Reference Trustable Decoding: A Training-Free Augmentation Paradigm for Large Language Models

Luohe Shi[1,†], **Yao Yao**[2], **Zuchao Li**[1,†,*], **Lefei Zhang**[1], **and Hai Zhao**[2]
[1]National Engineering Research Center for Multimedia Software,
School of Computer Science, Wuhan University, Wuhan, 430072, P. R. China
[2]Department of Computer Science and Engineering, Shanghai Jiao Tong University
shiluohe@whu.edu.cn, yaoyao27@sjtu.edu.cn, zcli-charlie@whu.edu.cn,
zhaohai@cs.sjtu.edu.cn

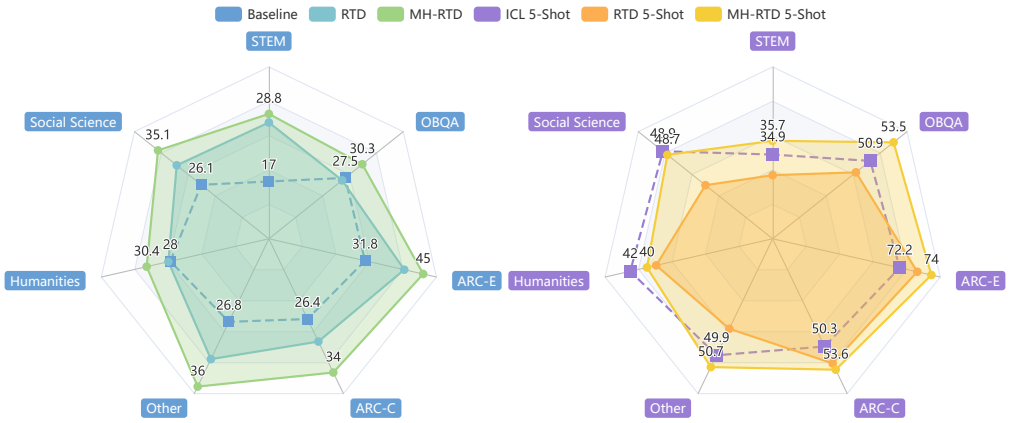

Figure 1: Performance comparison between default LLM and reference trustable decoding in reasoning tests.

Abstract

Large language models (LLMs) have rapidly advanced and demonstrated impressive capabilities. In-Context Learning (ICL) and Parameter-Efficient Fine-Tuning (PEFT) are currently two mainstream methods for augmenting LLMs to downstream tasks. ICL typically constructs a few-shot learning scenario, either manually or by setting up a Retrieval-Augmented Generation (RAG) system, helping models quickly grasp domain knowledge or question-answering patterns without changing model parameters. However, this approach involves trade-offs, such as slower inference speed and increased space occupancy. PEFT assists the model in adapting to tasks through minimal parameter modifications, but the training process still demands high hardware requirements, even with a small number of parameters involved. To address these challenges, we propose Reference Trustable Decoding (RTD), a paradigm that allows models to quickly adapt to new tasks without fine-tuning, maintaining low inference costs. RTD constructs a reference datastore from the provided training examples and optimizes the LLM's final vocabulary distribution by flexibly selecting suitable references based on the input, resulting in more trustable responses and enabling the model to adapt to downstream tasks at a low cost. Experimental evaluations on various LLMs using different benchmarks demonstrate that RTD establishes a new paradigm for augmenting models

* Corresponding author. † Equal contribution.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

to downstream tasks. Furthermore, our method exhibits strong orthogonality with traditional methods, allowing for concurrent usage. Our code can be found at https://github.com/ShiLuohe/ReferenceTrustableDecoding

1 Introduction

In the rapidly advancing field of artificial intelligence, Large Language Models (LLMs) have demonstrated substantial progress. With their extensive parameter size, LLMs have acquired emergent abilities [41] and been able to tackle diverse and challenging tasks in fields like education [22] and medicine [38]. Despite their immense potential, Large Language Models that have just completed pre-training often struggle to effectively adapt to downstream tasks. Moreover, the process of adapting the model is typically costly and requires careful execution by experienced individuals. Otherwise, it could lead to the model generating hallucination [50; 28] at best, or at worst, result in a loss of its language capabilities.

In-Context Learning (ICL), as a category of methods that do not require parameter adjustments, is one of the mainstream methods for adapting models to downstream tasks. ICL embeds domain knowledge, question-answering patterns, etc., into prompts through few-shot learning [6], prompt engineering [51], and Retrieval-Augmented Generation (RAG) [26] methods, leveraging the learning ability of the model itself to provide better answers. As pointed out in Figure 2, ICL focuses on the prompt stage. However, ICL significantly increases the length of the input, consequently increases the space occupied by the KV-Cache required for inference. Further, according to the Roofline model [46], this part of the KV-Cache cannot be parallelized through batch processing, making memory I/O throughput a system bottleneck, wasting hardware computing power, and increasing token generation time during the entire inference stage.

Fine-tuning is also used to adapt models to downstream tasks. By fine-tuning the pre-trained model based on domain tasks, the model can quickly acquire capabilities within the domain. However, traditional full-parameter fine-tuning often requires a large amount of resources (empirically 8-15 times that of inference), making Parameter-Efficient Fine-Tuning (PEFT) a more popular method. By freezing most parameters and only modifying a few, methods such as Adapters, P-tuning [27], LoRA [16] and others [48; 36; 44] have become mainstream methods for quickly adapting models to downstream tasks. However, fine-tuning methods introduce several hyperparameters, which require high experience from the fine-tuners and the effects are unpredictable. Furthermore, due to the need for backpropagation, the computation graph must be saved, meaning that even if only a few parameters need to be updated, there will be a large amount of additional computation and space requirements (several times that of inference), raising the threshold for methods based on fine-tuning.

