An Expert is Worth One Token: Synergizing Multiple Expert LLMs as Generalist via Expert Token Routing

Ziwei Chai[1,2], Guoyin Wang[3], Jing Su[3], Tianjie Zhang[1], Xuanwen Huang[1], Xuwu Wang[3]
JingJing Xu[3], Jianbo Yuan[3], Hongxia Yang[3], Wu Fei[1], Yang Yang[1]*
[1] Zhejiang University,
[2] Beijing Huairou Laboratory,
[3] ByteDance Inc.
{zwchai, yangya}@zju.edu.cn

Abstract

We present Expert-Token-Routing, a unified generalist framework that facilitates seamless integration of multiple expert LLMs. Our framework represents expert LLMs as special expert tokens within the vocabulary of a meta LLM. The meta LLM can route to an expert LLM like generating new tokens. Expert-Token-Routing not only supports learning the implicit expertise of expert LLMs from existing instruction dataset but also allows for dynamic extension of new expert LLMs in a plug-and-play manner. It also conceals the detailed collaboration process from the user's perspective, facilitating interaction as though it were a singular LLM. Our framework outperforms various existing multi-LLM collaboration paradigms across benchmarks that incorporate six diverse expert domains, demonstrating effectiveness and robustness in building generalist LLM system via synergizing multiple expert LLMs. [1]

1 Introduction

Large language models (LLMs)—notably, GPT-4 and LLaMA (Touvron et al., 2023)—have demonstrated remarkable capabilities across a wide spectrum of tasks. However, their performance and reliability in certain specialized domains sometimes still fall short of expectations (Wang et al., 2023; Zhao et al., 2024). Motivated by this, the recent year has seen emergence in the development of "expert" LLMs, with a particular focus on tailoring a general LLM to excel in specific domains or tasks. For example, there are efforts to refine the pretraining corpus to develop expert LLMs with enhanced capabilities in specific domains, such as CodeLlama (Rozière et al., 2024) and DeepSeek-Math (Shao et al., 2024). Furthermore, applying continual training or supervised fine-tuning (SFT) on specially selected corpora can lead to significant enhancements on targeted tasks like data analysis (Li et al., 2023b; Hu et al., 2024) and financial analysis (Chen et al., 2023), making a general LLM into a specialized expert.

In this study, we focus on the following specific research question: *how can we synergize various expert LLMs into a singular generalist framework?* The practicality of this question lies in the fact that it's unrealistic to ask users to precisely switch between various expert LLMs, given the diversity and subtle difference in their capabilities. There are mainly two paradigm to build such a generalist LLM system, termed in this paper as the prompt-based methods and the router-based methods. Prompt-based methods (Li et al., 2023a; Wang et al., 2024; Suzgun and Kalai, 2024) treat expert LLMs as conversational agents. A "meta" LLM guided by crafted instruction communicates with expert LLMs to acquire domain-specific information. For the router-based methods (Jiang et al., 2023; Lu et al., 2023), a router model is trained to route each query to decide which expert is responsible to answer it.

However, both of the aforementioned paradigms have some notable limitations (Table 1). When employing the prompt-based method, successful expert collaboration significantly relies on how the instruction are crafted. For cases where the expertise of expert LLMs is implicit or undisclosed, pinpointing relevant expert knowledge through prompts could be inefficient. In the other hand, router-based methods requires training an extra router model tailored to the current expert LLM library, which needs to be retrained whenever a new expert LLM is added.

In this work, we introduce a novel technique termed Expert-Token-Routing (ETR). ETR employs a multi-expert LLM collaboration framework, wherein a meta LLM can seamlessly switch to a specific expert LLM. The core idea of ETR is encoding expert LLMs as special tokens within

* Corresponding authors.
[1]Codes are available at https://github.com/zjunet/ETR.

the vocabulary of the meta LLM. Inspired by ToolkenGPT (Hao et al., 2023), which captures tool semantics using special tokens, ETR employs expert tokens to characterize the expertise of different expert LLMs. Under this formulation, the meta LLM can route to an expert LLM like generating new tokens. This approach allows for expert routing without training extra router model, simply by appending trained expert tokens into a frozen meta LLM. To train the expert tokens, we construct an automated pipeline to collect expert queries from existing datasets. ETR can learn expert token embeddings from collected expert queries, which profiles the strengths of expert LLMs, thereby handling situations where the expertise of expert LLMs is implicit or undisclosed.

Our framework also supports a plug-and-play integration for new expert LLMs, allowing providers to add their expert LLMs and trained expert tokens to the framework as modular plug-ins, eliminating the need for modifications to any other part of the framework. Another appealing characteristic of our method is that it conceals the collaboration process between the meta LLM and expert LLMs from the user's perspective, facilitating interaction with the framework as though it were a singular LLM.

In our comprehensive experiments across six different domains, we compare ETR with the existing paradigms for expert LLM collaboration. Our findings indicate that ETR significantly outperforms the baselines as a generalist, with an overall performance improvement of 5.64%. It is worth noting that our framework also integrates the advantages of existing paradigms, as illustrated in Table 1.

The core contribution of this work is the introduction of a singular generalist framework that facilitates seamless integration and collaboration of expert LLMs. This framework not only supports learning the implicit expertise of expert LLMs from existing instruction dataset but also allows for dynamic extension of new expert LLMs in a plug-and-play manner. The effectiveness of our framework is showcased across benchmarks that incorporate six diverse expert domains.

