R-Eval: A Unified Toolkit for Evaluating Domain Knowledge of Retrieval Augmented Large Language Models

Shangqing Tu*
DCST, Tsinghua Univerisity
Beijing 100084, China
tsq22@mails.tsinghua.edu.cn

Yuanchun Wang*
SoI, Renmin University of China
Beijing 100084, China
wangyuanchun@ruc.edu.cn

Jifan Yu
DCST, Tsinghua Univerisity
Beijing 100084, China
yujf21@mails.tsinghua.edu.cn

Yuyang Xie
DCST, Tsinghua Univerisity
Beijing 100084, China
xieyy21@mails.tsinghua.edu.cn

Yaran Shi
SIOE, Beihang Univerisity
Beijing 100084, China
syr2021@buaa.edu.cn

Xiaozhi Wang
DCST, Tsinghua Univerisity
Beijing 100084, China
wangxz20@mails.tsinghua.edu.cn

Jing Zhang
SoI, Renmin University of China
Beijing 100084, China
zhang-jing@ruc.edu.cn

Lei Hou
BNRist, DCST, Tsinghua Univerisity
Beijing 100084, China
houlei@tsinghua.edu.cn

Juanzi Li
BNRist, DCST, Tsinghua Univerisity
Beijing 100084, China
lijuanzi@tsinghua.edu.cn

ABSTRACT

Large language models have achieved remarkable success on general NLP tasks, but they may fall short for domain-specific problems. Recently, various Retrieval-Augmented Large Language Models (RALLMs) are proposed to address this shortcoming. However, existing evaluation tools only provide a few baselines and evaluate them on various domains without mining the depth of domain knowledge. In this paper, we address the challenges of evaluating RALLMs by introducing the R-Eval toolkit, a Python toolkit designed to streamline the evaluation of different RAG workflows in conjunction with LLMs. Our toolkit, which supports popular built-in RAG workflows and allows for the incorporation of customized testing data on the specific domain, is designed to be user-friendly, modular, and extensible. We conduct an evaluation of 21 RALLMs across three task levels and two representative domains, revealing significant variations in the effectiveness of RALLMs across different tasks and domains. Our analysis emphasizes the importance of considering both task and domain requirements when choosing a RAG workflow and LLM combination. We are committed to continuously maintaining our platform at https://github.com/THU-KEG/R-Eval to facilitate both the industry and the researchers.

CCS CONCEPTS

• **Computing methodologies** → **Discourse, dialogue and pragmatics**; **Natural language generation**.

*Both authors contributed equally to this research.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

KEYWORDS

Evaluation, Domain Knowledge, Large Language Model

ACM Reference Format:
Shangqing Tu, Yuanchun Wang, Jifan Yu, Yuyang Xie, Yaran Shi, Xiaozhi Wang, Jing Zhang, Lei Hou, and Juanzi Li. 2018. R-Eval: A Unified Toolkit for Evaluating Domain Knowledge of Retrieval Augmented Large Language Models. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The burgeoning advancements in large language models (LLMs), such as GPT-4 [29] and Llama [40], have sparked widespread interest across industry, academia, and the public sphere [2]. However, while LLMs excel in general tasks [5, 9], their performance can falter when confronted with domain-specific tasks [15, 50]. This shortfall is primarily due to the potential absence of domain knowledge [1], defined as pre-existing information and expertise within a specific field. Consequently, the technique of retrieval augmented generation (RAG) [18] has gained traction, particularly in adapting LLMs for domain-specific applications such as AI healthcare assistants, as it offers a solution to mitigate the propensity of LLMs to generate hallucinated responses [19, 39, 48].

As LLMs continue to evolve [55], becoming more powerful and resource-intensive than small pre-trained language models, a plethora of new RAG workflows have been introduced specifically for these models. As depicted in Figure 1, these innovative approaches typically start by retrieving relevant resources based on user input. Subsequently, they either synthesize outputs from independently executed tools, as exemplified by program-aided language model workflow (PAL) [10] and OpenAI's Function Calling[1] (Figure 1 bottom), or adopt a sequential execution and prompt-based reasoning process, as demonstrated by DFSDT [31] and ReAct [53] (Figure 1 top). These advancements in RAG workflows for LLMs are opening new avenues for exploration and evaluation.