To address these challenges, we introduce Reference Trustable Decoding (RTD), a novel framework designed to fit LLMs for downstream tasks. Distinct from a conventional `LM_Head` module, RTD strategically retrieves relevant references from a pre-constructed datastore, guided by the final hidden states of the language model. This approach not only enhances the final output distribution by recalculating it with the similarity score of the retrieved references but also allows for the seamless integration of new knowledge or constraints into the response generation process without increasing the input length or using gradient descent.

RTD, distinctively training-free, emphasizes compact input lengths to expedite inference. RTD's effectiveness was rigorously tested using varied benchmarks focused on different tasks and a Wikipedia-based knowledge injection scenario. On these benchmarks, RTD achieved results comparable to traditional methods like PEFT and ICL, providing significant improvement. Additionally, we combined RTD with traditional methods, further enhancing the model's capabilities and demonstrating the good orthogonality of RTD with other approaches.

Our contribution includes:

- We propose a new paradigm, called RTD, for fitting LLMs for downstream tasks. RTD is a training-free method that focused on the decoding stage of large language models (LLMs), as a alternation of `LM_Head`. It helps LLMs to adapt to different tasks with different demands and provide trustable response.
- RTD has achieved performance comparable to, or even better than, ICL and PEFT across different benches, while maintaining the desirable properties of training free and not intro-

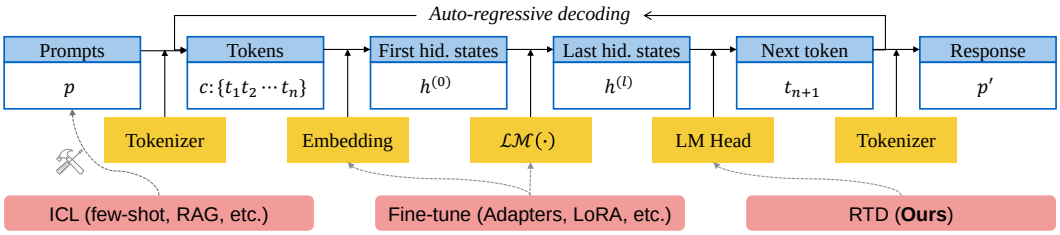

Figure 2: The pipeline of LLM inference and the focus of different methods: ICL focuses on the prompt stage, emphasizing the optimization of the model's input. Fine-tuning methods optimize the model itself by adjusting its parameters. In contrast, our proposed RTD method targets the decoding stage of the language model. By constructing a reference datastore, RTD optimizes the final output distribution without requiring additional training.

ducing additional input lengths. This demonstrates the potential of RTD as a new paradigm for LLMs to adapt to downstream tasks. Furthermore, RTD can be seamlessly integrated with other existing methods, such as in-context learning (ICL) and fine-tuning. The combination of RTD, ICL, and fine-tuning has the potential to achieve even higher performance.

2 Background and Related Work

In the field of NLPs, Transformer models have gained influence rapidly after it get original proposed in 2017. As larger scaled model been introduced, especially giant ones like GPT3 which has 175 Billion of parameters [9], the training process is getting more and more expensive, hence to fit the LLMs for downstream tasks.

2.1 Fine-tuning

Full Parameter Fine-tuning Full parameter fine-tuning refers to fully optimizing all the parameters of the model during the fine-tuning process. Full parameter fine-tuning has the advantage of allowing the model to adapt more closely to the specific task at hand, as well as injecting more information into the model. However, it also has the disadvantage of being the most computationally expensive and time-consuming, as it requires to manipulate all parameters of the model, with the modern optimizer like Adam [19], 8 to 15 times of more extra GPU memory is demanded comparing to inference empirically, resulting a must of multi-GPU server or even cross sever training.

Parameter Efficient Fine-tuning Parameter Efficient Fine-tune (PEFT), for example, LoRA [16] and P-tuning [27], is introduced to make fine-tune more reachable. By freezing most of the model parameters and only let a small amount of them accumulate gradient, the GPU memory and computation resource can be cut down by a large margin [14].

However, as fine-tuning introduce many tricky hyper-parameters like learning rate, the process is heavily task related and empirical, even experienced fine-tuner need some trials and error when tuning them. Moreover, even if the number of parameters trained is not large, processes such as backpropagation still need to be carried out. The computation graph generated on long sequences will also occupy a large amount of memory, making the threshold for computing power and memory still high, which any method that relies on gradient descent is difficult to avoid.

2.2 In-Context Learning

Few-Shot Learning Few-shot Learning is proved to be a great way for LLMs to gain capability. By appending the true task that LLMs are expecting to response after a couple of existing correct examples, LLMs can gain its reasoning ability [6; 31].

Retrieval Augmented Generation Retrieval Augmented Generation (RAG) [24] is an AI framework for retrieving facts from an external knowledge source to LLMs, which helps LLMs correct its hallucination and use latest fact [35]. RAG is to cut external knowledge source into multiple chunks,

then embed and store them in a database, then retrieve them at the process of generation to let LLMs get the knowledge in it. This technique allowed LLM to use extra information while maintaining their parameters untouched. RAG have been used on multiple fields, like coding [26] and question answering [29]. And it can be combined with few-shot [18].

The main drawback of ICL methods lies in their growth of the input sequence. Under the quadratic complexity of the Transformer architecture, this implies a longer KV-Cache, which not only increases the latency during the pre-fill stage but also adds delay each time a token is generated [45]. Moreover, unlike model parameters, each instance needs to save its dedicated portion of KV-Cache, leading to memory I/O bottlenecks and computational power waste. Finally, on some smaller models, the irrelevant information that ICL might contain can confuse the model, resulting in performance loss.

