Chain-of-Rank: Enhancing Large Language Models for Domain-Specific RAG in Edge Device

Anonymous ACL submission

Abstract

Retrieval-augmented generation (RAG) with large language models (LLMs) is especially valuable in specialized domains, where precision is critical. To more specialize the LLMs into a target domain, domain-specific RAG has recently been developed by allowing the LLM to access the target domain early via finetuning. The domain-specific RAG makes more sense in resource-constrained environments like edge devices, as they should perform a specific task (e.g. personalization) reliably using only small-scale LLMs. While the domain-specific RAG is well-aligned with edge devices in this respect, it often relies on widely-used reasoning techniques like chain-of-thought (CoT). The reasoning step is useful to understand the given external knowledge, and yet it is computationally expensive and difficult for small-scale LLMs to learn it. Tackling this, we propose the Chain of Rank (CoR) which shifts the focus from intricate lengthy reasoning to simple ranking of the reliability of input external documents. Then, CoR reduces computational complexity while maintaining high accuracy, making it particularly suited for resource-constrained environments. We attain the state-of-the-art (SOTA) results in benchmarks, and analyze its efficacy.

1 Introduction

The integration of retrieval-augmented generation (RAG) with large language models (LLMs) (Lewis et al., 2020) has emerged as a pivotal advancement in mitigating the issue of factual hallucination (Ji et al., 2023)—an inherent limitation of LLMs when generating knowledge-intensive responses. By leveraging external knowledge sources, RAG enables LLMs to utilize relevant knowledge dynamically, enhancing both the accuracy and reliability of their outputs.

RAG is especially crucial in the context of specialized domains, where precision is paramount and errors can be costly. Also, in RAG, LLMs must not only incorporate the relevant external information as the input, but also contextualize the information within the nuances of the target domain. To optimize the RAG-LLM for a specific domain, recently domain-specific RAG (Tianjun Zhang, 2024) has been developed where LLMs can early access the target domain through finetuning. The practicality of the domain-specific RAG is more noteworthy when computational resources are limited such as edge devices since with only a small-scaled LLM some tasks should be performed reliably.

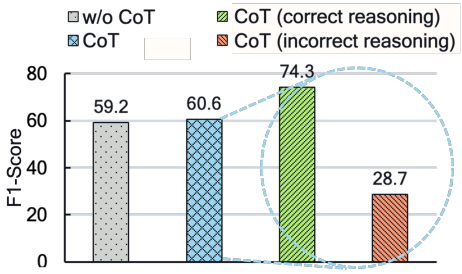

Figure 1: **Performance for domain-specific RAG on HotPotQA dataset on LLaMA3-8B with LoRA adapter.** The marginal effect of CoT (59.2% → 60.6%) is because of the generated incorrect reasoning which severely degrades the performance.

Despite the promise of the domain-specific RAG, the input external knowledge (generally retrieved information dubbed contexts) may consist of both irrelevant and relevant contexts. Hence, reasoning process such as chain-of-thought (CoT) (Wei et al., 2022) is useful for understanding and focusing on the relevant context. To this end, in RAFT (Tianjun Zhang, 2024), LLM learns the reasoning as well as answering in finetuning. Also, in (Yu et al., 2023), the reasoning is prompted to generate a summary of all the contexts. Elaborated reasoning is beneficial, and yet obtaining this kind of reasoning dataset for domain-specific learning is time-consuming and costful, and also it incurs a large testing cost.

