, it should be noted that each passage-level expert is trained individually, i.e., only a single passage-level expert is augmented with the original FFN layers in the Transformer, without incorporating all other experts. As a result, we will have a massive mixture of N static passage-level experts, given N number of passages in a collection.

2. **Retrieval-based Router with MMoE**. In the inference stage, since we have passage-level experts, routing mechanism is effectively managed through retrieval. We extensively utilize dense retrieval to design a router, ensuring that only the top-k experts corresponding to the top-k retrieved passages are selected to form a sparse MoE layer, resulting in consisting $k+1$ FFNs at a specific layer. Thus, no separate additional training is required for designing a router.

Experimental results on standard open-domain QA datasets – natural question (NQ) (Kwiatkowski et al., 2019) and Trivia QA (Joshi et al., 2017) show that MoPE improves the baseline performance of RAG and the results using the LoRA-based parameter efficient tuning, as briefly shown in Figure 1: MoPE outperforms "direct-RAG," which relies on a naively designed prompt for retrieval; MoPE combined with direct-RAG surpasses prompt-RAG, which utilizes a more refined prompt variant. For details on the input template, see Appendix A.

Our main contributions are as follows: 1) We investigate the 'RAG vs. model editing' paradigm, expanding beyond previous studies that focused on

'RAG vs. fine-tuning' (Alghisi et al., 2024). 2) We propose MoPE, a novel approach for model editing in open-domain QA. 3) We offer new experimental findings, showing that MoPE outperforms LoRA-based tuning and direct RAG, and the combination of MoPE and direct RAG surpasses even the more advanced variant of RAG, i.e., prompt RAG.

2 Related Work

2.1 Retrieval Augmented Generation

In open-domain question answering, Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) methods effectively enhance the accuracy of model responses by incorporating external knowledge. Recent studies have made significant progress in retrieval quality control and timing optimization. Shi et al. (2023) and Dong et al. (2024) emphasized the importance of enhancing the relevance of retrieved content to user query and the role of knowledge alignment in this process. Lin et al. (2023) proposed an approach that optimizes the query encoder by combining supervised and unsupervised tasks. Additionally, the Fusion-in-Decoder (FiD) method (Izacard and Grave, 2020; Hofstätter et al., 2023) significantly improved generation performance by integrating information from multiple documents during the decoding stage.

In structural model optimization, Self-RAG (Asai et al., 2023) enhances retrieval accuracy and robustness through fine-grained self-reflection, while Yan et al. (2024) and Jiang et al. (2023b) introduced an error-correction strategy to handle inaccurate information. SAIL (Luo et al., 2023) improves instruction tracking in complex contexts by integrating internal and external search engines.

For post-generation output enhancement, KNN-LM (Khandelwal et al., 2019) and RETOMATON (Alon et al., 2022) significantly improve generation quality through nearest-neighbor memory retrieval and weighted finite-state machines. Furthermore, end-to-end training (Lewis et al., 2020; Guu et al., 2020) has emerged as a crucial direction to optimize RAG architectures, allowing more efficient knowledge integration and response generation by minimizing manual intervention and iteratively refining the model.

2.2 Knowledge Injection

Knowledge injection has emerged as a crucial technique for enhancing the performance of language models by efficiently integrating external knowledge. Two primary approaches have gained significant attention: parameter-efficient tuning and model editing.

Parameter-efficient tuning enhances large pre-trained models by introducing small trainable modules while keeping most parameters frozen. Key approaches include adapter (Houlsby et al., 2019), which adds neural network bottlenecks to transformer blocks; Prompt Tuning (Li and Liang, 2021), which optimizes the appended prompts for task adaptation; and LoRA (Hu et al., 2021), which updates rank decomposition matrices. Recent advances, such as DyLoRA (Valipour et al., 2022), improve efficiency by selectively updating partial parameters. Building on DyLoRA, MELO (Yu et al., 2024) introduces a neuron-indexed dynamic LoRA mechanism.

Model editing focuses on maintaining the reliability of edited knowledge, ensuring that the changes successfully address the target queries. Additionally, it emphasizes enhancing the generality, allowing the edited model to generalize the new knowledge to related queries effectively. Furthermore, it seeks to preserve locality, ensuring that the modifications do not interfere with the retention of unrelated original knowledge. It is categorized into three types: Meta-learning editors (De Cao et al., 2021; Mitchell et al., 2021; Tan et al., 2023), which use hyper-networks to adjust gradients; Locate-then-edit editors (Meng et al., 2022a,b; Li et al., 2024), which identify and update relevant parameters; and Memory-based editors, where (Zheng et al., 2023; Zhong et al., 2023; Gu et al., 2023; Cheng et al., 2023) update knowledge from prompts using in-context learning without gradient updates or parameter modifications, where other approaches like T-Patcher (Huang et al.) and MEMoE (Wang and Li, 2024b) store the memory of edited facts using additional parameters.