3 Reference Trustable Decoding

In this section, we begin by presenting the fundamental formulas and concepts to elucidate the workings of Reference Trustable Decoding, followed by an exploration of the multi-head Reference Trustable Decoding method.

3.1 Preliminary

Given an input sentence $c = \{t_1, t_2, ..., t_n\}$, where t_i represents the i-th token and n denotes the sentence length, the last token's output of the last Transformer block in the language model can be represented as:

$$h^{(l)} = \mathcal{LM}(c) \tag{1}$$

In this equation, $h^{(l)} \in \mathbb{R}^{d_m}$ is the output of the last token from the final, or the l-th, Transformer block of the language model, where d_m denotes the hidden size of the model.

Traditionally, a standard decoder-only architecture Transformer usually employs `LM_Head`, which is, a fully connected layer, usually includes a learnable weight matrix W and no bias, followed by a softmax function $\mathbf{Softmax}(\cdot)$ to predict the output probability distribution $\mathbf{p}$ of the next token from the last hidden states:

$$\mathbf{p} = \texttt{LM\_Head}(h^{(l)}) = \mathbf{Softmax}(W \cdot h^{(l)}) \tag{2}$$

where v is the vocabulary size and $W \in \mathbb{R}^{v \times d_m}$.

However, traditional next token prediction does not support incorporating external information and therefore, we introduce reference trustable decoding where we build a bypass around the `LM_Head`, showcased in Figure 3, as the entrance of additional knowledge or guidance.

3.2 Reference Trustable Decoding

3.2.1 Generation of Reference Datastore

In reference trustable decoding, we first build the reference datastore $\mathcal{L}$, which stores key-value pairs $(k, v) \in (\mathcal{K}, \mathcal{V})$. Here, the key $k = \mathcal{LM}(c)$ represents the last hidden states of the token generated by the LMs from the context c, and the value v is the corresponding label y. Mathematically, we have:

$$\mathcal{L} = \{(k, v) | (k, v) \in (\mathcal{K}, \mathcal{V})\} = \{(\mathcal{LM}\left(c\right), y) \mid (c, y) \in \mathcal{D}\} \tag{3}$$

where $\mathcal{D} = (\mathcal{C}, \mathcal{Y})$ is the task dataset with input context set $\mathcal{C}$ and label set $\mathcal{Y}$, and $|\mathcal{Y}|$ refers the number of possible labels. This process is depicted in Figure 3. It's obvious that **the computational requirement is same as performing a forward pass to every content in the task dataset**, which aligned with the minimal requirement of the inference stage, denotes the superiority of RTD as a gradient-free method.

3.2.2 Decoding Stage

At each decoding round, given the input context c, we first compute $h^{(l)} = \mathcal{LM}(c)$, which is the input to RTD and `LM_Head`. Then we use a three stage approach to get the RTD output, **Fetch**, **Normalization**, and **Aggregation**, depicted in Figure 4.

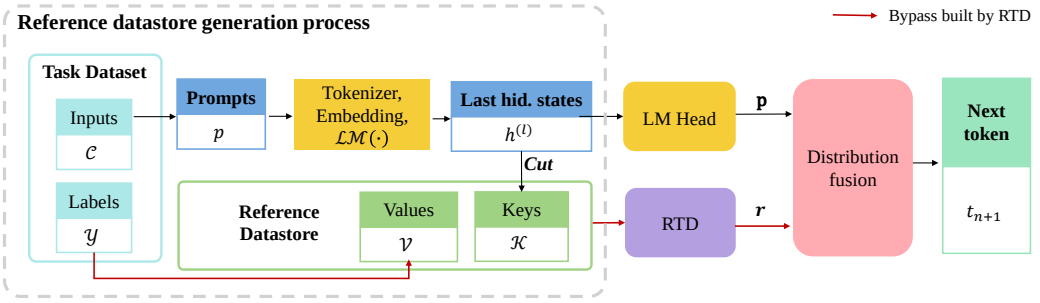

Figure 3: Overview of the reference datastore generation and reference trustable decoding process.

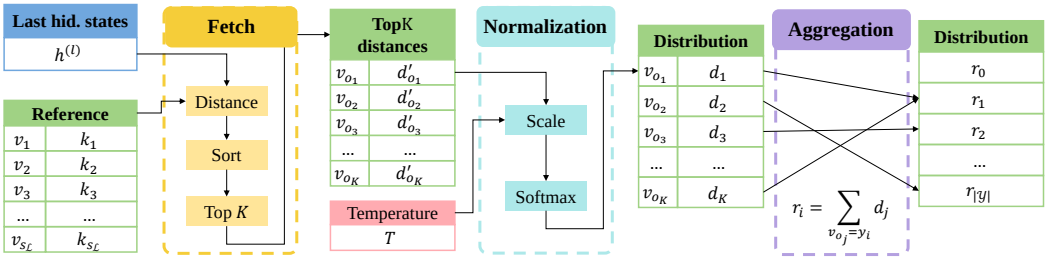

Figure 4: Three stages of reference trustable decoding.