Unreliability of reasoning is also a critical issue,

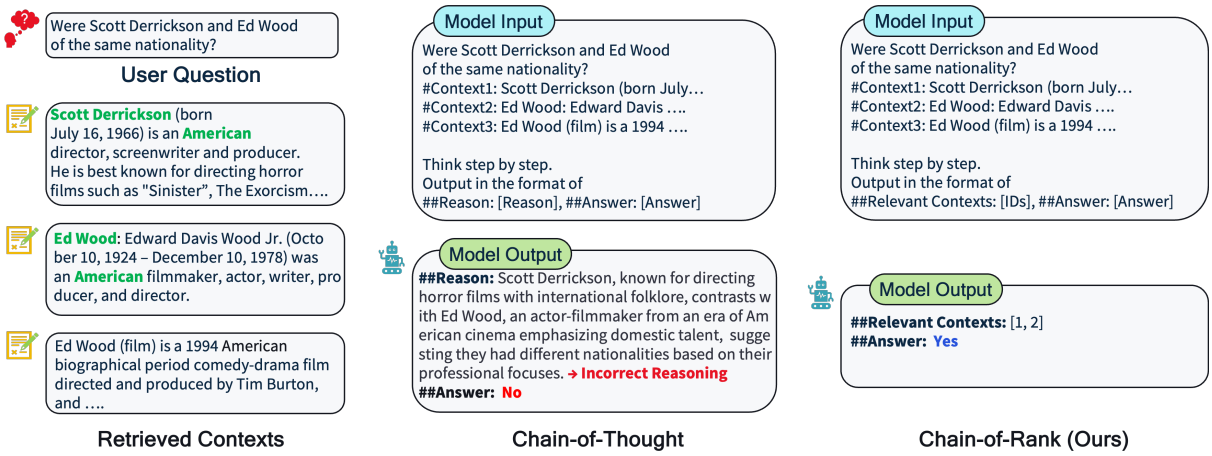

Figure 2: **Illustration of the proposed chain-of-rank for domain-specific RAG.** CoR streamlines the reasoning step, which is easier to be learned.

especially when the parameter-efficient fine-tuning (PEFT) like LoRA adapters (Hu et al., 2021; Huang et al., 2023; Bang et al., 2024) are used to reduce the computational burden in training resource constraint environment. Namely, the PEFT adapters are efficient but lack enough learning capacities, and then struggle to learn the intricate reasoning process. As shown in Fig. 1, LLaMA3-8B (Dubey et al., 2024) with LoRA exhibits marginal gains when learned with CoT. It means that the intricate reasoning can become a hindrance (Shi et al., 2023) in resource-constrained domain-specific RAG.

Instead of focusing the intricate reasoning processes, we narrow the focus to the ranking of the contexts' relevance, then the LLM can streamline its reasoning and reduce the cognitive load required to generate accurate final answers. Building on this insight, we propose the Chain-of-Rank (CoR) approach. In this method, the model is learned to output just the ID of the contexts which are relevant to the query, and the answer. Then, CoR not only reduces computational complexity but also enables the LLM to concentrate more fully on the critical information, leading to more accurate and domain-specific outputs. This focus on relevance rather than elaborate reasoning aligns well with the resource limitations of small-scale LLMs and edge devices.

2 Related Works

Domain-specific RAG. In the existing training-based RAG (Lin et al., 2024; Wang et al., 2024; Asai et al., 2023), the LLM is learned for various domains, and then applied to unseen domains. However, for better contextualization or under constrained resource condition, it is beneficial for LLM to be early accessed to the target domain via training on the domain. To this end, RAFT (Tianjun Zhang, 2024) pioneered domain-specific RAG. In RAFT, the LLM is learned by alternating two loss functions which are designed to simulate open-book and closed-book cases, respectively. The first loss addresses both distracting and golden contexts, while the second loss does only distracting ones. However, for decent performance evenly across various datasets, they trained the LLM to learn how to make the intricate reasoning as well as the answer.

Reasoning techniques in LLM. CoT (Wei et al., 2022) reasoning has been shown to enhance performance in LLMs, sparking numerous studies aimed at improving its efficiency. To delve into CoT, more complex approaches like CoT decoding by sampling (Wang et al., 2023; Wang and Zhou, 2024) and analogous reasoning (Yasunaga et al., 2023) have emerged. Considering the lengthy inputs which contain the retrieved contexts as well as the query in RAG, sampling to find the optimal decoding path or generating reasoning examples by itself are too burden in terms of computational cost. Tailored for RAG, methods like RAFT (Tianjun Zhang, 2024) and CoN (Yu et al., 2023) have demonstrated the effectiveness of reasoning in RAG. However, as highlighted in (Shi et al., 2023), errors in reasoning can lead to incorrect answers. When using low-capability LLMs to learn both reasoning and answering, these errors become more pronounced. Since retrieved contexts contain factual knowledge, focusing on simpler and more efficient reasoning that just prioritizes relevant contexts can mitigate this issue, making the identification of relevant context alone sufficient.