2 Approach

In this section, we present ETR, which enables the integration of multiple expert LLMs into a singular generalist LLM framework for leveraging expert knowledge without the need for heavily fine-tuning or designing crafted instruction templates.

Table 1: Comparison of different expert LLM collaboration paradigms. Implicit Expertise means it's difficult to obtain a precise depiction of the expertise of an expert LLM. Dynamic Extension is about whether an expert LLM can be incorporated in a plug-and-play manner without making changes to the rest of the framework.

Expert Collaboration Paradigm	Extra Model Free	Implicit Expertise	Dynamic Extension	Transparent to Users
Prompt-based	✓	✗	✓	✗
Router-based	✗	✓	✗	✓
ETR (**Ours**)	✓	✓	✓	✓

Typically, an expert LLM ϵ is derived from base pre-trained LLMs through continual pretraining or supervised fine-tuning (SFT) on data of target domain. By careful data selection, an expert LLM of smaller scale can exceed a larger, general-purpose LLM in specific areas like coding (Guo et al., 2024) or math problem-solving (Shao et al., 2024).

Given an expert LLM library $E = \{\epsilon_1, \epsilon_2, \ldots\}$, our goal is to develop an LLM framework which can utilizing the expert knowledge available in the library E when solving problems within specific domains. To determine when and which expert to corporate with, we use a general-purpose LLM Ω as a meta LLM. And expert LLM ϵ_i is represented by a special token (expert token) witin the meta LLM's vocabulary.

2.1 Framework Overview

The core design of ETR is to encode expert LLMs as special tokens (termed "expert tokens") in the meta LLM's vocabulary. In this formulation, the expert routing is redefined within the LLM's standard next word prediction paradigm. An expert LLM is activated for decoding once its expert token is predicted as the next token by the meta LLM. The expert tokens are parameterized as an embedding matrix $W_E \in \mathbb{R}^{|E| \times d}$ and are appended to the language model head W_V of the meta LLM like regular word tokens.

Assuming we've already trained expert token embeddings W_E (to be described in Section 2.2), we will start by introducing how our framework works in inference. As shown in Figure 1, the meta LLM is served as the gateway for handling queries by default. And it is designed to treat word tokens and expert tokens uniformly. Specifically, the language modeling head of the meta LLM is a concatenation of original word token head and expert embeddings. Consequently, the meta LLM generates the response by predicting the next token

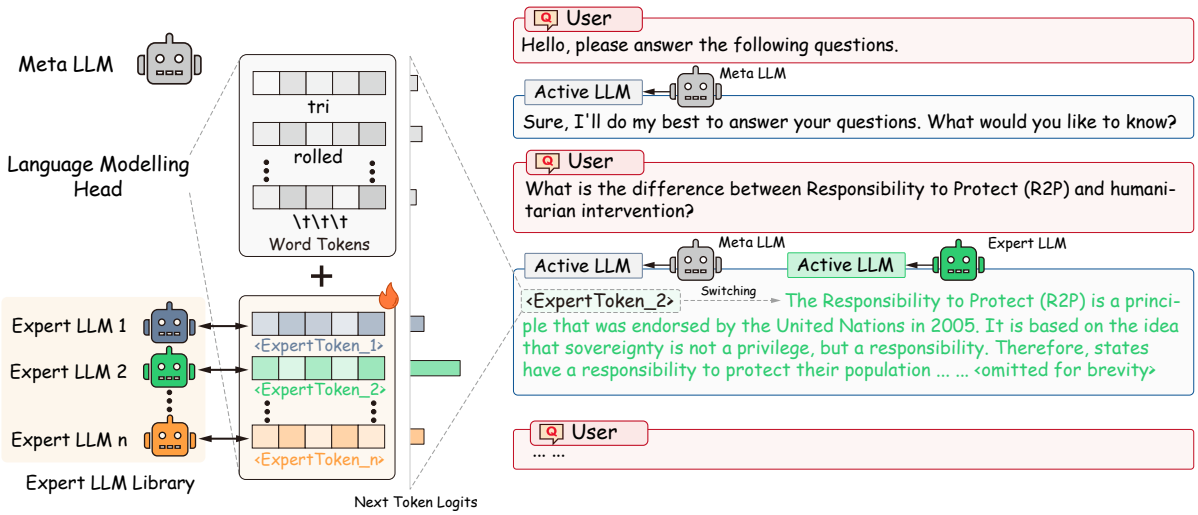

Figure 1: The framework of ETR. During the ETR's decoding process, the meta LLM serves as the default active LLM. When predicting the expert token, the active LLM switches to the corresponding expert LLM. The Expert tokens are appended to the frozen language modeling head of the meta LLM, where it is treated equally with word tokens during the next token prediction.

with the following probability:

$$P_\Omega(t_i|t_{<i}) = \text{softmax}([W_V; W_E] \cdot \mathbf{h}_{i-1}) \quad (1)$$

where $W_V \in \mathbb{R}^{|V|\times d}$ and $|V|$ is the vocabulary size of word tokens. Note that the next token can be either a word token or a expert token, i.e. $t_i \in V \cup E$.

Once an expert token is predicted, the meta LLM halts decoding. Subsequently, the corresponding expert LLM continues decoding, using the already generated context as its prefix. Algorithm 1 provides a detailed description of the inference procedure of ETR. Our framework operates as a singular LLM, making the underlying expert collaboration invisible to the user. As a result, interacting with our framework is indistinguishable from interacting with a general LLM.

2.2 Learning Expert Embeddings via Expert Query Set

Ideally, when a query is falls within the expertise of a specific expert, the corresponding expert token should be generated. Therefore, we hope that the expert embedding implicitly encodes the expert's strengths.