[1]https://platform.openai.com/docs/guides/function-calling

Despite the progress in the field, there is still a notable scarcity of software tools that offer a simple, all-in-one evaluation system for different retrieval augmented large language models (RALLMs). However, prior evaluation works lacks consideration of two key factors: (1) Insufficient exploration of **combinations between LLMs and RAG workflows**. Recent RAG evaluation frameworks such as RAGAS [6] and ARES [35], offer only a limited number of baselines and do not fully explore the myriad possible combinations of RAG workflows and LLMs [27]. (2) Lack comprehensive **mining of the domain knowledge**. Furthermore, recent benchmarks like ToolBench [31] and API-Bank [21] primarily focus on the general capabilities of RALLMs across various domains, with each domain typically containing only about 10 test cases on average, resulting in a shortage of testing data. A unified toolkit could facilitate fair comparisons and promote wider adoption of various RALLM systems for domain-specific applications.

To address this gap, we introduce R-Eval (RALLM Evaluation Toolkit), a Python toolkit designed to streamline the evaluation of different RAG workflows in conjunction with LLMs. Our toolkit offers users the flexibility to explore four popular built-in RAG workflows: ReAct [53], PAL [10], DFSDT [31], and function calling[1]. In addition, it provides the capability to incorporate customized testing data in specific domains through template-based question generation. An included analysis module can automatically perform performance, error, and deployment analyses. Distinct from other LLM evaluation toolkits, R-Eval is one of the first to prioritize the evaluation of domain knowledge in RALLMs. It is designed to be user-friendly, modular, and extensible, offering a robust and comprehensive evaluation framework for the community.

Based on the R-Eval toolkit, we conduct an evaluation of 21 RALLMs for three task levels on two representative domains: Wikipedia [51] and Aminer [38], leading to some interesting findings:

We discover that the effectiveness of RALLMs can vary significantly across different tasks and domains. In particular, the combination of the ReAct workflow with the GPT-4-1106 model exhibits exceptional performance across all tasks and domains, making it a robust choice for a variety of applications. However, the optimal RAG workflow and LLM combination may vary depending on the specific task or domain. For instance, other combinations such as PAL with GPT-4-1106 or ReAct with Llama2-7B-chat [40] also demonstrate strong performance in certain areas. Our Matching Analysis further reveals that while GPT-4-1106 is the best matching LLM for the ReAct workflow across all tasks and domains, other workflows like PAL may yield comparable results with different LLMs, such as GPT-3.5-turbo-1106.

In our Error Analysis, we find that the PAL workflow produces the highest proportion of tool-using errors, indicating that it often fails to successfully retrieve domain knowledge, which could explain why GPT-4 performs poorly with PAL. On the other hand, workflows like ReAct, DFSDT, and FC generate fewer tool-using errors, resulting in more successful knowledge retrieval and better overall performance. In terms of practical deployment, we find that GPT models, which are called through APIs, "outperform" open-source LLMs in terms of both efficiency and effectiveness. Among the open-source LLMs, Tulu-7B [46] strikes a good balance between efficiency and effectiveness, showcasing its potential for practical applications. Our findings underscore the importance of considering both task and domain requirements when choosing a RAG workflow and LLM combination. These insights offer valuable guidance for developers and researchers in the selection of suitable RALLMs for specific tasks or domains.

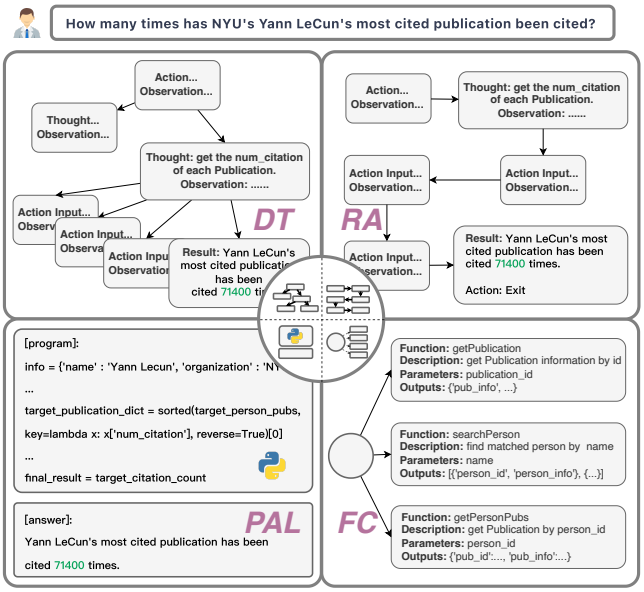


Figure 1: Responses from 4 RAG workflows, including DFSDT(DT) [31], ReAct(RA) [53], PAL [10] and GPT Function Calling (FC) for a domain-specific question.