3 Method

3.1 RAG Problem Set-up

In RAG, an LLM can be formalized as $p(y|x) = \sum_D p(y|x, D)p(D|x)$, where x denotes the input query, y represents the LLM generated answer, and $D = \{d^k\}_{i=1}^K$ contains the K individual contexts. This formulation takes into account the joint probability of retrieving a set of contexts, rather than assuming the contexts are selected independently.

As this sum over all the context sets is impractical, generally an off-the-shelf retriever selects the top-K most relevant contexts. This leads to the approximation: $p(y|x) \approx p(y|x, D)$. Furthermore, when a reasoning step is considered, it becomes as follows.

$$p(y|x) \approx \sum_R p(y|x, D, R)p(R|x, D) \quad (1)$$

where R represents the generated reasoning.

3.2 Chain-of-Rank

Framework. We streamline the reasoning process by shifting the focus from complex reasoning to identifying the IDs of the given contexts that correspond to the most relevant ones for x. With just this process, the model can reduce cognitive overhead on less relevant information, and more pay attention to the relevant information. As illustrated in Fig. 2, each context is identified by its unique ID. Then, CoR involves two main steps: (1) selecting the ID of relevant contexts (*i.e.,* R) and (2) generating the final answer y. Consequently, the CoR framework significantly simplifies the reasoning step, making it a practical solution for scenarios with limited computational resources while enhancing performance in domain-specific applications.

Model training. We concatenate the instruction, question, and retrieved documents into a single prompt, allowing the model to learn in a standard supervised manner. The model is trained to optimize both the selection of relevant document IDs and the accuracy of the generated answer by minimizing a joint loss function.

$$\mathcal{L} = -\sum_{i=1} \log p(y_i|x_i, D_i, R_i) - \log p(R_i|x_i, D_i) \quad (2)$$

We designed the top-k documents in D_i to include at least one positive document for a query x_i during training. Also, we employed LoRA to efficiently fine-tune the model parameters assuming the low resource constraint (details are in Appendix).

4 Experiments

We provide more details and analysis in Appendix.

Datasets. In our experiments, we use the following datasets to evaluate the proposed method. We selected these datasets to represent both popular and diverse domains including Wikipedia and Coding/API documents.

In specific, we select HotPotQA (Yang et al., 2018) and Gorilla API datasets (Patil et al., 2023). The HotPotQA is the open-domain question-answers based on Wikipedia, mainly focused on common knowledge. In testing, we used 'HotpotQA distractor dev. set' which is designed to provide ten contexts including at least one relevant context for a query. TensorFlow, HuggingFace, and TorchHub of the Gorilla API are to measure how to generate the correct, functional, and executable API calls based on the documentation. For each of HuggingFace, TorchHub, and TensorFlow, train and test splits are provided, which share the API pool. Also, following the officially-released code[1], we utilized BM25 retriever.

Evaluation. We set K as 10 for the HotPotQA, then all the available irrelevant contexts distract the relevant ones for a query. For the Gorilla API, K is set to 5. To minimize the influence of the quality of the off-the-shelf retriever, we set up our experiments to include at least one relevant context for each input query. For the HotPotQA, we used two standard metrics: Exact Match (EM) and F1 score, following prior work (Chen et al.; Karpukhin et al., 2020; Zhu et al., 2021). An answer is correct in the EM if its normalized form, based on (Karpukhin et al., 2020), matches exactly the ground-truth answer. F1 score calculates token overlap between the prediction and ground truth (Zhu et al., 2021). For the Gorilla API, following the official benchmark, we perform AST sub-tree matching to identify which API the LLM is calling by matching key arguments, and report AST accuracy.