Drawing on the idea that experts outperform non-experts in addressing problems within their expertise area, we utilize an expert query set ϵ_i to learn an expert LLM's token embedding. Specifically, we consider a query-response pair that has significantly lower loss on the expert LLM ϵ_i compared to a general LLM to be ϵ_i's expert query. Given an instruction dataset D consisting of query-response pairs $\{(q_j, r_j)_{j=0,1,...}\}$, the loss of LLM ϵ on j-th query-response pair is defined as

$$L_\epsilon(j) = \sum_{x_k \in r_j} -log P_\epsilon(x_k|q_j) \quad (2)$$

where x_k is the tokens in the response. The instruction dataset used for collecting expert query set can be either existing instruction-tuning dataset (Taori et al., 2023) or synthetic data generated by LLMs

Algorithm 1 Inference procedure of ETR

Input: Query q, expert LLM Library $E = \{\epsilon_1, \epsilon_2, \ldots\}$, meta LLM Ω
Output: Response r
procedure GENERATE(q, E, Ω)
    Initialize context $c \leftarrow q$
    Initialize active LLM $\lambda \leftarrow \Omega$ ▷ Start with meta LLM
    while Response not complete **do**
        $t_i \leftarrow$ Predict next token with λ given c
        if t_i is a word token **then**
            Update context $c \leftarrow c \,||\, t_i$
        else if t_i is ϵ's expert token **then**
            $\lambda \leftarrow \epsilon_j$ ▷ Switch active LLM to the expert LLM
        end if
    end while
    $r \leftarrow c$
    return r
end procedure

For each expert LLM ϵ_i in the expert library, we build an expert query set $\mathcal{A}_i$ according to the following criteria:

$$\mathcal{A}_i = \{(q_j, r_j) | L_\Omega(j) - L_{\epsilon_i}(j) > \tau\} \quad (3)$$

where L_Ω is the loss of the meta LLM. Therefore, $\mathcal{A}_i$ is the collection of queries that the expert LLM ϵ_i adept at solving, outperforming the general-purpose meta LLM Ω. In the training process, queries in $\mathcal{A}_i$ serves as prefix and a special expert token $<$ `ExpertToken_i` $>$ is appended as ground truth for the next token prediction. Specifically, the training objective of ETR is:

$$\mathcal{L}(W_E) = \sum_{i}^{|E|} \sum_{q_j \in \mathcal{A}_i} -logP(\texttt{<ExpertToken\_i>} \ | q_j) \quad (4)$$

This embedding matrix W_E are the only tunable parameters. Therefore, similar to pioneering work on expanding the language modeling head of LLMs (such as ToolkenGPT (Hao et al., 2023)), training ETR doesn't require gradient propagation through the main part of the LLM parameters, leading to training that is both more stable and efficient than prompt tuning (Lester et al., 2021) or prefix tuning (Li and Liang, 2021). Thus, the GPU memory usage for expert token embedding tuning is nearly the same to that of LLM inference.

3 Experiments

3.1 Experimental Settings

Datasets We build a dataset comprising a mix of questions from six diverse expert domain. Specifically, we employ GPT-4 to rank the 57 subjects in MMLU (Hendrycks et al., 2021) across STEM, the humanities, the social sciences, and more, based on their level of specialization. We select the six highest-ranked subjects, which are astronomy, electrical engineering, security studies, prehistory, international law and human sexuality. Their test question sets are merged to form a dataset, named MMLU-Expert, which includes questions from six different specialized fields.

Expert LLM Construction via SFT Most previous research (Li et al., 2023a; Xu et al., 2023; Liu et al., 2023; Suzgun and Kalai, 2024) create expert LLMs by constructing specialized role prompts (i.e. mathematics, laywer). Although creating experts by role-playing is straightforward, it is difficult to effectively inject domain knowledge into an LLM through just several lines of expert description prompt. We believe that constructing expert LLMs through continual training/supervised fine-tuning is more aligned with real-world applications than the role-playing prompting approach.

Therefore, we collect a domain expert instruction dataset comprising knowledge across 6 different domains in MMLU-Expert. We synthesize 7.5K question-answer pairs for each domain, by harnessing unnatural instructions (Kervadec et al., 2023) to extract domain knowledge from GPT-4. Specifically, a seed question set is initialized using the MMLU validation questions. In each cycle, three questions are randomly selected from the seed question set, and GPT-4 is tasked with generating a question within the same domain. To avoid the risk of data leakage by GPT-4 reproducing MMLU-Expert test questions from its memory, We filter out questions which have a high similarity (measured by BERTScore (Zhang* et al., 2020) > 0.8) with test questions. We once again utilize GPT-4 to get answers for these generated questions. This approach allows extracting domain knowledge from GPT-4 into a question-and-answer dataset.

We then apply supervised fine-tuning on the synthetic dataset to get expert LLMs injected with domain knowledge, namely `Expert-DomainName`. Figure 2 shows the expert LLMs' performance across different domains of MMLU-Expert. Compared to non-expert models before fine-tuning, the expert models exhibit a significant performance gain of 17.90% on the in-domain test sets.