2.3 Mixture of Experts

In transformer-based Large Language Models (LLMs), Mixture-of-Experts (MoE) layers utilize a set of expert networks and a gating mechanism to route inputs to the most suitable experts (Shazeer et al., 2017; Antoniak et al., 2023). These layers are strategically positioned after the self-attention sublayer to optimize feed-forward network (FFN) selection, significantly reducing computational overhead in large models like PaLM (Chowdhery et al., 2023), where FFN layers account for the majority of parameters.

Dense MoE approaches activate all available experts simultaneously, which enhances predictive accuracy but demands substantial computational resources. Early implementations (Jacobs et al., 1991; Rasmussen and Ghahramani, 2001; Aljundi et al., 2017) demonstrated this effectiveness, and more recent methods like EvoMoE (Nie et al., 2021), MoLE (Wu et al., 2024), LoRAMoE (Dou et al., 2023), and DSMoE (Pan et al., 2024) have refined the dense MoE structure to balance performance and efficiency.

Sparse MoE improves computational efficiency by selecting only the top-k experts for each input, thereby maintaining accuracy while reducing processing demands (Shazeer et al., 2017). However, this selective activation can cause load imbalances, where certain experts are overused while others are underutilized. To counter this, auxiliary loss functions are introduced to distribute tokens more evenly across experts, as seen in (Lepikhin et al., 2020; Jiang et al., 2024; Du et al., 2022; Fedus et al., 2022). This strategy allows sparse MoE models to scale effectively by expanding parameter capacity without a corresponding increase in computational cost.

3 Methodology

Figure 2 presents an overview of MoPE, highlighting its training and inference stages: (1) *Passage-level editing with MMoE* and (2) Inference using a *retrieval-based router with MMoE*. In the passage-level editing stage, a set of question-answer (QA) pairs is automatically generated for each passage using named entity recognition (NER)-based answer extraction and a simple prompting method. Each passage-level expert is trained individually using the generated QA pairs, following an MRC-like objective function. Given N passages, we construct N corresponding passage-level experts. During inference, given these N passage-level experts and a test question, we first perform retrieval-based routing. Specifically, we use dense retrieval with a reranking method to identify the top-k retrieved passages. The corresponding k passage-level experts are then selected from the N experts and integrated with the original FFN layer. This selected sparse MoE is used to generate an answer to the test question.

3.1 Passage-level Editing with MMoE

Suppose that there is N number of passages in a collection, formulated as $\mathcal{C} = \{p_i\}_{i=1}^{N}$ where $\mathcal{C}$ is a collection, and p_i is i-th passage [1]. The passage-level editing process creates N passage-level experts, comprising two key steps: (1) the *construction of QA pairs* and (2) the *training passage-level experts*, as detailed below.

3.1.1 Construction of QA pairs

Given a passage, we create its set of QA pairs by using the automatic data augmentation method (Yu et al., 2022). To this end, we first separately train a question generator G_θ based on the T5-xl model to learn how to generate questions from textual segments, based on an additional training dataset $D_{QG} = \{(p_{\psi(j)}, q_j, a_j)\}_{j=1}^{M}$, where $p_{\psi(j)}$ is the $\psi(j)$-th passage in the training set[2], q_j is the j-th question for $p_{\psi(j)}$, a_j is the j-answer for q_j on $p_{\psi(j)}$. It is important to note that D_{train} is separate from the test dataset in the final evaluation. In our case, we use only the training samples from the NQ dataset. We also have the instruction template function $\mathcal{T}$ is defined as follows: *"Generate a question with <answer> as '{a}' from the <context>: '{p}'."*. Thus, $\mathcal{T}(p, a)$ indicates the prompted input that takes an answer a for a passage p. The generator G_θ is trained by minimizing the negative log-likelihood (NLL) loss as follows:

$$\mathcal{L}(\theta) = -\sum_{(p,q,a)\in\mathcal{D}_{QG}} logP(q|\mathcal{T}(p,a);\theta) \quad (1)$$

where $P(y|x;\theta)$ represents the conditional probability of generating question y given input x.

Once G_θ is trained, given a i-th passage $p_i \in \mathcal{C}$, we first extract named entities in p_i an answers using a NER method, and generate corresponding questions to the extracted answers. The resulting set of QA pairs for p_i is denoted as $\mathcal{D}_{p_i}$. The generation process please refer to Appendix B.

3.1.2 Training Passage-Level Experts

Once $\mathcal{D}_{p_i}$ is obtained, the editing step uses $\mathcal{D}_{p_i}$ as training set to create the corresponding passage-level expert E_i under the MoE framework. To

[1]While we need to create experts for all passages in a collection, evaluation only requires the experts corresponding to the top-retrieved passages for test questions. Therefore, instead of generating a full set of experts, we construct only the minimum necessary number of experts, ensuring that the subset is sufficient for evaluation.

[2]Here, we use ψ as an existential Skolem-like function that maps the j-th training example to its corresponding passage index, for notational convenience.