Fetch First, we calculate the distance d' between $h^{(l)}$ and all the k in the reference datastore $\mathcal{L}$. Otherwise stated, we use the Euclidean distance $d'_i = ||h^{(l)} - k_i||_2$. We then select the top K instances from $\mathcal{L}$ which have the smallest distance, and for the j-th ($1 \leq j \leq K$) closest (k_i, v_i), we define $o_j = i$. Then we create a set L_h, storing the top K distances and values:

$$L_h = \{(d'_{o_j}, v_{o_j})\} = \{(||h^{(l)} - k_{o_j}||_2, v_{o_j})\}, \quad o_j = i \text{ for } j\text{−th closest } (k_i, v_i) \tag{4}$$

Normalization We first scale the d' we got from the previous stage by temperature T, as $d''_j = d'_{o_j}/T$. The scale operation is introduced to prevent overflow in the following Softmax operation. We take the Softmax of $-d''$ as d, guaranteed d as a valid possibility distribution.

$$d = \mathbf{Softmax}(-d''), \quad d_j = \frac{\exp\{-d''_j\}}{\sum_{\iota=1}^{K} \exp\{-d''_\iota\}} = \frac{\exp\{-d'_{o_j}/T\}}{\sum_{\iota=1}^{K} \exp\{-d'_{o_\iota}/T\}} \tag{5}$$

Aggregation We calculate the final reference possibility distribution $\mathbf{r} = [r_1, r_2, ..., r_{|\mathcal{Y}|}] \in \mathbb{R}^{|\mathcal{Y}|}$ by aggregating all d_j that satisfies $v_{o_j} = y_i$, where $y_i \in \mathcal{Y}$.

$$r_i = \sum_{v_{o_j}=y_i} d_j \tag{6}$$

We denote $\mathcal{R}(\cdot, \mathcal{L}) : \mathbb{R}^{d_m} \to \mathbb{R}^{|\mathcal{Y}|}$ as the function represents all three stages of querying the datastore $\mathcal{L}$ and building the corresponding reference possibility distribution $\mathbf{r}$. Therefore, we have

$$\mathbf{r} = \mathcal{R}(h^{(l)}, \mathcal{L}) \tag{7}$$

Additionally, when $|\mathcal{Y}| = v$, we can merge the distribution $\mathbf{p}$ given by $\texttt{LM\_Head}(\cdot)$ and $\mathbf{r}$ given by $\mathcal{R}(\cdot, \mathcal{L})$ with a hyper-parameter λ:

$$d' = \lambda \cdot \mathbf{r} + (1 - \lambda) \cdot \mathbf{p} \tag{8}$$

which is a common fusion method for mixing two distributions [34; 23; 12; 10].

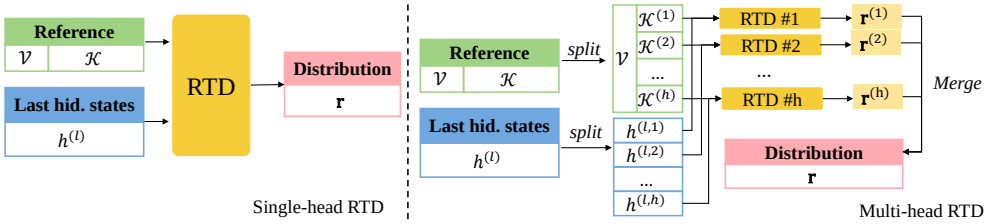

Figure 5: Comparison between RTD and multi-head RTD.

Table 1: Comparison of RTD and MH-RTD on Open Book QA.

Method	RTD	MH-RTD
MPT-7B	27.4	30.9
LLaMA2-7B	47.1	52.4
LLaMA2-70B	63.3	65.6

3.3 Multi-head Reference Trustable Decoding

Large language models like LLaMA2-70B [39] or Mistral-7B [1] utilized MHA and GQA mechanism [2], implies the potential of splitting a large attention vector into smaller ones. So we adapt this method into our RTD process. We define n_h of the head count of the LM model, and d_h the dimension of the each attention head where $d_m = n_h \cdot d_h$. with this in mind, we split the reference datastore into n_h sub-datastore by head. When decoding, we first split $h^{(l)}$ in to heads, then query each sub-datastore and merge the result, showcased in Figure 5. Mathematically,

$$\begin{aligned} k^{(i)} &= k\left[d_h \times (i-1):\ d_h \times i\right], \quad h^{(l,i)} = h^{(l)}\left[d_h \times (i-1):\ d_h \times i\right] \\ \mathcal{L}^{(i)} &= \{(k^{(i)}, v) | (k, v) \in (\mathcal{K}, \mathcal{V})\} \end{aligned} \tag{9}$$

And we denote $\mathcal{R}_{\text{MH}}(\cdot, \mathcal{L})$ as the function of the multi-head RTD query process, we have:

$$\mathbf{r} = \mathcal{R}_{\text{MH}}(h^{(l)}, \mathcal{L}) = \frac{1}{n_h} \sum_{i=1}^{n_h} \mathcal{R}(h^{(l,i)}, \mathcal{L}^{(i)}) \tag{10}$$

3.4 Time and Memory Consumption

Time Consumption The time consuming is largely depended on the vector datastore used. For a brute force searching datastore, the time complexity will be $\mathcal{O}(s_{\mathcal{L}} \cdot d_m)$ where $s_{\mathcal{L}} = |\mathcal{L}|$ is the size of the datastore. However, for those more powerful database like faiss [20] by Meta, with extra training after the generation of reference datastore, the process which have to be done again if the datastore changes, the time consumption can be cut to $\mathcal{O}(k \cdot d_m)$, where k is a constant related the parameters used to train the database.

For multi-head reference trustable decoding, the performance cost remains the same. The time complexity of each attention-head wise query is $\mathcal{O}(d_h \cdot s_{\mathcal{L}})$, the overall query time complexity is $\mathcal{O}(n \cdot d_h \cdot s_{\mathcal{L}}) = \mathcal{O}(d \cdot s_{\mathcal{L}})$, which is the time complexity of convention reference trustable decoding processing. The calculation remains the same for a trained database, the overall time complexity is $\mathcal{O}(n \cdot k \cdot d_h) = \mathcal{O}(k \cdot d_m)$.