Baselines. We consider the following baselines for our experiments based on LLaMA3-8B: Naive LLaMA3-8B, Domain-specific fine-tuning (DSF) without reasoning, RAFT (DSF with CoT) (Tianjun Zhang, 2024), DSF + CoN (chain-of-note) (Yu et al., 2023). In the DSF baselines, we commonly used the LoRA adapter. And, all the baselines are in the zero-shot setting. In addition, RAFT suggested to alternating two losses of the irrelevant contexts-only and the mixing irrelevant and rele-

[1]https://github.com/ShishirPatil/gorilla

	Method	HotPotQA		TensorFlow	HuggingFace	TorchHub
		EM	F1 score			
	LLaMA3-8B	40.84	52.47	32.11	10.14	22.13
Domain-specific	DSF	44.98	59.15	83.91	87.42	70.08
	RAFT (DSF-CoT)	46.79	60.59	88.98	89.68	74.05
	DSF-CoN	48.60	62.04	84.52	79.05	76.21
	DSF-CoR (Ours)	**49.23**	**64.11**	**95.68**	**92.52**	**80.54**

Table 1: **Comparative results on domain-specific RAG.** EM and F1 score for the HotPotQA, and AST sub-tree matching accuracy scores (%) for the Gorilla API (TensorFlow, HuggingFace, TorchHub) are reported.

Method	DSF-CoT	DSF-CoN	**DSF-CoR**
Reasoning Accuracy (↑)	68.21	69.02	72.31
Reasoning Tokens (↓)	90.15	143.18	8.00

Table 2: **Analyses on reasoning.** Accuracy (%) and cost (used tokens) for reasoning are on the HotPotQA.

vant contexts. For a fair comparison, we tried to find the optimal combination ratio of the two losses for this baseline.

4.1 Comparative Results

We evaluate our CoR and demonstrate the efficacy in Table 1. The non-specified pre-trained LLM (LLaMA3-8B) shows severely degraded scores in the API datasets than in the natural questions of HotPotQA, which proves the requirements and importance of domain-specific RAG. The reasoning-based methods, RAFT and CoN, attain better results than DSF. However, in F1 score, the effect of CoT is marginal. Also, although the noting strategy of CoN is tailored for RAG, it sometimes shows lower performance than the straightforward DSF as well as CoT (see TensorFlow and Huggingface results). Whereas, we see that the proposed CoR consistently and significantly outperforms the baselines in all the datasets. It means that learning the complex reasoning can be a burden to the PEFT on the smaller-scale LLM, and thus simply identifying the IDs of the relevant contexts is more beneficial. We also study the extension of the proposed CoR to domain-agnostic RAG in Appendix.

4.2 Analysis

Reasoning quality. In RAG, the reasoning can be utilized to support the answer. Therefore, the quality of reasoning is also substantial, then we quantitatively compare the reasoning of CoT, CoN, and our CoR. We evaluate CoT and CoN using a pre-trained LLM (e.g. GPT) in a massive scale. Since CoT and CoR produce lengthy reasoning that incorporates details of the retrieved contexts which may lead to some errors in detail. Hence, to ensure a fair comparison, we prompt the LLM evaluator to assess whether the reasoning is related with the relevant golden context. Nevertheless, in the top row of Table 2, the proposed CoR attains clearly higher reasoning accuracy.

Cost in reasoning. We also evaluate the proposed method in terms of the cost for reasoning. To this end, we compare CoT, CoN, and our CoR according to the number of the tokens used for reasoning. As shown in the bottom row of Table 2, CoR uses significantly lower tokens for reasoning, which shows the efficiency of the proposed approach.

Importance of correct ranking: To see this, we obtain the answer giving incorrect ranking for DSF-CoR. Despite domain-specific learning, it yields severely degraded results, 24.20% EM and 32.34% F1 score.

5 Conclusions

We proposed the Chain of Rank (CoR) to address the limitations of the existing intricate reasoning processes like chain-of-thought in training-based, domain-specific RAG. For domain-specific RAG training, annotation expense for the reasoning data is required. Also, especially in testing on smaller LLMs in resource-constrained environments, it poses challenges in terms of the accuracy as well as computational cost. We observed that the inaccurate reasoning adversely affect the quality of final answer. By shifting the focus from elaborate reasoning to a simplified ranking of the reliability of retrieved documents, CoR significantly reduced computational complexity while attaining higher accuracy. Our experimental results demonstrated that CoR achieves SOTA results on RAG benchmarks, confirming its effectiveness in improving the domain-specific RAG performance of small-scale LLMs.