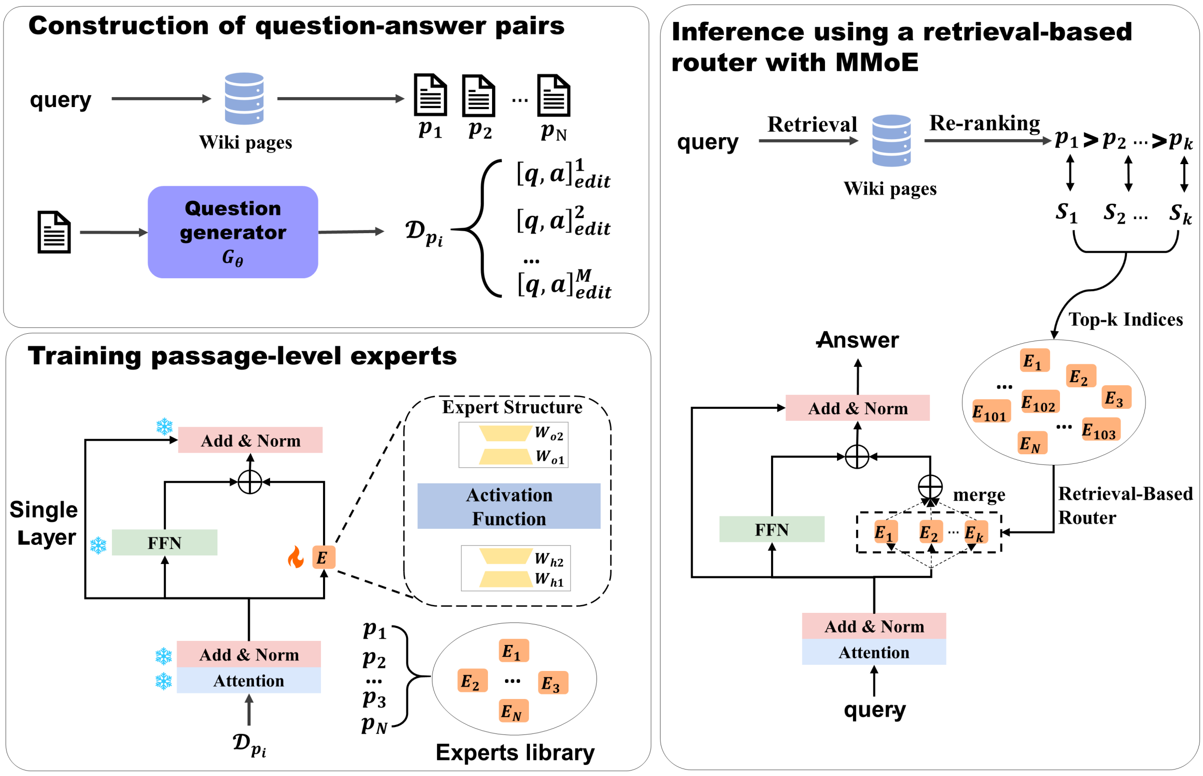

Figure 2: An overview of the proposed MoPE framework: 1) Passage-Level Editing with MMoE: For a given passage, its QA pairs are automatically constructed using the fine-tuned question generator (see Section 3.1.1). The passage-specific expert is then trained within the MoE layer using these QA pairs. 2) Inference with Retrieval-Based Router: MMoE is deployed, where dense retrieval and a fine-tuned reranker are applied to select the top-retrieved relevant passages (see Section 3.2.1), and the corresponding passage-level experts are then incorporated to form a sparse MoE. The relevance vector of the router is computed based on either the top-1 or top-k passages, using Eq. (5) and Eq. (12), respectively.

formally describe the MoE setting, suppose that at each token position t, $\mathbf{x}_t^l \in \mathbb{R}^{d_m}$ is given input at layer l. The original FNN block, denoted as $\mathrm{FFN}^\ell(\mathbf{x}_t^\ell)$, is formulated as follows:

$$\mathrm{FFN}^\ell(\mathbf{x}_t^\ell) = \mathrm{relu}\left(\mathbf{x}_t^l \mathbf{W}_K^l\right) \mathbf{W}_V^l \qquad (2)$$

where $\mathbf{W}_K^l \in \mathbb{R}^{d_m \times d}$ and $\mathbf{W}_V^l \in \mathbb{R}^{d \times d_m}$ represent the key and value projection matrices of the original FNN layer, respectively. For a i-th passage, we now have the passage-level expert, $\mathrm{E}_i^\ell(\mathbf{x}_t^\ell)$ is formulated, based on another FNN block, as follows:

$$\mathrm{E}_i^\ell(\mathbf{x}_t^\ell) = \mathrm{relu}\left(\mathbf{x}_t^l \mathbf{W}_{i,K}^l\right) \mathbf{W}_{i,V}^l \qquad (3)$$