Memory Consumption The use of time can be optimized by utilizing vector database, however the memory consumption cannot shrink easily. We further define b as the bit cost of the models' `dtype`, where $b_{\texttt{float32}} = 4, b_{\texttt{float16}} = b_{\texttt{bfloat16}} = 2, b_{\texttt{int8}} = 1, b_{\texttt{int4}} = \frac{1}{2}$. The overall memory cost is $d_m \cdot b \cdot s_{\mathcal{L}}$. Due to the lack of lower precision `dtype` support on CPU, even the base model utilized popular half precision `dtype` like `bfloat16`, it still need to be converted into larger ones to be stored. Since all the hidden states have to be saved to calculate precise distanced when rescalled, the memory cost can't be reduced significantly by making it irrelevant with $s_{\mathcal{L}}$.

On the Multi-head RTD side, the memory cost remains the same as the regular RTD takes. The proof is same as the Section 3.4. For instance, reference datastore and head-wise reference datastore with $20,480$ entries with $d_m = 4096$, $n = 32$ and $d_h = 128$, stored in `float32`, takes 320`MB` of memory and hard disk space.

MH-RTD for Resource Saving As MH-RTD splits long vectors into multiple smaller ones, it gives us the opportunity to cut time and memory cost by merging different heads together, or directly evict some of them. If on average, p heads are merged into one head, then we expect a $\frac{1}{p}$ resource consumption. The time and memory improvement and corresponding performance impact can be find in the tuning Section 4.3.

4 Settings and Experiment

We categorize the common downstream tasks of language models into two types: language understanding and language generation. The former focuses on understanding the input information, based on the context and the information stored within the model, and then outputs the answer in the form of a few tokens, usually in a very simple form. The latter focuses on generating new sentences with complete semantics. We explored the potential of RTD compared to other methods on these two types of tasks. We first compared the effects of RTD and MH-RTD. As shown in Table 1, we found that MH-RTD effectively enhances the capabilities of RTD. Therefore, we default to using the MH-RTD method in the following tests.

4.1 Language Understanding

We tested the language understanding capabilities of RTD on multiple benchmarks. When testing, question without answer be shown to the LLM, then we will gather it's baseline output by LLMs' first output token and our RTD result through searching our reference datastore. λ is set to 1 in this task. How the reference datastore is generated can be fount at appendix B.1.

Models we used are: LLaMA2-7B and 70B [39], LLaMA3-8B [8], MPT-7B [37], GLM3-6B [47] [7], Yi-34B. Includes model size from 6B to 70B, as most of the major current models are. We use the *base* version of the model by default. Testing benchmarks are: Massive Multitask Language Understanding (MMLU) [15], AI2 Reasoning Challenge (ARC, both Easy (E) and Challenge (C) parts) [4], Reasoning about Physical Commonsense in Natural Language (PIQA) [5], Open Book Question Answering (OBQA) [30], and Massive Multitask Language Understanding in Chinese (CMMLU) [25]. C-MMLU is a Chinese benchmark, so only Chinese models, GLM3 and Yi, participated in this benchmark.

The multiple-choice benchmarks we chose is challenging enough in itself and requires strong reasoning ability from the model; moreover, the answer format is fixed, which can simultaneously detect the ability to follow instructions. Since that most tasks in the traditional NLP field can be quickly converted into tasks of choosing one from several categories, even some generative tasks, so the results on the multiple-choice test can also represent many other tasks.

The performance boost can be found both with or without ICL. Results are in table 2. Besides testing scores, we also record the confused rate of baseline, the proportion of the questions that failed to be answered properly, including output irrelevant text or can't give a certain answer, in table 3. Meanwhile RTD is designed to given the LLMs' decision in a trustable and controllable way. In comparison with fine-tuning methods in table 4, we can notice that RTD can achieve approximate performance improvements as using PEFT methods like LoRA. Although it is still insufficient compared to full-parameter fine-tuning, the latter has a higher cost and has undergone knowledge injection (which is not considered in this part of the experiment). The dataset used for full-parameter fine-tuning is MMLU-Recall [32; 33], and the hyper-parameters of LoRA can be found in Appendix D. Moreover, we've tested obqa score with different source of reference library, testing the generalization ability of RTD, as shown in Table 5, RTD yields satisfactory results. We've also tested the performance of RTD with different λ for language understanding, shown in Table 6. Lastly, we've tested the iteration speed of these benchmarks, as shown in Table 7, the efficiency impact of RTD is minimized comparing to ICL.

Model	Benchmark	Baseline	5-shot ICL	RTD (Δ)	5-shot RTD (Δ)
LLaMA2-7B	MMLU	43.8	45.8	45.1 (1.3↑)	**47.2** (2.1↑)
	ARC (E & C)	30.1	65.0	41.4 (11.3↑)	**67.3** (2.3↑)
	PIQA	56.5	62.1	71.4 (14.9↑)	**73.2** (11.1↑)
	Openbook QA	27.8	51.0	30.4 (2.6↑)	**53.6** (2.6↑)
LLaMA2-70B	MMLU	56.7	67.9	56.9 (0.2↑)	**68.5** (0.6↑)
	ARC (E & C)	67.4	91.6	86.1 (19.7↑)	**91.7** (0.1↑)
	PIQA	72.3	85.3	81.9 (9.6↑)	**86.6** (1.3↑)
	OpenbookQA	53.7	84.4	68.2 (14.5↑)	**85.4** (1.0↑)
LLaMA3-8B	MMLU	47.5	**63.9**	57.2 (9.7↑)	61.9 (2.0↓)
	ARC (E & C)	71.2	**87.3**	83.7 (12.5↑)	87.1 (0.2↓)
	PIQA	69.9	78.9	76.3 (6.4↑)	**80.0** (1.1↑)
	OpenbookQA	53.3	77.5	71.4(18.1↑)	**78.6** (1.1↑)
MPT-7B	MMLU	27.4	29.6	**30.4** (3.0↑)	29.8 (0.2↑)
	ARC (E & C)	27.5	failed	27.6 (0.1↑)	**30.1**
	OpenbookQA	29.4	failed	27.2 (2.2↓)	**30.4**
GLM3-6B	MMLU	41.9	48.6	47.6 (5.7↑)	**49.8** (1.2↑)
	ARC (E & C)	59.1	75.3	75.0 (15.9↑)	**76.5** (1.2↑)
	PIQA	66.8	73.6	**75.9** (9.1↑)	74.5 (0.9↑)
	OpenbookQA	55.1	67.1	64.0 (8.9↑)	**68.8** (1.7↑)
	C-MMLU	48.8	54.5	53.3 (4.5↑)	**54.7** (0.2↑)
Yi-34B	MMLU	68.6	**74.3**	70.3 (1.7↑)	73.3 (1.0↓)
	ARC (E & C)	93.3	94.0	90.7 (2.6↓)	**94.6** (0.6↑)
	PIQA	88.3	83.5	**88.4** (0.1↑)	87.7 (4.2↑)
	OpenbookQA	83.5	**89.8**	88.4 (0.9↑)	88.8 (1.0↓)
	C-MMLU	70.3	81.0	73.9 (3.6↑)	**81.8** (0.8↑)
Avg	-	56.41	65.28	63.31	**68.88**