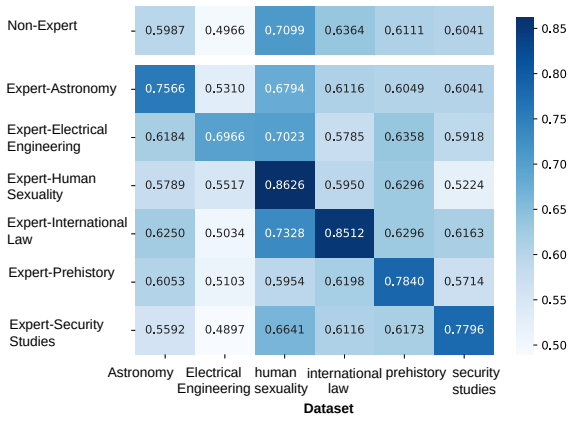

Figure 2: Expert LLMs' accuracy on MMLU-Expert.

Evaluation Metric With access to both an expert LLM library and a general LLM, the performance of a multi-expert LLM framework are evaluated through the overall accuracy on the MMLU-Expert. For each question, there exists an optimal expert in

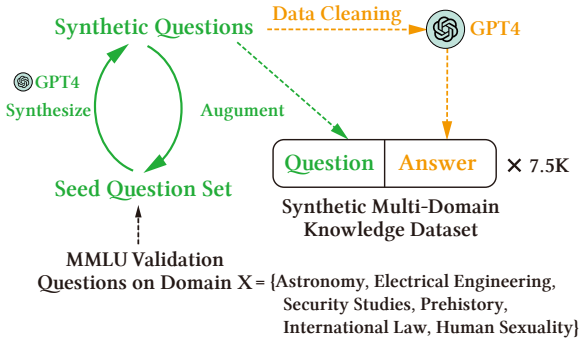

Figure 3: Collection process of multi-domain knowledge dataset synthesized through GPT-4. To prevent potential data leakage, synthetic questions with high BERTScore (Zhang* et al., 2020) to any question in the test set are filtered out.

the expert LLM library. The framework's capability to accurately route questions to these optimal experts serves as a critical measure.

Baselines We compare our framework with two types of baseline methods: the prompting-based expert collaboration methods and the router-based expert routing methods. It's worthy noting that in the existing prompting-based methods, as exemplified by Meta-Prompting (Suzgun and Kalai, 2024), there are two distinct components in its instruction: collaborating with experts and forming expert identity, both of which are crucial to the effectiveness of this method. However, in this paper's setting, an external expert LLM library are provided, removing the need to create experts via prompting. Therefore, we adapt the segment of the Meta-prompting instruction dedicated to expert LLM collaboration to serve as a comparative method, which we refer to as "Meta-prompting-E" in this paper. We also compare the multi-LLM debate (Du et al., 2023) method, as a specific case within prompting-based methods, where all the experts are invoked for all queries. In router-based methods, we compare with LLM-Blender (Jiang et al., 2023), which ranks the responses from each LLM. And we select the top-ranked result as the routing expert. In the original design of the LLM-Blender, it is also capable of fusing the top-k responses. This paper marks these two methods as LLM-Blender (Top 1) and LLM-Blender (Fused), respectively. It is noteworthy that Zooter (Lu et al., 2023) is also a high-related router-based method. We plan to compare with it in the future since its implementation hasn't been released.

Implementations We employ Qwen-7B-Chat (Bai et al., 2023) as the meta LLM. The expert token embedding is initialized with the average value of the language modeling head of word tokens for training stability. We utilize AdamW (Loshchilov and Hutter, 2019) optimizer, with a learning rate of 5e-4 and weight decay set to 1.0, for a training duration of 5 epochs. We apply greedy decoding for all comparative methods, unless specifically stated otherwise. The ETR framework's training is performed on 1 × A100 GPU. The inference is conducted on 8 × A100 GPUs as it requires serving LLMs simultaneously.

3.2 Expert Query Set Collection

In our section, we collect the expert query set using the automatic pipeline described in Section 2.2, on an instruction dataset generated by GPT-4 containing 15,000 query-response pairs. The instruction dataset is generated by adopting a process similar to that depicted in Figure 3, using the dev set questions from all categories of MMLU as the seed question set. We also conduct a similar data cleaning process to ensure there is no data leakage in the instruction dataset. After filtering for qualifying queries, we perform down-sampling for each expert LLM, resulting in a set of 100 queries for each expert's query set.

3.3 Evaluation of Generalist framework on MMLU-Expert

Given a general-purpose LLM and an expert LLM library, we evaluate the performance of the generalist framework developed by comparison methods using the overall accuracy of MMLU-Expert (Table 2). From the data shown in the table, we can draw the following observations. (1) ETR achieves the highest overall accuracy (73.52%) among all methods, outperforming the runner-up method by a margin of 5.64%. Its performance also comes closet to the ideal scenario of selecting expert LLMs as if by an oracle. This indicates that ETR is highly effective to build a generalist framework by synergizing expert LLMs across diverse domains, significantly outperforming other approaches in nearly all six domains. (2) The multi-LLM debate strategy, despite boosting overall accuracy, does not guarantee improvements across all domains. (3) The performance of the LLM-blender series methods is subpar, possibly because the ranker model designed for general scenarios is not well-suited for expert routing and collaboration. (4) The Meta-Prompting-E technique stands as the runner-up methods, which utilizes carefully

crafted instructions for expert routing and collaboration, shows marked enhancements in fields like electrical engineering and international law. In Section 3.4, we delve deeper into the uneven performance across various domains. Yet, it falls short in overall accuracy compared to our approach.