Impact and Beneficiaries. We've crafted the R-Eval toolkit for thorough RALLMs evaluations and performance analysis. This toolkit aids: (1) *Researchers* in evaluating and contrasting RALLMs across tasks and domains, thereby guiding future research. (2) *Industry Professionals*, particularly in AI, by offering a resource for assessing RALLMs' real-world applicability, aiding in informed model selection. (3) *Developers*, by providing a flexible platform to test, refine, and deploy their RALLMs, and understand trade-offs between efficiency and effectiveness.

2 BACKGROUND

Retrieval Augmented LLMs. Since LLMs have successfully improved the performance of various general domain tasks [2, 29], researchers have been exploring the methods that adapt LLMs to domain-specific tasks. The most common solution is to build a retrieval augmented LLM (RALLM) that utilize the domain knowledge via retrieval [24]. Previous retrieval workflows for LLM can be broadly categorized into two categories: (1) Planned Retrieval: The retriever plans what knowledge to fetch according to the question and pass them to LLMs for the answer generation [17, 23, 25, 36]. (2) Interactive Retrieval: As the retriever may occasionally fail to yield accurate or comprehensive results [44], an interactive retrieval mechanism that grants LLMs the capability to refine the retrieval process can effectively tackle these challenges [12, 37, 42, 43, 53].

Domain-specific Evaluation. The powerful generation ability of large language models (LLMs) has had a profound impact in

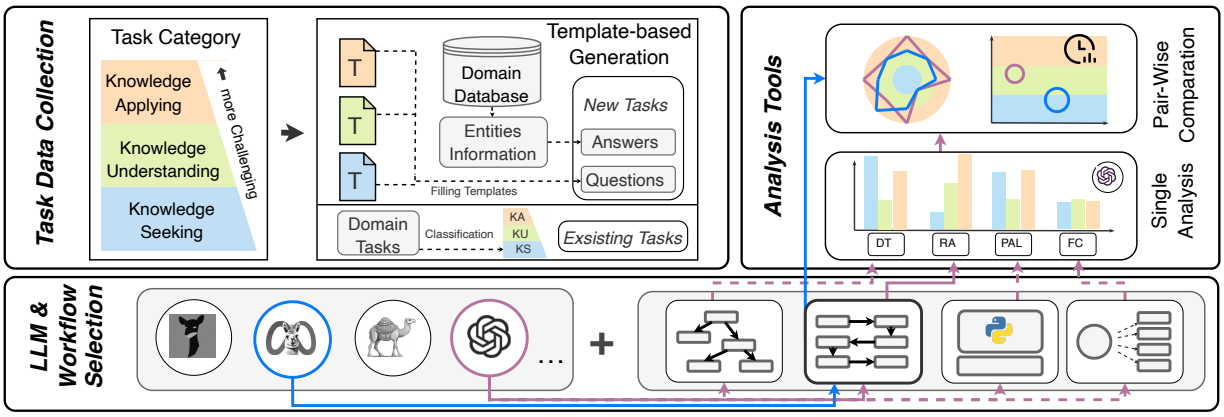


Figure 2: The framework of our evaluating and analysing process. We first choose an environment and collect testing data from both existing benchmarks and template-based QA pairs. Then we select the RAG workflows and LLMs to form a RALLM for running the evaluation. After that, we perform a comprehensive analysis on results to get insights.

various fields, such as law [8], finance [49], medicine [13, 20] and education [7, 26]. Therefore, many efforts have been devoted to evaluating the capabilities of large language models for a certain domain with various new benchmarks, such as LegalBench [11], CMB [45] and EcomInstruct [22]. However, previous domain-specific evaluations mainly focus on providing tailored task data, ignoring the importance of building an interactive environment for RAG settings [18], which are widely adopted in real-world applications. This limitation inspires our work to build easy-to-adapt environments and free-to-combine retrieval workflows for RALLM evaluations.