Table 2: RTD on language understanding benches. Baseline refers to zero-shot performance. ICL exceeds MPT-7B's 2048 context window, with a 0 score result, recorded as failed in the table.

Table 3: Confused rate.

Model	Llama2-7B	GLM3-6B	Yi-34B
Rate	8.6%	11.81%	0.44%

Table 4: RTD comparing with fine-tune methods.

Methods	baseline	LoRA	FT	RTD
Score	41.9	42.5	46.31	42.8

4.2 Language Generation

Reasoning with Context Generative tasks are generally subjective and difficult to test. We constructed a benchmark based on Retrieval-Augmented Generation (RAG) and Open Book Question Answering [30] to test the potential of RTD in areas requires advance reasoning such as knowledge injection. Chain-of-Thought [42] is a method that encourage the model to provide a step-by-step analysis before giving the final answer, thereby enhancing the model's capabilities. We compared the performance of the model when introducing references through the ICL method and the RTD method, to determine the effectiveness of the RTD method. The extra knowledge source was Wikipedia. The generation of the datastore can be found in detailed in Appendix B.2. With the results of table 8, it can be seen that RTD was indeed helpful in knowledge injection. Besides, the context length is shrunk by a lot, thus saves reasoning GPU time and memory consumption. A detailed exploration of why RAG score is lower than baseline can be find in Appendix C.

Style transfer To explore whether the RTD method can be used to modify the language style of the model, we designed a style transfer experiment. We used a moderately scaled and strongly styled dataset, Tiny-Shakespeare [21; 40], and compared the perplexity (PPL) of the model on the test set after LoRA and RTD, to measure whether our method can help the model change the output style.

Table 5: Generalization of RTD.

Source	OBQA	ARC	MMLU
OBQA	71.4	71.4	71.2

Table 6: Different λ in Language Understanding

λ	1	0.8	0.6	0.4	0.2	0
OBQA	71.4	68.0	67.0	66.8	66.6	53.3

Table 7: Efficiency of RTD.

Methods	baseline	RTD	ICL	ICL + RTD
Speed(it/s)	25.1	23.6	7.90	7.85
Extra Memory Usage (MB)	0	$\tilde{1}6$	$\tilde{3}7$	$\tilde{5}2$

The results in Table 9 prove that our RTD method can reduce the perplexity of the model, enabling the model to adapt to the style of different datasets. The hyperparameters of LoRA are in Appendix D.

4.3 Influence of Hyper-parameters in RTD

Although our method is quick and efficient, it still introduces several hyper-parameters. We hope to explore the relationship between these hyper-parameters and the final performance of RTD. We conducted a series of ablation experiments on LLaMA2-7B [39] and OBQA [30] to explore the impact of different hyper-parameters on performance and how to quickly determine the optimal hyper-parameters. The overall result can be found in Figure 6. If not tuned, we set $k = 1024$, $s_{\mathcal{L}} = 19,828$, $\lambda = 1$ and $T = 750$ by default.

Depicted in Figure 6 (a), RTD's performance improves initially with increasing $s_{\mathcal{L}}$ but eventually maxed out and starts oscillating when $s_{\mathcal{L}}$ reaches 4096. Generally, a larger $s_{\mathcal{L}}$ gives a better performance, but it do get maxed out depends on the specific task. Figure 6 (b) showcased us how RTD's performance consistently improves as k increases initially, but eventually reaches a plateau, similiar with the $s_{\mathcal{L}}$. To be denoted is that a larger k could harm efficiency. Figure 6 (c) implies that RTD can only reach it's best performance when T is large enough. Empirically, due to the characteristics of the exponential function, as long as the range of scaled distances d'' is kept between 1-2, a sufficiently good effect can be achieved. In RTD, λ is an important variable, especially in generation tasks. However, λ does not require high precision, and the range is relatively limited, so a good enough effect can be achieved quickly through a few attempts. Empirically speaking, 0.4-0.7 is a suitable range for λ. Previous studies indicated that by pruning the dimension of attention won't hurt $k$nn algorithm's performance [13]. In the case of RTD, showcased in Figure 6 (d), it can be found that the performance won't drop with at least $\frac{1}{4}$ heads remained, and the generation speed was boosted as more heads are dropped.