where $\mathbf{W}_{i,K}^l \in \mathbb{R}^{d_m \times d}$ and $\mathbf{W}_{I,V}l \in \mathbb{R}^{d \times d_m}$ represent the key and value projection matrices of the expert FNN layer for i-th passage, respectively. Given N passages, we have N passage-level experts, thereby forming the following MMoE:

$$\mathbf{y}_t^l \quad = \quad \mathrm{FFN}^\ell(\mathbf{x}_t^\ell) + \lambda_l^t \sum_{j=1}^{N} \mathbf{r}[j] \cdot \mathrm{E}_j^\ell(\mathbf{x}_t^\ell) \quad (4)$$

where λ_l^t is the mixing parameter, which is set to 1 for a specific layer l and 0 otherwise depending on token index t, and $\mathbf{r} \in \mathbb{R}^N$ is the "relevance" vector to a given query q, obtained by the "retrieval-based router," which consists of N relevance values, defined as follows:

$$\mathbf{r}[j] = \begin{cases} 1, & \text{if } p_j \text{ in top-}k(q, \mathcal{C}) \\ 0, & \text{else} \end{cases} \qquad (5)$$

where top-$k(q, \mathcal{C})$ is the set of the top-k retrieved passages, in general. During the editing, when $q \in \mathcal{D}_{p_i}$, top-$k(q, \mathcal{C}) = \{p_i\}$, because it is clear that p_i is the gold relevant passage for q. Thus, during editing, given $\mathcal{D}_{p_i}$, Eq. (4) is simplified into:

$$\mathbf{y}_t^l \quad = \quad \mathrm{FFN}^\ell(\mathbf{x}_t^\ell) + \lambda_l^t \mathrm{E}_i^\ell(\mathbf{x}_t^\ell) \qquad (6)$$

Under Eq. (6), given $\mathcal{D}_{p_i}$, we use an autoregressive generative loss function to train the i-th passage-level expert E_i, independently from other experts.

Furthermore, to effectively maintain parameters of E_i, instead of using the full parameters of FNN as in Eq. (3), we perform low-rank matrix factorization on its standard linear transformation layers to reduce computational overhead and parameter redundancy, thereby decomposing the fully connected weight matrices into low-rank components.

$$\mathrm{E}_i^{\ell}(\mathbf{x}_t^{\ell}) = \mathrm{relu}\left(\mathbf{x}_t^l \mathbf{W}_{i,K2}^l \mathbf{W}_{i,K1}^l\right) \mathbf{W}_{i,V1}^l \mathbf{W}_{i,V2}^l \quad (7)$$

Where $\mathbf{W}_{i,K2}^l \in \mathbb{R}^{d_m \times k}$, $\mathbf{W}_{i,K1}^l \in \mathbb{R}^{k \times d_m}$, $\mathbf{W}_{I,V1}^l \in \mathbb{R}^{d_m \times k}$ and $\mathbf{W}_{I,V}^l \in \mathbb{R}^{k \times d_m}$ represent the corresponding low-rank key and value projection matrices of the i-th passage-level expert.

3.2 Inference with a Retrieval-Based Router with MMoE

Once MMoE is trained during the passage-level editing stage, inference begins by retrieving relevant passages for a given test query, based on dense retrieval, followed by a reranking step to refine the results. Then, the retrieval-based router is applied to determine the relevance vector $\mathbf{r}$ in Eq. (4), based on similarity scores obtained from retrieval. This process depends on two different variants of using MMoE: 1) *Single-expert*: Uses only the top-1 expert, where the relevance vector remains the same as in Eq. (5) with $k = 1$ used during the editing stage. 2) *Multi-expert*: Uses the top-k passage-level experts, where the relevance vector is a soft version computed using a softmax-based probability distribution after top-k truncation.

3.2.1 Retrieval-Reranker

Given a query q, DPR uses both the query vector $\mathsf{Emd}(q)$ and the passage vector $\mathsf{Emd}(p)_i$ for p_i, and computes inner product similarity between them:

$$score(q, p_i) = \mathsf{Emd}(q)^{\top} \mathsf{Emd}(p)_i \quad (8)$$

For the reranking, we incorporate *BAAI/bge-reranker-v2-gemma*[3] (Xiao et al., 2024) as a reranker model and fine-tune it to improve passage relevance. For the finetuning the reranker, we additionally construct a training dataset passages retrieved by DPR. To optimize the reranker, we adopt a contrastive loss function (Sohn, 2016), ensuring that relevant passages receive higher scores than irrelevant ones. For a given question q in the training set, let p_+ be a positive passage, and let $(p_-^{(1)}, \ldots, p_-^{(L)})$ be L negative passages for q. The loss function is defined as:

$$\mathcal{L}_{reranker} = -log \frac{exp\left(score_{rerank}\left(q, p_+\right)\right)}{\sum_{i=1}^{L+1} exp\left(score_{rerank}(q, p^{(i)})\right)} \quad (9)$$

The re-ranker results can be found in Appendix C.

3.2.2 Retrieval-Based Router

The retrieval-based router determines the relevance vector $\mathbf{r}$ in Eq. (4), depending on we use the top-1 and top-k experts.