Problem Definition We define the problem as follows: Given a domain-specific task T that necessitates specialized knowledge from a particular field, an environment K for querying domain knowledge, and a RALLM R with a workflow to fetch relevant information based on the input and generate a response via LLM, our objective is to assess the performance of R on T using K. The main challenge lies in the fact that different combinations of large language models and retrieval workflows can lead to different results, and the optimal combination may vary depending on the specific task or domain. Furthermore, the evaluation should not only consider the final output but also the process of retrieving information and generating the response, which involves analyzing the strengths and weaknesses of different RALLMs.

3 SYSTEM

To investigate how well LLMs can master domain knowledge, we propose R-Eval, a unified evaluation toolkit that can be adapted to any specific domain. As shown in Figure 2, R-Eval can generate test cases, load various RALLMs and analyse their performances.

3.1 Environment Setting

In the context of a Retrieval Augmented Large Language Model system, the LLM interacts with the domain's API in a tool-using manner, retrieving the necessary knowledge to accomplish related tasks. In order for LLMs to retrieve domain knowledge, we design an environment with several query APIs for each domain. We begin with two representative domains as use cases: (1) *Wikipedia*[2] is a high-quality knowledge source with over 6.6 million English articles, which has provided supporting facts for building various open-domain QA tasks [18]. We use a simple Wikipedia web API with three types of functions to enable interactive information retrieval, including Search, Lookup and Finish, which is the same as ReAct's environment [53]. (2) *Aminer*[3] [38] is an academic information service system containing over 69 million scholar profiles and 290 million publications. We define 7 query APIs for data retrieving about scholars and publications, including searchPerson, searchPublication and getCoauthors, etc (More are in Appendix A).

3.2 Task Data Collection

Knowledge, which means information including facts, events, and skills, has been utilized as an indicator for AI's intelligence level [28].

Task Category. Considering the cognitive ability modeling for knowledge [55], we select three widely accepted processes in Bloom's taxonomy [16] for organizing the tasks in R-Eval benchmark:

- **Knowledge Seeking (KS)** is designed to assess the model's capacity to accurately recall established facts from the environment, as demonstrated by the prior Open-domain QA task [4].
- **Knowledge Understanding (KU)** aims to evaluate the model's proficiency in comprehending the inherent knowledge embedded in texts, as manifested by the tasks of information extraction [30, 54].
- **Knowledge Application (KA)** measures the ability of models to utilize retrieved knowledge in performing reasoning and problem-solving tasks, which is assessed through various knowledge reasoning tasks like multi-hop reasoning [3, 41, 52].

Test Datasets. The R-Eval benchmark, as depicted in Table 1, encompasses 12 tasks designed to assess the three levels of cognitive ability for 2 domains. These tasks are constructed from both existing datasets and refreshing template-based question generation.

[2]https://www.wikipedia.org
[3]https://www.aminer.cn

Table 1: The domain specific tasks in R-Eval. Test Set and Pool correspond to the testing instances used in each season and the overall available instances. *Existing* tasks means their test sets are taken from the original dataset. *Refreshing* tasks' instances are newly developed by our template-based question generation process.

Domain	Level	ID	Dataset	Metrics	Context Type	Test Set	Pool	Source
Wiki	KS	1-1	High-Freq.	**F1**	Triple	100	20.6M	Refreshing
		1-2	Low-Freq.	**F1**	Triple	100	20.6M	Refreshing
	KU	2-1	COPEN-CSJ	**F1**	Entity, Concept	100	3.9k	Existing
		2-2	COPEN-CPJ	**F1**	Concept	100	4.7k	Existing
		2-3	COPEN-CiC	**F1**	Concept	100	2.3k	Existing
	KA	3-1	HotpotQA	**F1**	Document(s)	100	7.4k	Existing
		3-2	2WikiMulti.	**F1**	Document(s)	100	12.6k	Existing
		3-3	MuSiQue	**F1**	Document(s)	100	2.5k	Existing
		3-4	KQA Pro	**F1**	KG	100	1.2k	Refreshing
Aminer	KS	1-3	Soay-Easy	**F1**	Triple	100	12.7k	Refreshing
	KU	2-4	Profiling	**F1**	Document, Entity	100	1.8k	Existing
	KA	3-5	Soay-Hard	**F1**	KG	100	12.7k	Refreshing

Knowledge Seeking Tasks evaluate the model's ability to recall facts. The tasks on wiki domain are constructed from triplets in Wikidata5M, transformed into sentences with relation-specific templates. Two test sets are created, one for high-frequency knowledge (1-1) and another for low-frequency knowledge (1-2). A knowledge seeking test of aminer domain (1-3) is also included, which annotates knowledge triplets from the aminer knowledge base.