6 Limitations

This work acknowledges the significance of reasoning in domain-specific RAG models and presents an efficient approach that reduces the need for complex training data labeling and significantly lowers reasoning costs during testing. However, we did not thoroughly investigate whether the proposed method would be equally effective in more general RAG frameworks that do not rely on task-specific training. That said, preliminary results presented in the appendix indicate the potential for success in general RAG settings, suggesting that this area warrants deeper exploration in future work. Therefore, our findings provide a promising foundation for future research.

7 Ethical Consideration

In the field of domain-specific RAG, if the applications involve sensitive areas such as personal information, special caution must be taken during the model training process to ensure privacy and data protection. Beyond this consideration, methodologically, our research focuses on improving the accuracy and efficiency of RAG in LLMs, we do not foresee any direct negative ethical concerns stemming from our contributions. Nonetheless, it is important to recognize that generative AI technologies, including those using LLMs, come with potential risks. As such, careful consideration of their broader ethical and societal implications is necessary when these systems are applied in the real world.

References

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*.

Jihwan Bang, Juntae Lee, Kyuhong Shim, Seunghan Yang, and Simyung Chang. 2024. Crayon: customized on-device llm via instant adapter blending and edge-server hybrid inference. In *The 62nd Annual Meeting of the Association for Computational Linguistics*.

D Chen, A Fisch, J Weston, and A Bordes. Reading wikipedia to answer open-domain questions. arxiv 2017. *arXiv preprint arXiv:1704.00051*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *ArXiv:2407.21783*.

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. *International Conference on Learning Representations*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

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. In *Proceedings on Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, et al. 2024. Ra-dit: Retrieval-augmented dual instruction tuning. In *Proceedings on International Conference on Learning Representations*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

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*.

Naman Jain Sheng Shen Matei Zaharia Ion Stoica Joseph E. Gonzale Tianjun Zhang, Shishir G Patil. 2024. Raft: adapting language model to domain specific rag. In *Conference on Language Modeling*.

Boxin Wang, Wei Ping, Lawrence McAfee, Peng Xu, Bo Li, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Instructretro: Instruction tuning post retrieval-augmented pretraining. In *Proceedings on International Conference on Machine Learning*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Xuezhi Wang and Denny Zhou. 2024. Chain-of-thought reasoning without prompting. *arXiv preprint arXiv:2402.10200*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Michihiro Yasunaga, Xinyun Chen, Yujia Li, Panupong Pasupat, Jure Leskovec, Percy Liang, Ed H Chi, and Denny Zhou. 2023. Large language models as analogical reasoners. In *The Twelfth International Conference on Learning Representations*.

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu. 2023. Chain-of-note: Enhancing robustness in retrieval-augmented language models. *ArXiv:2311.09210*.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*.

- Appendix -

Chain-of-Rank: Enhancing Large Language Models for Domain-Specific RAG in Edge Device

A Prompt template for chain-of-rank

Prompt template for domain-specific RAG with CoR

```
Contexts and Question are given.

Let's think step by step to make
correct output.

First, reranking goal: select the
relevant contexts, important to answer
the qeustion correctly.

Then, answering goal: Focusing on the
selected context, answer the question.


Question: {question}
Context1: {context_1}
Context2: {context_2}
...
ContextN: {context_N}

Output:
## Relevant Context ID: {IDs}
## Answer: {answer}
```

B Feasibility in task-agnostic RAG-LLM.

To identify the potential of the proposed chain-of-rank as the general reasoning technique in RAG beyond domain-specific RAG, we applied the proposed method on the pre-trained model (LLaMA3-8B). As shown in the Table 3, the CoR is comparable to CoT. Further, we combined the reasoning of the CoT and CoR. On top of the CoT-style reasoning, the reasoning of CoR makes meaningful synergistic results using a small cost.