Single-Expert: In the single-expert case ($k = 1$), the relevance vector is just one-hot vector, the router-applied MMoE layer is degenerated into Eq. (6).

Multi-Expert: In the multi-experts case, the relevance vector based on the dense retrieval is computed as follows:

$$\mathbf{r} = \mathrm{softmax}\left(\mathrm{top}_k\left(\mathsf{Emd}(q)^T W_{\mathcal{C}}\right)\right) \quad (10)$$

where $\mathsf{Emd}(q) \in \mathbb{R}^{d_{\mathsf{emd}}}$ is the query embedding vector, d_{emd} is the dimension of embedding vectors for query and passages, and $W_{\mathcal{C}} \in \mathbb{R}^{d_{\mathsf{emd}} \times N}$ is the matrix consisting of all passage embeddings in the collection $\mathcal{C}$, defined as follows:

$$W_{\mathcal{C}} = [\mathsf{Emd}(p_1), \cdots, \mathsf{Emd}(p_N)] \quad (11)$$

When reranking is applied, we then use reranking-based scores over all passages, resulting in the relevance vector, with a slight abuse of notation, as follows:

$$\mathbf{r} = \mathrm{softmax}\left(\mathrm{top}_k\left([score_{rerank}(q, p_i)]_{i=1}^N\right)\right) \quad (12)$$

4 Experiments

In this part, we explain the experimental setup and present the key results of our findings.

4.1 Experimental Setup

Dataset: We utilize the Natural Questions (NQ) dataset (Kwiatkowski et al., 2019) for training to enhance the model's performance on open-domain question answering. To further evaluate the model's generalization ability, we also test it on the TriviaQA (TQA) (Joshi et al., 2017) dataset, which presents diverse and challenging questions from trivia domains.

4.2 RAG settings

We explore two variants of RAGs – **Direct-RAG**, and **Prompt-RAG** – using prompting templates to evaluate the RAG capabilities of the models, with the input format shown in Table 4.

[3] https://huggingface.co/BAAI/bge-reranker-v2-gemma

4.3 MoPE settings

General: For a given test query q_{test}, we first perform retrieval to identify top-k experts and apply the retrieval-based router by determining relevance vectors depending on single-expert and multi-expert cases, as in Section 3.2.2. Suppose that $\mathcal{M}_{MoPE}$ is the model using MMoE based on Eq. (4) at λ_l^t is 1 only for $l = 12$. The final answer a is determined as follows.

$$a = \arg\max_{a} \mathcal{M}_{MoPE}(a|q_{test}) \tag{13}$$

RAG on MoPE We apply RAG under MoPE but use a mixed approach that combines the original pre-edited base model and the MoPE-based model. More specifically, we use the original model until the passage knowledge is encoded, and then switch to MoPE when processing the question part.

Thus, given the specific layer l where new experts are additionally located, λ_t^l is defined as:

$$\lambda_t^l = \begin{cases} 0, & \text{if } t \leq L_p \text{ or } l \neq l_{\text{moe}} \\ 1, & \text{otherwise} \end{cases} \tag{14}$$

where L_p is the last token position where passage knowledge is injected into a prompted RAG, and l_{moe} is the layer into which experts are newly added.

Backbone Language Models: We select Llama2-7B (Touvron et al., 2023) as the base model and integrate expert modules into it to implement our framework. We also compare our approach with several advanced models (Yang et al., 2023; Team, 2024; Jiang et al., 2023a) built on Llama2-7B, as well as the larger Llama2-13B model (Touvron et al., 2023) with more parameters.

Metric: We use Exact Match (EM) and F1-score (F1), which are standard metrics (Rajpurkar, 2016) in question answering for evaluating answer accuracy and completeness.

Implementation Details: All experiments, including data construction, knowledge injection, and evaluation, were conducted on workstations with 8×NVIDIA RTX A6000 GPUs. For training the experts, we used the AdamW optimizer for 10 epochs, with the learning rate decaying from 1e-4 to 1e-6 using cosine annealing.

4.4 Why l=12 was Chosen?

(Wang et al., 2023) suggests that the information captured at different layers can vary in its contribution to the final output, especially for inputs with contextual information. We randomly selected a small subset of sample questions for an expert insertion experiment across different layers. As show in figure 3, the results indicate that the 12th layer has the most significant impact on model performance, outperforming other layers. Although performance differences across layers are generally minor, the middle layers, particularly the 12th layer, demonstrate better performance in integrating injected knowledge compared to the initial and final layers.