Knowledge Understanding Tasks assess the model's understanding of various types of knowledge from texts. They include Concept Probing (2-1/2-2/2-3) from the COPEN dataset for Wikipedia, which are conceptual similarity judgment (CSJ), conceptual property judgment (CPJ) and conceptualization in contexts (CiC) respectively. A dataset on aminer (2-8) is also included, which evaluates the ability of extracting structured profiles from plain text.

Knowledge Application Tasks measure the model's ability to apply retrieved knowledge in multi-hop reasoning tasks. They include tasks from the HotpotQA (3-1), 2WikiMultihopQA (3-2), MuSiQue (3-3), and KQA Pro (3-4) datasets. From 3-1 to 3-4, the reasoning difficulty on wiki domain knowledge is increasing. Another KA test for aminer domain (3-6) is also included, producing questions based on the domain knowledge.

Template-based Generation. Inspired by previous works [3, 47], we utilize a template-based generation approach to rapidly construct evaluation sets from a given domain database. More specifically, we initially compose a series of template questions with placeholders based on the content we need to evaluate, then fill these templates with meaningful information by sampling entries from the domain database. Additionally, the answers to these questions can be also derived directly from the entity information. Notably, after manually crafting a few template questions, LLMs can be employed to quickly complete subsequent template construction.

Taking the domain of academic intelligence as an example, we could design a template question such as *"What are the research interests of XXX at X institution?"*. From the AMiner database, we can extract a plethora of scholar entity information like names, affiliations, interests, and email addresses, for example, *{'name' : 'Yann Lecun', 'organization' : 'New York University', 'Interest' : 'AI, Machine Learning, Computer Vision, Robotics, Image Compression', 'email' : 'yl22@nyu.edu', ...}*. Consequently, this template question can be filled as, *"What are the research interests of Yann Lecun at New York University?"* and the answer can be found within the entity information. Employing this method, we've designed an extensive range of QA test sets for distinct task difficulty levels. The *aminer KS* and *aminer KA* tasks shown in table 3 are generated in this way with varying difficulty levels.

3.3 Workflow and LLM Selection

As shown in Table 2, a RALLM system usually composes of a workflow for retrieving domain knowledge and an LLM for reasoning [24]. R-Eval presents a feasible protocol that combines different workflows and LLMs for comprehensive evaluation.

Workflow Selection. Following BOLAA [27], we select 4 typical baseline workflows for retrieving knowledge from our environment: (1) *ReAct* [53] is a prompt-based paradigm to synergize reasoning and acting in language models for task solving, which create action spaces for LLMs to retrieve external knowledge. (2) *PAL* [10] is a program-aided workflow that allows LLMs to generating python programs for executing API queries. (3) *DFSDT* [32] is a general decision-making strategy for API using, which enhances the reasoning capabilities of LLMs by considering multiple reasoning traces in the environment. (4) *FC*[1] is short for Function Calling, which is a close-source tool using workflow by OpenAI.

LLM Selection. The evaluated LLMs consist of two categories: (1) *Open-source Model*, including Llama2-chat (7B) [40], Tulu (7B) [46], Vicuna (13B) [56], Llama2 (13B) [40], CodeLlama-instruct (13B) [34], ToolLlama-2 (7B) [31]. (2) *Commercial Model*: GPT3.5-turbo[4] and

[4]https://platform.openai.com/overview

Table 2: Comparison with existing works' evaluations of RALLMs. # is the short for 'number'. The column *Levels of Tasks* refers to the task categories divided in the evaluation. *Analysis Toolkit* means whether this work provides an analysis toolkit.