3.4 Expert Routing Accuracy

In this section, we compare the expert routing accuracy in Table 3. For example, in the MMLU-Expert dataset, the questions in electrical engineering category has an oracle expert LLM, which is fine-tuned on the synthetic electrical engineering knowledge dataset (Section 3.1). The accuracy is 100% when all the queries are routed to its corresponding domain expert LLM, where random routing has an accuracy of 16.67%. It can be seen that the LLM-Blender, of which the ranking model is trained based on rewards aligned with human preferences, doesn't perform well in accurately routing to the domain expert. Meta-Prompting-E achieves a 67.08% expert routing accuracy on the MMLU-Expert dataset, while ETR reaches 82.11%, exceeding it by 15.03%.

Figure 4 visualizes the routing expert distribution of ETR and Meta-Prompting-E. It can be found that the Meta-Prompting-E performs poorly in certain domains, like astronomy, where it yields a mere 18.42% accuracy. This indicates that by merely designing crafted instructions, LLMs may struggle to accurately handle complex or implicit expertise. Comparing with the results in Table 2, it can be observed that in domains where Meta-Prompting-E's routing performance is subpar, the answer accuracy of meta-prompting is relatively low, affecting its overall performance as a generalist. In contrast, our approach maintains a routing accuracy exceeding 65% across all six domains, demonstrating greater robustness.

3.5 Analysis on Expert Query Set

The construction of the expert query set is crucial for learning expert tokens with good generalizability. Ideally, the more queries in the expert query set, and the higher their quality, the more comprehensive and accurate the depiction of expertise by the learned expert tokens. However, in practical use, because the instruction dataset we use to build the expert query set automatically can be relatively small, the number of expert queries obtained may be limited. Fortunately, ETR freezes the entire LLM and only tunes the expert tokens, which involves a very small number of parameters (expert number $\times$ embedding dimension). Thus, empirically we find that only 100 expert queries for each expert LLM are sufficient to achieve good results.

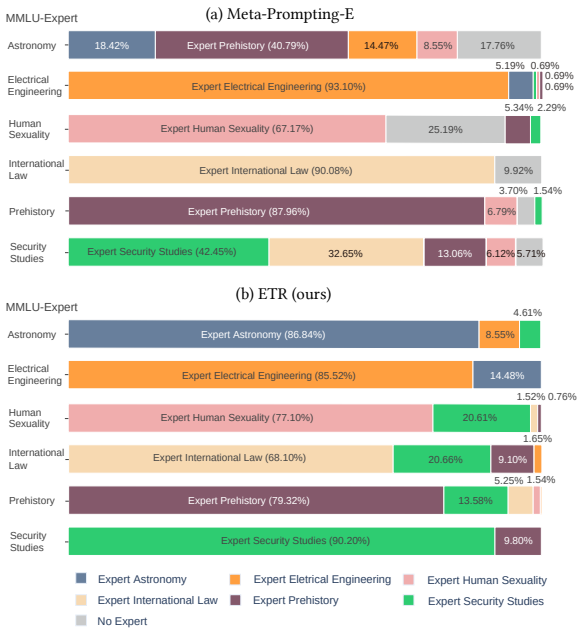

Figure 4: Routing expert distribution of Meta-Prompting-E (top) and ETR (down).

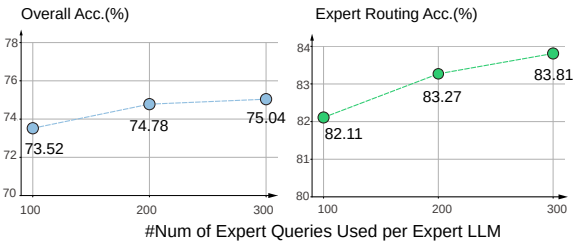

Figure 5: Overall Accuracy (Left) and Expert Routing Accuracy (Right) on MMLU-Expert as the size of the expert query set per expert increases.

Figure 5 illustrates that when the size of the expert query set is further expanded, both the overall accuracy and expert routing accuracy of our method improve. In our comparative experiments on effectiveness, we use a expert query set size of 100 per expert LLM to simulate scenarios where expert queries are limited. However, in scenarios where expert queries are easily accessible, the performance of our method can be further enhanced.

3.6 Dynamic Extension of Expert LLMs

In practical applications, the expert LLM library may continuously expand with the emergence of new expert LLMs. To adapt to the extended expert LLM library, prompt-based approaches must update their instructions, whereas router-based tech-

Table 2: Accuracy (%) on MMLU Expert dataset.

Method (7 LLMs) (1 Meta + 6 Experts)	Astronomy	Electrical Engineering	Human Sexuality	International Law	Prehistory	Security Studies	**Overall**
meta LLM	59.87	49.66	70.99	63.64	61.11	60.41	60.73
Oracle Expert	75.66	69.66	86.26	85.12	78.40	77.96	78.44
Meta-Prompting-E	58.55	64.14	83.21	72.73	72.22	59.59	67.89
Multi LLM Debate	55.26	54.48	71.76	61.98	63.89	64.49	62.29
LLM-Blender (Top1)	68.42	54.48	74.05	66.94	67.59	53.88	63.69
LLM-Blender (Fused)	59.87	46.21	74.05	64.46	59.26	44.90	56.80
ETR (Ours)	72.36	63.44	84.73	76.03	70.06	77.55	**73.52**

Table 3: Expert Routing Accuracy (%) on MMLU-Expert dataset.