Methods	NQ EM	NQ F1	TQA EM	TQA F1
Without-RAG				
Llama2$_{7B}$	7.80	16.89	49.47	59.38
Llama2$_{13B}$	10.40	20.62	50.87	61.21
Baichuan2$_{7B}$	8.60	16.84	43.73	53.29
QWen2.5$_{7B}$	1.20	6.24	31.53	40.19
Mistral$_{7B}$	3.27	12.11	52.60	61.71
PEFT-Lora$_{7B}$	14.87	23.66	51.73	61.37
MoPE$_{7B}$	**26.40**	**35.40**	**56.13**	**64.08**
Direct-RAG				
Llama2$_{7B}$	13.60	23.46	54.40	64.49
Llama2$_{13B}$	15.20	25.79	55.53	65.77
Baichuan2$_{7B}$	11.67	21.32	50.40	60.11
QWen2.5$_{7B}$	0.07	7.47	4.27	19.09
Mistral$_{7B}$	11.20	21.76	53.60	63.75
PEFT-Lora$_{7B}$	20.20	30.86	62.21	71.13
MoPE$_{7B}$	**38.87**	**49.27**	**68.27**	**75.64**
Prompt-RAG				
Llama2$_{7B}$	29.00	40.61	60.80	69.39
Llama2$_{13B}$	29.73	40.57	63.00	71.71
Baichuan2$_{7B}$	28.87	38.14	57.20	65.11
QWen2.5$_{7B}$	29.80	38.80	55.27	61.95
Mistral$_{7B}$	32.07	44.43	59.80	68.71
PEFT-Lora$_{7B}$	34.26	46.54	65.34	74.28
MoPE$_{7B}$	**39.53**	**49.86**	**68.33**	**75.48**

Table 1: Results on the NQ dataset, where inference is performed with a single expert.

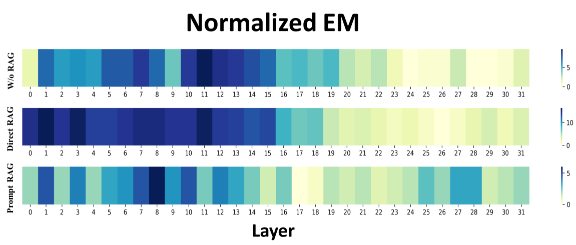

Figure 3: Normalized EM metric for expert insertion across different layers.

		1 expert		2 expert		3 expert		4 expert	
	Methods	EM	F1	EM	F1	EM	F1	EM	F1
NQ	W/o-RAG	26.40	35.40	28.20	37.15	28.93	37.41	29.07	37.65
	D-RAG	38.87	49.27	39.87	50.36	40.40	50.77	40.53	50.99
	P-RAG	39.53	49.86	40.20	50.20	40.40	50.44	40.06	50.59
TQA	W/o-RAG	56.13	64.08	57.07	64.47	58.27	65.74	57.73	65.39
	D-RAG	68.27	75.64	69.27	76.12	69.47	76.25	69.20	75.92
	P-RAG	68.33	75.48	69.53	76.08	69.07	75.69	68.47	75.38

Table 2: The results of inference with varying numbers of experts.

4.5 Main Results

Table 1 presents a comparison of MoPE with other baseline methods under the W/o-RAG, D-RAG, and P-RAG settings. As shown in the table, when expert modules are injected with paragraph knowledge, the performance of MoPE on the LLaMa2-7B model significantly surpasses that of the original model in answering questions. On the other hand, by injecting question-style knowledge, MoPE's performance under D-RAG is comparable to the results obtained from providing prompts in P-RAG, suggesting that the method can achieve similar outcomes without explicit prompts. This opens up new possibilities for tasks such as simplifying or distilling in-context learning.

We also tested the performance after varying the number of experts involved in the query, as shown in the table 2. The results indicate that as the number of experts increases, the model's performance, measured by both EM and F1 scores, gradually improves, particularly under the D-RAG and P-RAG settings, where the performance boost is more pronounced. This suggests that knowledge injection through expert modules effectively enhances the model's performance, and with the increase in the number of experts, the model is able to better leverage different paragraph of knowledge. Under the W/o-RAG setting, despite the absence of external prompts, the model's performance remains relatively stable, yet still improves as the number of experts increases, demonstrating the effectiveness of expert injection.

4.6 Efficiency Analysis of Expert Insertion

Table 3 shows the throughput comparison between the base LLama2-7B model and the MoPE framework. Despite the additional steps required for loading expert parameters, and performing inference, the throughput of MoPE (0.598 samples/s) remains close to that of the base model (0.628 samples/s). This indicates that the expert selection and integration process introduces minimal time overhead. The marginal reduction in throughput is compensated by a significant improvement in accuracy, demonstrating that MoPE effectively balances efficiency and performance. This makes it a practical solution for enhancing model accuracy without compromising too much on processing speed.

Method	Throughput
Base	0.628
MoPE	0.598

Table 3: Throughput comparison between the base model and MoPE.

5 Conclusion

In this study, we proposed the MoPE framework, which applies knowledge injection via MMoE to address the issue of LLMs' insufficient attention to external prompts in RAG. Given a query, relevant paragraphs are transformed into edited knowledge that the model can learn and store in the designed expert modules. Experiments on NQ and TQA demonstrated that this expert memory-based approach mitigates the inadequate attention to external prompts, significantly enhancing performance in both W/o RAG and W/ RAG modes.