Research for RALLM	Number of RAG workflows	Number of LLMs	# of evaluated feasible RALLMs	Levels of Tasks	Number of Tasks	Avg. test data for each domain	Analysis Toolkit
ToolQA [57]	2	2	3	(easy, hard)	6	255	×
ToolBench [31]	3	5	6	(I1, I2, I3)	6	258.3	×
API-Bank [21]	1	6	6	None	4	6.1	×
FLARE [12]	4	1	4	None	4	432.3	×
IRCoT [42]	3	2	6	None	4	500	×
DECOMP [14]	3	2	6	None	4	100	×
ReAct [53]	**4**	3	12	None	2	500	×
R-Eval (ours)	**4**	**8**	**21**	(KS, KU, KA)	**12**	**600**	✓

GPT-4 [29]. As the workflows may require LLMs having the instruction following abilities to produce specific output format like code for PAL, not all LLMs can fit the workflows. Therefore, we tried to combine LLMs with the workflows and keep those combination that can achieve non-zero performance on our benchmark. Finally, we get 21 feasible RALLM systems for evaluation, which has more RALLMs than those implemented in previous works [31, 53].

3.4 Analysis Tools

We developed three automated analysis tools to evaluate the performance of various RALLMs in domain-specific tasks. (1) **Matching Analysis** facilitates comparison between systems on the same RAG workflow by visualizing the evaluation results. It normalizes systems' performance via a radar map, which can reflect the difference directly. (2) **Error Analysis** provides a detailed view of each system's weakness across different tasks and identifies common error types. This helps to pinpoint each system's areas for improvement. (3) **Deployment Analysis** assesses the trade-off between runtime speed and performance for each system, aiding decision-making in system deployment. These tools offer a comprehensive, multifaceted analysis of RALLM system performance, providing valuable insights for their application in domain-specific tasks.

Error and Response Types. To provide a more comprehensive error analysis for RALLMs, we have conducted a fine-grained taxonomy of system response types, which include all error and correct cases.

Under the R-Eval framework, all RAG systems, besides preserving the final outcome (response), also document the information retrieved (scratchpad), and whether the inference process was interrupted due to erroneous usage or faults with the retrieval tools. Based on these records, we categorized the system responses into six types in a fine-grained manner. When the response aligns with the standard answer, if the scratchpad contributes to the final response, we define this situation as **E**xact **M**atch (EM). If the information in the scratchpad doesn't relate to the response, i.e., the system can answer consistently with the standard answer relying on the knowledge stored within the model, it's considered an **A**nswer **M**atch (AM). As to the situation that the response does not match the standard answer, if the scratchpad provides useful information, there are no problems with the retrieval process but something is wrong in the model's generation process grounded on the retrieved content, we define this as **G**rounded-generation **E**rror (GE). If the scratchpad's information is irrelevant, the faulty retrieval process leads to undesirable responses. We then have to ascertain whether the retrieval process was interrupted. There is an issue with the logic of the retrieval if the process is halted. We denote this as a **R**easoning **E**rror (RE). If the retrieval process was interrupted, depending on whether the interruption was due to faults in the retrieval tool itself or improper use of the tool, it is classified as a **M**odel **E**rror (ME) or **T**ool-using **E**rror (TE), respectively. Specially, R-Eval uses the F1-Score between system responses and standard answers to determine whether the response matches the standard answer, and the usefulness of the scratchpad is assessed by checking if the response is contained within the scratchpad.

This kind of fine-grained classification enables us to quickly assess the capabilities of an RAG system and diagnose the cause of errors. For instance, a higher proportion of EMs signifies better RAG capabilities. A high proportion of AMs implies the system model has strong internal capabilities but weak retrieval capabilities. GE is in contrast to AM: the higher the proportion of GEs, the stronger the retrieval capabilities but the weaker the model's capabilities. A high proportion of REs might indicate that the current system's reasoning capabilities are insufficient to support the current workflow. The presence of MEs and TEs reveals issues with the model itself or the retrieval tool interface, necessitating separate inspections.