5 Conclusions

In this paper, we introduce Reference Trustable Decoding, a novel training-free method designed to augment Large Language Models in downstream tasks. RTD refines the output distribution by leveraging references retrieved from a specially curated datastore, as a bypass of conventional `LM_Head`. Our experimental results demonstrate RTD achieved superior performance compared to the In-Context Learning baseline in 21 out of 25 different dataset and model configurations as well as fine-tune based methods. This result highlights the effectiveness of RTD across a diverse range of scenarios, underscoring its potential as a robust solution for enhancing language model capabilities in downstream tasks.

Limitations & Future Work

RTD is an efficient and quick method to augment the capabilities of models on specific downstream tasks. However, for some tasks, especially generative tasks, the large reference datastores that are difficult to directly compress may pose challenges for applications. Nevertheless, we believe that there is likely inherent redundancy in such large datastores. We hope to enable machines to identify these redundancies while maintaining a gradient-free method, in order to achieve efficient fine-tuning.

Table 8: Comparison of RTD and RAG using Wikipedia on LLaMA2-7B-Chat.

LLaMA2-7B-Chat	Acc	Latency (ms)
Baseline	39.0	**42.5**
Wiki RAG	29.0	> 200
Wiki RTD	**44.4**	46.5

Table 9: PPL of the fitted model on domain datasets.

Dataset	Baseline	LoRA	RTD
Tiny-S	1.6982	1.3710	1.4501

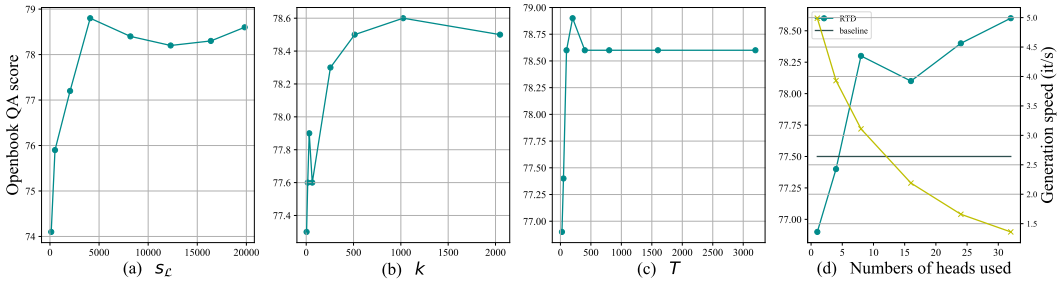

Figure 6: Hyper-parameters' influence on RTD's performance

How to make RTD accomplish tasks with high quality while being space-efficient is our following research direction.

Acknowledgments

We sincerely appreciate the valuable feedback provided by all reviewers during the review process, as well as the efforts of the area chairs. This work was supported by the National Natural Science Foundation of China (No. 62306216), the Natural Science Foundation of Hubei Province of China (No. 2023AFB816), the Fundamental Research Funds for the Central Universities (No. 2042023kf0133).

References

[1] Mistral AI. Mistral 7b, the best 7b model to date, apache 2.0, 2023.

[2] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4895–4901. Association for Computational Linguistics, 2023.

[3] Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian Khabsa. The belebele benchmark: a parallel reading comprehension dataset in 122 language variants. *arXiv preprint arXiv:2308.16884*, 2023.

[4] Sumithra Bhakthavatsalam, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, and Peter Clark. Think you have solved direct-answer question answering? try arc-da, the direct-answer AI2 reasoning challenge. *CoRR*, abs/2102.03315, 2021.

[5] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott

Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[7] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335, 2022.

[8] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

[9] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.

[10] Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O.K. Li. Search engine guided neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[11] Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang, and Han Xiao. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents, 2023.

[12] Kazuma Hashimoto, Raffaella Buschiazzo, James Bradbury, Teresa Marshall, Richard Socher, and Caiming Xiong. A high-quality multilingual dataset for structured documentation translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 116–127, Florence, Italy, August 2019. Association for Computational Linguistics.

[13] Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Efficient nearest neighbor language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 5703–5714. Association for Computational Linguistics, 2021.

[14] Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. On the effectiveness of adapter-based tuning for pretrained language model adaptation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2208–2222. Association for Computational Linguistics, 2021.

[15] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International*

Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021.

[16] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[17] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems*, 2023.

[18] Gautier Izacard, Patrick S. H. Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24:251:1–251:43, 2023.

[19] Imran Khan Mohd Jais, Amelia Ritahani Ismail, and Syed Qamrun Nisa. Adam optimization algorithm for wide and deep neural network. *Knowledge Engineering and Data Science*, 2(1):41–46, 2019.

[20] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[21] Andrej Karpathy. char-rnn. `https://github.com/karpathy/char-rnn`, 2015.

[22] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274, 2023.

[23] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*, 2020.

[24] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[25] Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. CMMLU: measuring massive multitask language understanding in chinese. *CoRR*, abs/2306.09212, 2023.

[26] Shangqing Liu, Yu Chen, Xiaofei Xie, Jing Kai Siow, and Yang Liu. Retrieval-augmented generation for code summarization via hybrid GNN. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[27] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, 2022.

[28] Xuan Liu, Jie Zhang, Song Guo, Haoyang Shang, Chengxu Yang, and Quanyan Zhu. Exploring prosocial irrationality for LLM agents: A social cognition view. *CoRR*, abs/2405.14744, 2024.

[29] Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. Generation-augmented retrieval for open-domain question answering. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting*

of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pages 4089–4100. Association for Computational Linguistics, 2021.

[30] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

[31] Feng Nie, Meixi Chen, Zhirui Zhang, and Xu Cheng. Improving few-shot performance of language models via nearest neighbor calibration. *arXiv preprint arXiv:2212.02216*, 2022.

[32] Liu Peng. llama2-7b-mmlu, 2023.

[33] Liu Peng. Mmlu-recall, employ mmlu (cmmlu) questions as initial seeds to retrieve related articles from multiple training data corpora., 2023.