In future work, we aim to explore the concept of 'meta-learning experts,' where trained offline experts do not require additional storage. Instead, we propose dynamically storing expert parameters within a small-scale supernetwork, using it to generate expert parameters based on the query. This approach reduces storage costs while enabling more flexible expert selection and knowledge injection. Furthermore, we plan to investigate stable continual learning mechanisms for updating the supernetwork, allowing it to adapt to new textual data and improve the model's long-term adaptability.

Limitations

In the current setup, our framework requires training an offline expert repository, which necessitates a certain amount of storage space to maintain these expert modules. Future research will explore the use of compact hypernetworks to dynamically generate expert parameters, potentially reducing the storage requirements. Additionally, more advanced expert merging techniques need to be investigated to better leverage complementary knowledge from different experts.

Furthermore, the quality of the generated QA pairs for editing significantly influences the performance of the experts. It is crucial to avoid injecting incorrect knowledge into the experts. However, as shown in Appendix B, the quality of QA pairs generated by T5-xl as the question generator is not always satisfactory and sometimes contains errors. Future work should explore or propose more advanced components and methodologies for converting paragraphs into correct editing facts without relying on structured data. This approach could also overcome limitations of existing editing methods, such as MEMIT (Meng et al., 2022b), which depend on fixed editing formats like triple-based data.

References

Simone Alghisi, Massimo Rizzoli, Gabriel Roccabruna, Seyed Mahed Mousavi, and Giuseppe Riccardi. 2024. Should we fine-tune or rag? evaluating different techniques to adapt llms for dialogue. *arXiv preprint arXiv:2406.06399*.

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. 2017. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3366–3375.

Uri Alon, Frank Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. 2022. Neuro-symbolic language modeling with automaton-augmented retrieval. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 468–485. PMLR.

Szymon Antoniak, Sebastian Jaszczur, Michał Krutul, Maciej Pióro, Jakub Krajewski, Jan Ludziejewski, Tomasz Odrzygóźdź, and Marek Cygan. 2023. Mixture of tokens: Efficient llms through cross-example aggregation. *arXiv preprint arXiv:2310.15961*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Siyuan Cheng, Bozhong Tian, Qingbin Liu, Xi Chen, Yongheng Wang, Huajun Chen, and Ningyu Zhang. 2023. Can we edit multimodal large language models? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13877–13888, Singapore. Association for Computational Linguistics.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*.

Guanting Dong, Yutao Zhu, Chenghao Zhang, Zechen Wang, Zhicheng Dou, and Ji-Rong Wen. 2024. Understand what llm needs: Dual preference alignment for retrieval-augmented generation. *arXiv preprint arXiv:2406.18676*.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, et al. 2023. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. *arXiv preprint arXiv:2312.09979*, 4(7).

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

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

Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. 2023. Pokemqa: Programmable knowledge editing for multi-hop question answering. *arXiv preprint arXiv:2312.15194*.

Aman Gupta, Anup Shirgaonkar, Angels de Luis Balaguer, Bruno Silva, Daniel Holstein, Dawei Li, Jennifer Marsman, Leonardo O Nunes, Mahsa Rouzbahman, Morris Sharp, et al. 2024. Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture. *arXiv preprint arXiv:2401.08406*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*.

Sebastian Hofstätter, Jiecao Chen, Karthik Raman, and Hamed Zamani. 2023. Fid-light: Efficient and effective retrieval-augmented text generation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1437–1447.

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

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

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

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*.

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*, 7:453–466.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

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

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18564–18572.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, et al. 2023. Ra-dit: Retrieval-augmented dual instruction tuning. *arXiv preprint arXiv:2310.01352*.

Hongyin Luo, Yung-Sung Chuang, Yuan Gong, Tianhua Zhang, Yoon Kim, Xixin Wu, Danny Fox, Helen Meng, and James Glass. 2023. Sail: Search-augmented instruction learning. *arXiv preprint arXiv:2305.15225*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Xiaonan Nie, Xupeng Miao, Shijie Cao, Lingxiao Ma, Qibin Liu, Jilong Xue, Youshan Miao, Yi Liu, Zhi Yang, and Bin Cui. 2021. Evomoe: An evolutional mixture-of-experts training framework via dense-to-sparse gate. *arXiv preprint arXiv:2112.14397*.

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2023. Fine-tuning or retrieval? comparing knowledge injection in llms. *arXiv preprint arXiv:2312.05934*.

Bowen Pan, Yikang Shen, Haokun Liu, Mayank Mishra, Gaoyuan Zhang, Aude Oliva, Colin Raffel, and Rameswar Panda. 2024. Dense training, sparse inference: Rethinking training of mixture-of-experts language models. *arXiv preprint arXiv:2404.05567*.