Method	Expert Routing Acc.
Random	16.67
Oracle Expert	100.00
LLM-Blender (Top 1)	28.26
Meta-Prompting-E	67.08
ETR (Ours)	82.11

niques may need retraining the router model. Motivated by the weights of parameter-efficient tuning can be effectively combined (Wu et al., 2023), we train the expert tokens for the newly added expert LLMs and merge newly-learned expert tokens directly into the frozen expert token head in a plug-in manner. Table 4 compares the performance of the following two different settings on MMLU-Expert.

- ETR-Static: All expert LLMs exist in the expert LLM library at timestep 0 (i.e., the default setting of this paper).
- ETR-Dynamic: At timestep 0, expert tokens are trained based on the current expert LLM library. When new LLM experts are added at timestep 1, only the expert tokens of these new expert LLMs are trained and then concatenated with the previously obtained expert tokens.

Specifically, for the ETR-dynamic setting, we randomly selected four expert LLMs to form the expert LLM library at timestep 0. The remaining two serve as new LLM experts to be added at timestep 1. Table 4 shows that under the setting of dynamically adding expert tokens, ETR suffers only a minor performance drop. This reveals an attractive feature of our framework: providers of new expert LLMs can add their trained expert LLMs and expert tokens to the framework as plug-ins, without needing to modify any other part of the framework.

Table 4: Performance comparison between ETR-Static/Dynamic on MMLU-Expert.

Variant	Expert Routing Acc.	Overall Acc.
ETR-Static	82.11	73.52
ETR-Dynamic	81.75 (-0.26)	73.25 (-0.27)

3.7 Running Case Analysis

In Figure 6, we present a running case comparing between ETR and the runner-up prompt-based method (Meta-Prompting-E). Our observations yield the following insights:

- The prompt-based method requires the design of complex instruction templates to facilitate collaboration between the meta LLM and expert LLMs. ETR, however, obviates the need for using any specially designed instructions.
- In prompt-based methods, the information exchanging between LLMs is through conversations. While this approach is simple and effective, it adds an extra layer of complexity on the user end. In contrast, our ETR framework ensures that the collaboration between LLMs remains invisible to users, allowing them to interact with our framework as if it were a single LLM.
- In prompt-based methods, the routing to the appropriate expert relies on the meta LLM's intrinsic capabilities in zero-shot or few-shot scenarios. In contrast, our ETR framework can learn from expert queries, thus introducing new knowledge to the meta LLM, resulting in more precise expert routing.