C Datasets

HotPotQA. HotPotQA is a large-scale question-answering dataset designed to evaluate both factual reasoning and multi-hop question answering. The training set with approximately 90,000 examples and development (dev.) set containing around 7,400 examples. For each question, ten contexts are provided where a context consists of several sentences and the key sentences (supporting facts to the query) are annotated. We experimented with the whole sentences in every context for a challenging set-up.

Method	EM	F1 Score
LLaMA3-8B pre-trained	40.84	52.47
+ CoT	42.41	53.96
+ CoR	41.20	55.92
+ CoR&CoT	44.15	58.09

Table 3: **Results on a pre-trained LLM by applying the CoT, the proposed CoR, the mixture of CoT and CoR.** EM and F1 score are reported on the HotPotQA dataset.

Gorilla API. Gorilla API is multi-faceted, comprising three domains: TensorFlow, HuggingFace, TorchHub where training data includes 6190, 8191, 337 entries and testing data does 688, 911, 186 entries, respectively. Each entry of a domain conveys a detailed description for an API call. In specific, it consists of the following fields: {domain, framework, functionality, api_name, api_call, api_arguments, environment_requirements, example_code, performance, and description}.

Reasoning dataset for baseline training. Gorilla API dataset provides the explanation for every API document, and hence we use that as the reasoning following (Tianjun Zhang, 2024) for domain-specific training. In HotPotQA dataset which does not include reasoning, we utilized a significant-scale LLM to generate the intricate reasoning dataset for domain-specific training. We used the prompt of (Tianjun Zhang, 2024) to generate the reasoning.

D Evaluation Metric

Exact Match. Exact Match (EM) evaluates whether the model's generated response is identical to the ground truth answer. It is computed as the percentage of predictions where the generated output exactly matches the reference answer, including the order and wording. EM is strict, meaning any deviation results in a score of 0 for that prediction, and only exact matches count as correct.

F1 score. F1 score is a measure that balances precision and recall. It is computed by comparing the overlap of tokens between the generated response and the ground truth. Precision is the ratio of correct tokens in the generated response to the total

number of tokens in the response, while recall is the ratio of correct tokens to the total number of tokens in the ground truth. The F1 score is the harmonic mean of precision and recall, allowing for partial credit when the generated answer partially overlaps with the correct answer.

AST accuracy. AST (Abstract Syntax Tree) accuracy is a metric used to evaluate the correctness of generated API calls by comparing their structural representation to reference APIs. The generated API call is parsed into an AST, and its structure is matched against the corresponding reference API from the dataset. The accuracy is determined by how well the generated API's function names and key arguments align with the reference. If the AST of the generated call matches a subtree of the reference API, it is considered correct.

E Prompt template to evaluate the reasoning

Prompt template to evaluate reasoning

```
You are an expert at evaluating
reasoning based on provided
information. Given a question, five
retrieved contexts, and reasoning,
your task is to determine whether
the reasoning is based on the correct
context. The correct context is the
one that contains the most relevant
and accurate information to answer
the question.

Follow these steps:
1. Identify which retrieved context
contains the most accurate information
to answer the question (the "golden
context").
2. Evaluate if the reasoning is based
primarily on this golden context.
3. Provide a clear answer (Yes or No).

### Question:
{question}

### Retrieved Contexts:
1. {context_1}
2. {context_2}
3. {context_3}
4. {context_4}
5. {context_5}

### Reasoning:
{reasoning}

### Evaluation:
Is the reasoning based on the correct
context? Answer with "Yes" or "No".
```

F LoRA-based training details

We implemented the proposed and baseline approaches based on the Huggingface PEFT library (Mangrulkar et al., 2022). We set the rank r and scaling factor of a LoRA as 128 and 16, respectively. In training, we use the AdamW optimizer with a learning rate 0.0003 which is cosine annealed. We also set the batch size as 128 and the maximum epoch as 1. All the proposed and baseline methods are implemented with PyTorch 2.0.1 and executed on two NVIDIA A5000 GPUs.