P Rajpurkar. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Carl Rasmussen and Zoubin Ghahramani. 2001. Infinite mixtures of gaussian process experts. *Advances in neural information processing systems*, 14.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Chenmien Tan, Ge Zhang, and Jie Fu. 2023. Massive editing for large language models via meta learning. *arXiv preprint arXiv:2311.04661*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2022. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558*.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Label words are anchors: An information flow perspective for understanding in-context learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9840–9855, Singapore. Association for Computational Linguistics.

Renzhi Wang and Piji Li. 2024a. LEMoE: Advanced mixture of experts adaptor for lifelong model editing of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2551–2575, Miami, Florida, USA. Association for Computational Linguistics.

Renzhi Wang and Piji Li. 2024b. Memoe: Enhancing model editing with mixture of experts adaptors. *arXiv preprint arXiv:2405.19086*.

Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of lora experts. *arXiv preprint arXiv:2404.13628*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19449–19457.

Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2022. A survey of knowledge-enhanced text generation. *ACM Computing Surveys*, 54(11s):1–38.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

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? *arXiv preprint arXiv:2305.12740*.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

A RAG Template

In this section, we present the input formats for *without-RAG*, *Direct-RAG*, and *Prompt-RAG*.

Input Format of Without-RAG
Question: {***question***} Answer:

(a) Input Format of Without-RAG.

Input Format of Direct-RAG
Knowledge: {***Top-1 paragraph***} Question: {***question***} Answer:

(b) Input Format of Direct-RAG.

Input Format of Prompt-RAG
Knowledge: {***Top-1 paragraph***} Base above knowledge, answer the following question with a very short phrase, such as “1998”, “May 16th, 1931”, or “James Bond”, to meet the criteria of exact match datasets. Question: {***question***} Answer:

(c) Input Format of Prompt-RAG

Table 4: The input format of the LLM.

B Synthetic Q-A pairs

The process consists of the following steps:

1. We leverage the spacy [4] library to perform Named Entity Recognition (NER) on the passage p, obtaining a set of entities:

$$A = spacy(p) \tag{15}$$

2. For each entity $a \in A$, we use the trained question generator G_θ to generate a corresponding question.

$$q_{edited} = G_\theta(q|\mathcal{T}(p, a)) \tag{16}$$

[4]https://spacy.io/

Table 6 presents an example of synthetic QA generation using G_θ, where the passage is the retrieved paragraph, followed by several generated QA pairs. As observed, the quality of the generated pairs is not perfect, with irrelevant or incorrect questions, such as "*Where does the story Don Quixote take place?*" with the answer "*Western*," which is factually incorrect, or "*Sancho Panza was the steed of which fictional character?*" with the answer "*Don Quixote's*", which is a misleading formulation. These inaccuracies and mismatches between questions and answers may significantly impact the overall system performance. This highlights a key factor affecting the results, making it evident that generating high-quality and factually accurate edited knowledge is crucial. Therefore, we are currently exploring methods to improve the accuracy and relevance of these generated QA pairs, with a focus on more reliable techniques for editing and validating the knowledge. The need to generate high-quality edited knowledge is urgent to ensure that the knowledge injection process leads to better, more precise performance in downstream tasks.

C Re-ranker Results

This section presents the performance of applying a re-ranker to refine the retrieval results of DPR (Karpukhin et al., 2020), comparing the accuracy before and after re-ranking. In a question-answering system, the accuracy of the retrieval source plays a crucial role in the overall result. Additionally, the scoring produced by the re-ranker is important for merging between experts. The improvements were evaluated using Top-K accuracy, which measures the presence of a golden passage, ensuring a stable document source for responing answer.

Top-K	Type	NQ	TQA
@1	Before	44.60	56.53
	After	64.67	76.33
@2	Before	55.73	65.27
	After	71.73	79.60
@4	Before	64.47	72.07
	After	77.87	82.80
@8	Before	72.93	76.73
	After	81.47	84.53

Table 5: Comparison of DPR retrieval accuracy results before and after applying the re-ranker we trained.

Passage: "forced to deceive him at certain points. The novel is considered a satire of orthodoxy, veracity and even nationalism. In exploring the individualism of his characters, Cervantes helped move beyond the narrow literary conventions of the chivalric romance literature that he spoofed, which consists of straightforward retelling of a series of acts that redound to the knightly virtues of the hero. The character of Don Quixote became so well known in its time that the word q̈uixoticẅas quickly adopted by many languages. Characters such as Sancho Panza and Don Quixote's steed, Rocinante, are emblems of Western literary culture."

"question": "Who wrote the novel don quixote and the adventures of sancho panza?",
"answer": "Cervantes"

"question": "Sancho Panza is a character in which novel?"
"answer": "Don Quixote"

"question": "What was the name of Don Quixote's faithful companion?"
"answer": "Sancho Panza"

"question": "Sancho Panza was the steed of which fictional character?"
"answer": "Don Quixote's"

"question": "What was the name of Don Quixote's horse?"
"answer": "Rocinante"

"question": "Where does the story don quixote take place?"
"answer": "Western"

Table 6: An example of synthetic QA pairs.