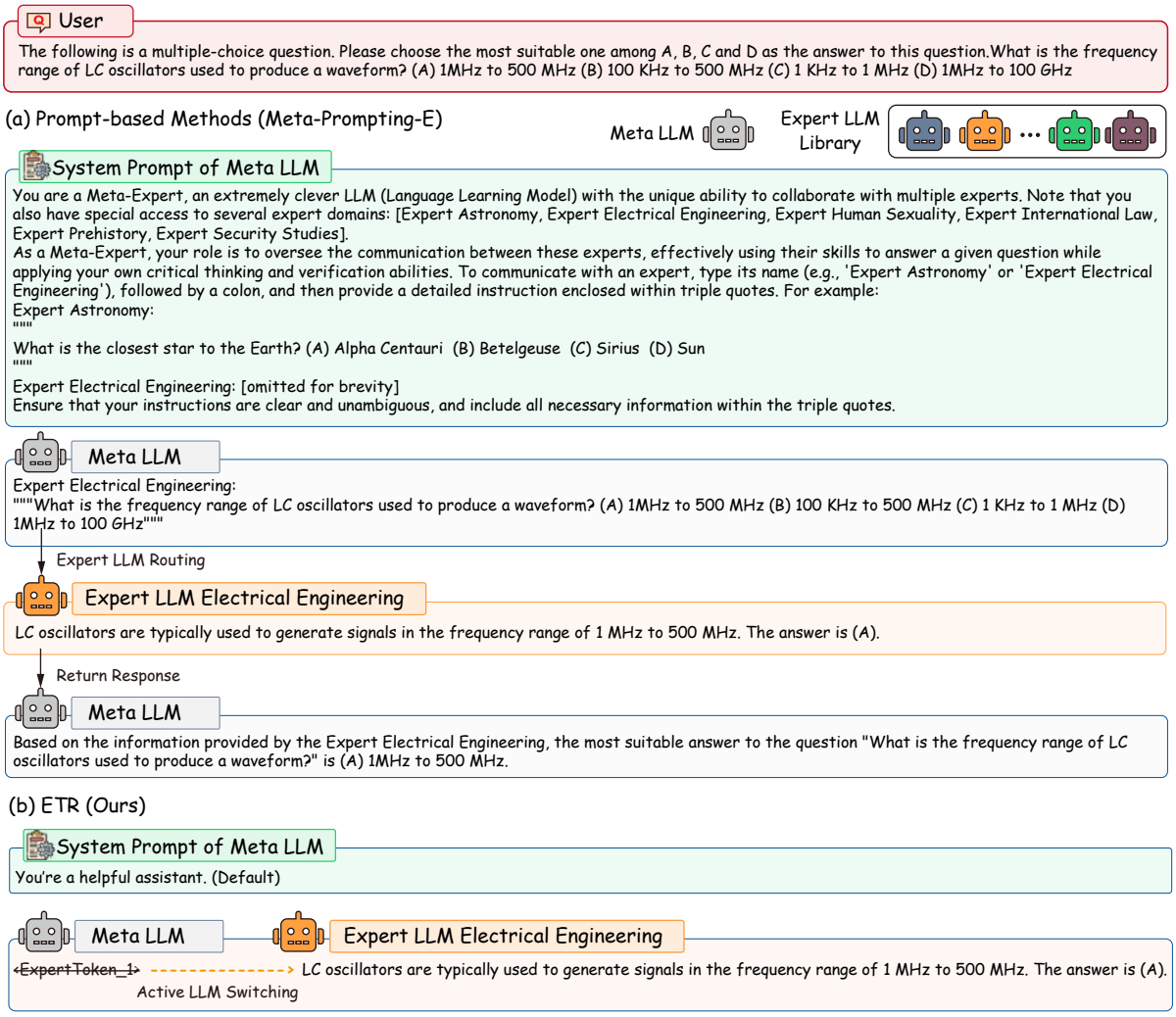

Figure 6: Comparison of running cases between prompt-based methods (Meta-Prompting-E) and ETR (Ours) on MMLU-Expert dataset. <ExpertToken_1> is the corresponding expert token to Expert LLM Electrical Engineering.

4 Related Works

expert LLM Collaboration A bunch of methods aim at enhancing the reasoning capabilities of LLMs in specific tasks by instructing LLMs into experts through the crafting of prompts. These methods mainly focus on boosting a single LLM by simulating multiple expert LLM instances, and can also be applied to our paper's setting. Expert-Prompting (Xu et al., 2023) elicits LLMs to generate expert-level responses using proper crafting of prompts. Meta-Prompting (Suzgun and Kalai, 2024) breaks down complex tasks into subtasks handled by expert instances of a single LLM. Multi-LLM Debate (Du et al., 2023) involves the use of multiple LLM instances to propose and debate their individual responses and reasoning processes over several rounds in order to converge on a common final answer.

There is also a line of methods highly relevant to this paper, which involves training an additional router model to integrate multiple LLMs into one framework. LLM-Blender (Jiang et al., 2023) uses a pair-ranker model for optimal LLM output selection and a gen-fuser model for merging the best outputs. Zooter (Lu et al., 2023) is a reward-guided routing method for LLM ensembles that efficiently assigns queries to the most expert LLM. However, in scenarios where expert models are dynamically extended, these methods often require retraining the router model.

5 Conclusion

In this study, we introduced a novel approach ETR to integrating various expert LLMs into a unified generalist framework. This method facilitates seamless collaboration among LLMs without exposing the complexity to the user, effectively turning the framework into a single, versatile LLM.

Through comprehensive experiments across multiple domains, ETR demonstrated superior performance over existing paradigms, offering a promising direction for enhancing LLM's generalist capabilities by combining the strengths of both general and expert models.

6 Limitations

The main limitation addressed in this paper is the necessity for a multi-expert LLM framework to operate all expert LLMs concurrently to provide real-time services. As the number of expert LLMs grows significantly, the associated costs could become prohibitive. A potential compromise involves distilling the expert knowledge from these LLMs into more efficient, lightweight modules.

Acknowledgment

This work is supported by National Natural Science Foundation of China (No. 62322606, No. 62441605) and SMP-IDATA Open Youth Fund.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report.

Wei Chen, Qiushi Wang, Zefei Long, Xianyin Zhang, Zhongtian Lu, Bingxuan Li, Siyuan Wang, Jiarong Xu, Xiang Bai, Xuanjing Huang, and Zhongyu Wei. 2023. Disc-finllm: A chinese financial large language model based on multiple experts fine-tuning. *arXiv preprint arXiv:2310.15205*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. Deepseek-coder: When the large language model meets programming – the rise of code intelligence.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding.

Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. 2024. Infiagent-dabench: Evaluating agents on data analysis tasks.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion.

Corentin Kervadec, Francesca Franzon, and Marco Baroni. 2023. Unnatural language processing: How do language models handle machine-generated prompts?

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. Camel: Communicative agents for "mind" exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023b. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls.

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

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2023. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. Routing to the expert: Efficient reward-guided ensemble of large language models.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron

Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

Mirac Suzgun and Adam Tauman Kalai. 2024. Meta-prompting: Enhancing language models with task-agnostic scaffolding.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, Yidong Wang, Linyi Yang, Jindong Wang, Xing Xie, Zheng Zhang, and Yue Zhang. 2023. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2024. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration.

Chengyue Wu, Teng Wang, Yixiao Ge, Zeyu Lu, Ruisong Zhou, Ying Shan, and Ping Luo. 2023. π-tuning: Transferring multimodal foundation models with optimal multi-task interpolation.

Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. 2023. Expertprompting: Instructing large language models to be distinguished experts.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. 2024. Loraretriever: Input-aware lora retrieval and composition for mixed tasks in the wild. *arXiv preprint arXiv:2402.09997*.

A Appendix

A.1 MMLU-Expert Dataset Statistic

Table 5: Dataset Statistics of MMLU-Expert

Domain	Test Question Number	Synthetic Dataset Size
Astronomy	152	7,500
Electrical Engineering	154	7,500
Human Sexuality	131	7,500
International Law	121	7,500
Prehistory	324	7,500
Security Studies	245	7,500

B Time Cost of LLM Switching

We evaluate the running time of ETR in two different conditions: no active LLM switching (line 9 in Algorithm 1) versus active LLM switching (line 11 in Algorithm Algorithm 1). We randomly select 100 queries to calculate the average running time. To eliminate the impact of different response length by the two conditions on the running time, the maximum generation length is set on the shorter response length of the two (greedy decoding is used for deterministic response). The experimental results are as shown in the following table.