[34] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[35] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 3784–3803. Association for Computational Linguistics, 2021.

[36] Weixi Song, Zuchao Li, Lefei Zhang, Hai Zhao, and Bo Du. Sparse is enough in fine-tuning pre-trained large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[37] MosaicML NLP Team. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. Accessed: 2023-05-05.

[38] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

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

[40] Trelis. tiny-shakespeare. `https://huggingface.co/datasets/Trelis/tiny-shakespeare`, 2023.

[41] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022, 2022.

[42] Jason Wei and Xuezhi Wang. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.

[43] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[44] Jun Cheng Yang, Zuchao Li, Shuai Xie, Wei Yu, Shijun Li, and Bo Du. Soft-prompting with graph-of-thought for multi-modal representation learning. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 15024–15036, Torino, Italia, May 2024. ELRA and ICCL.

[45] Yao Yao, Zuchao Li, and Hai Zhao. SirLLM: Streaming infinite retentive LLM. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2611–2624, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[46] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 521–538, Carlsbad, CA, July 2022. USENIX Association.

[47] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Zhiyuan Liu, Peng Zhang, Yuxiao Dong, and Jie Tang. GLM-130B: an open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[48] Hongyi Zhang, Zuchao Li, Ping Wang, and Hai Zhao. Selective prefix tuning for pre-trained language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics ACL 2024*, pages 2806–2813, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics.

[49] Yixuan Zhang and Haonan Li. Can large langauge model comprehend ancient chinese? A preliminary test on ACLUE. In Adam Anderson, Shai Gordin, Bin Li, Yudong Liu, and Marco Carlo Passarotti, editors, *Proceedings of the Ancient Language Processing Workshop, ALP@RANLP 2023, Varna, Bulgaria, 8 September, 2023*, pages 80–87. INCOMA Ltd., Shoumen, Bulgaria / ACL, 2023.

[50] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren's song in the ai ocean: A survey on hallucination in large language models, 2023.

[51] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Appendix

A Testing Environments

All testing are done on a server with 8*A100 80G SXM. For models with less than 15B parameters, 2 of 8 GPUs are used. For models with more than 15B parameters, 4 of 8 GPUs are used. All testing are carried out under HuggingFace Transformers library [43].

B Generation of Reference Datastore

B.1 Benchmark Testing

To generate reference datastores, LLMs are shown to the questions and options in the training split of the benchmarks and we store the attention output. For each question this process is repeated four times cycling through A, B, C, D as the correct value. $3,500$ to $5,000$ question is shown to the LLMs and about $20,000$ (k, v) entries are generated. To be noted is that the reference datastore of CMMLU is generated from validation set of C-eval [17], split *zho_Hans* of belebele [3] and testing set of ACLUE [49] since there is no training split for the benchmark.

B.2 Wikipedia Fact Retrieval

For our reference datastore, we encoded all of the Wikipedia sentences using the Jina [11] model, which is smaller in both it's parameter count and hidden size, resulting in a faster generation speed and smaller space cost for encoded vector datastore. Every usable sentence in Wikipedia is encoded, meanwhile the sentences from the same page share a same value, which is the no. of this page. When testing, we use the same model to encode the question, then we search the most relevant pages in the datastore, be the metric of cosine similarity, to retrieve the most relevant pages. In this section, $s_{\mathcal{L}}$ is same as the sentences count of Wikipedia, around 73M. $k = 1024$. T and λ are not applicable here.

With retrieved pages, we generate a reference datastore with every sentence in the pages. We first calculate attention representations for every token, whose corresponding value is the id of next token, *eos* for the last token. Then we use this dynamically generated reference datastore for following RTD. In this section, $s_{\mathcal{L}}$ is the same as the length of tokenized sequence, 6200 on average, and we use $T = 750, k = 1024, \lambda = 0.4$.

C RAG's Deficiency in Testing

RAG method's shows a decline in performance in Table 8. To explain this, we can further examine the average length of the tokenized sequences of the retrieved context, which is around 6200, showcased in Table 10. This length will hardly increase any inference cost for the RTD method, due to the small $s_{\mathcal{L}}$, but it exceeds the pre-training sequence length of LLaMA2-7B-Chat, which is 4096. That is to say, the naive RAG method here will cause sequence length overflow, thereby significantly affecting performance. If the overflow happened, then the model's ability is cut down significantly.

D LoRA Hyperparamters

See Table 11. For LoRA tuning on MMLU, any question whoes tokenized length exceed 4096 was evicted from both training and testing. The maximum tokenized length of the Tiny-Shakespeare dataset is 900.

Table 10: Average length by token in OBQA question answering process, split by sections.

Section	Average Length
Wikipedia Context	6192
Question	84
Response	231

Table 11: LoRA Hyper-parameters

Hyper-parameter	Value
Batch Size	4
Epochs	2
Max Seq. Len.	4096
LoRA Target	{Q, K, V, O, Up, Down, Gate}_proj
LoRA Rank	16
LoRA α	32
LoRA dropout	0.01
Learning Rate	1e-5
Optimizer	AdamW
Adma RMS ϵ	2e-4
Adam β	$(0.9, 0.999)$
Adam Weight Decay	0.01
Scheduler	Constant LR

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Section 3 and section 4 reflects our main claim.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Our main theoretical result is about efficiencies in section 3.4, in which they were proved.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The detailed description of our experiments and hyper-parameters can be found in section 4 and appendix D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case

of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Our main codes can be found in Supplementary Material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Discussions of hyper-parameters of our methods can be found in section 4.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The experimental results are definitive and do not involve any random factors.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All information above are given in section 4 and with specific experiment focused on some of them.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: [NA]

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All models and datasets we've used have been cited properly in section 4.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Our main codes, as assets, are updated in Supplementary Material.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.