Table 6: Time Cost of LLM Switching

	Running Time (s)
No Active LLM Switching	1.589
Active LLM Switching	1.616

The experiment runs with CUDA 12.1, Pytorch 2.1 and Flash-Attention 2. Each LLM is served on an A100-SXM4-80G GPU, utilizing bf16 quantization. We notice that active LLM switching results in roughly 1.7% extra running time costs. This ratio is expected to further decrease as the generated response becomes longer.

C Hyper-parameter Setup

C.1 expert LLM Fine-tuning Setup

We provide the hyper-parameters on how different domain expert LLMs are fine-tuned in Appendix C.1 for reproduction. To prevent overfitting on relatively small datasets during the fine-tuning process, we employ an early stopping mechanism to save the expert LLM's checkpoints.

Table 7: Hyperparameters for expert LLM fine-tuning.

Hyperparameter	Astronomy	Electrical Engineering	Human Sexuality	International Law	Prehistory	Security Studies
Gradient Accumulation	8	8	8	8	8	8
Batch size	16	16	16	16	16	16
Learning Rate	$5e-5$	$5e-5$	$5e-5$	$5e-5$	$5e-5$	$5e-5$
Epochs	5	8	5	8	5	6
Warmup steps	50	50	50	50	50	50
Weight decay	$1e-1$	$1e-1$	$1e-1$	$1e-1$	$1e-1$	$1e-1$
Tunable parameters	Full	Full	Full	Full	Full	Full

C.2 Expert Token Training Setup

We provide the hyper-parameters on expert token training in Table 8 for reproduction. For the sake of training stability, the initialization of the expert token is conducted using the mean value of embeddings found within the language modeling head of the meta LLM. It can be seen that when using Qwen-7B-Chat as the meta LLM, the parameters that need to be trained amount to only 24,567, which accounts for merely 3.5110^{-6} of the total parameters.

Table 8: Configuration of training expert tokens

Hyper Parameter	Value
Meta Model	Qwen-7B-Chat
Train Batch Size	16
Model Max Length	1024
Learning Rate	0.0005
Weight Decay	1.0
Number of Training Epochs	5
Learning Rate Scheduler Type	Cosine
Number of Warmup Steps	5
Expert Query Set Size	600
# of Trainable Params.	24,567

D A Running Case of Extracting Domain Knowledge from GPT-4

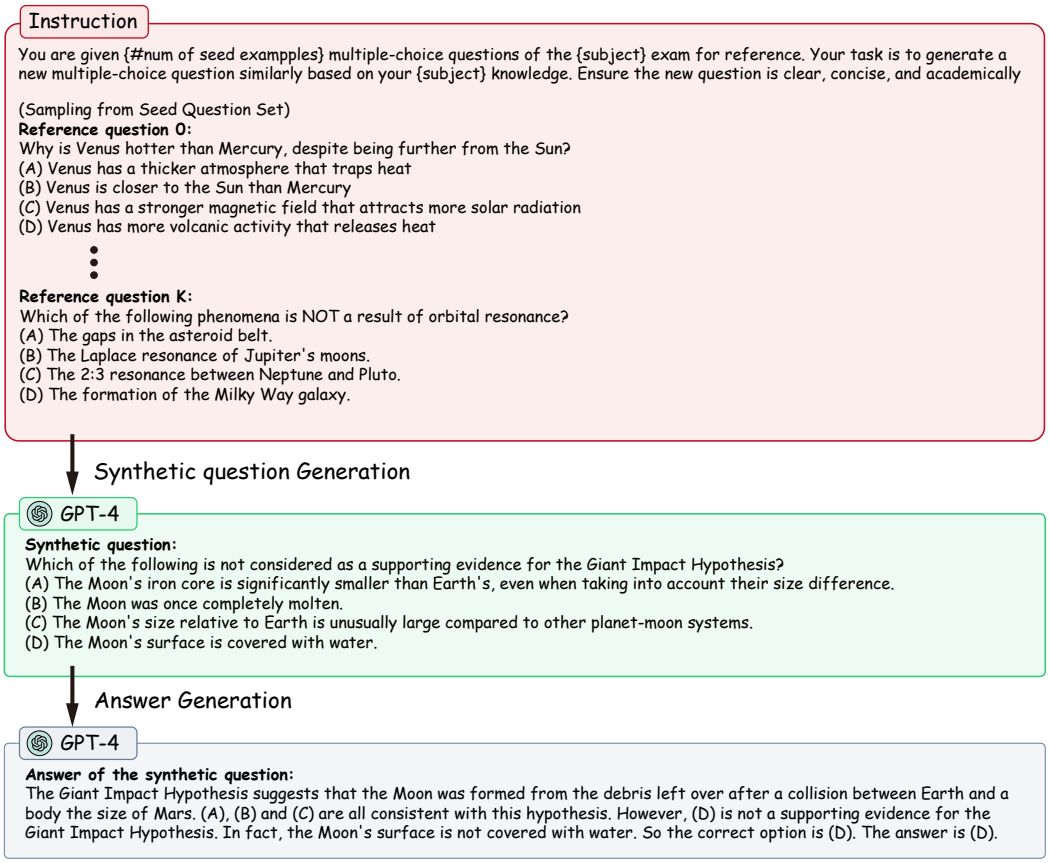

Figure 7: A running case of extracting knowledge from GPT-4 by generating synthetic questions and answers.

In the evaluation phase, the domain-specific dataset for creating the expert model and the instruction dataset from which expert queries are extracted are both constructed using GPT-4. Thus, Figure 7

illustrates a example to generate synthetic questions for the extraction of domain knowledge. The generated questions are expanded into the seed question set.