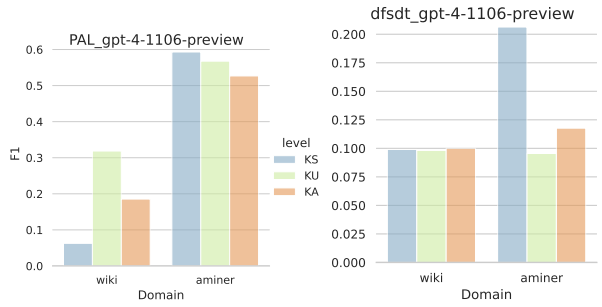
4 EVALUATION EXPERIMENTS

4.1 Evaluation Settings

We evaluate all the combinations between four RAG workflows and eight LLMs. However, as FC (Function calling) is implemented by OpenAI, only GPT-3.5 and GPT-4 can use it. Besides, the DFSDT also resembles the format of API calling of FC, therefore, for open-sourced models, only toolllama that trained on this format can use it. Therefore, we get total 21 feasible RALLM systems and report their results on aminer domain (Table 3) and wiki domain (Table 5). All RALLMs are evaluated under the same one-shot setting, with the same decoding hyper-parameters. More details are in Appendix B. In the following sections, we will organize our experiment and analysis results with six research questions (RQs).

Table 3: Performance of different systems for each task on Aminer domain and the average performance of overall tasks.

Workflow	LLM	aminer KS		aminer KU		aminer KA		Overall Average (Level 1, 2, 3)					
		1-3	Rank	2-4	Rank	3-5	Rank	wiki	Rank	aminer	Rank	all	Rank
ReAct	gpt-4-1106	89.7	1st	46.7	3rd	57.7	1st	38.8	1st	64.7	1st	45.3	1st
PAL	gpt-3.5-turbo	80.1	3rd	50.7	2nd	54.9	2nd	19.9	6th	61.9	2nd	30.4	2nd
PAL	gpt-4-1106	59.3	4th	56.8	1st	52.7	3rd	20.3	5th	56.2	3rd	29.2	3rd
ReAct	llama2-7b-chat	45.2	5th	36.5	6th	21.5	6th	23.8	3rd	34.4	5th	26.4	4th
PAL	llama2-13b	25.3	6th	36.4	7th	20.3	7th	25.2	2nd	27.3	6th	25.7	5th
ReAct	gpt-3.5-turbo	84.6	2nd	4.0	14th	33.0	4th	19.6	7th	40.6	4th	24.9	6th
ReAct	vicuna-13b	19.9	10th	6.0	13th	7.1	16th	20.7	4th	11.0	17th	18.2	7th
PAL	tulu-7b	9.1	15th	26.8	9th	11.5	12th	18.9	8th	15.8	9th	18.1	8th
PAL	vicuna-13b	4.5	17th	40.9	4th	2.3	20th	16.7	9th	15.9	8th	16.5	9th
ReAct	llama2-13b	16.7	13th	0.7	19th	23.2	5th	15.0	10th	13.5	12th	14.6	10th
PAL	llama2-7b-chat	18.7	12th	2.8	15th	16.1	8th	12.4	11th	12.5	14th	12.4	11th
PAL	codellama-13b	4.4	18th	38.3	5th	8.1	14th	10.0	14th	16.9	7th	11.7	12th
PAL	toolllama2-7b	1.6	20th	24.4	10th	4.6	18th	12.2	12th	10.2	18th	11.7	13th
ReAct	tulu-7b	4.0	19th	27.8	8th	7.9	15th	10.3	13th	13.2	13th	11.0	14th
DFSDT	gpt-4-1106	20.6	9th	9.6	12th	11.8	11th	9.9	15th	14.0	11th	10.9	15th
FC	gpt-4-1106	24.7	7th	10.9	11th	10.2	13th	8.2	18th	15.3	10th	9.9	16th
FC	gpt-3.5-turbo	19.0	11th	1.0	17th	15.9	9th	8.8	16th	12.0	15th	9.6	17th
ReAct	toolllama2-7b	15.0	14th	2.2	16th	5.7	17th	8.3	17th	7.6	19th	8.1	18th
DFSDT	gpt-3.5-turbo	20.7	8th	0.2	20th	13.8	10th	4.8	20th	11.6	16th	6.5	19th
ReAct	codellama-13b	0.2	21th	0.8	18th	0.7	21th	7.0	19th	0.6	21th	5.4	20th
DFSDT	toolllama2-7b	7.1	16th	0.0	21th	2.3	19th	3.5	21th	3.1	20th	3.4	21th



(a) PAL (b) DFSDT

Figure 3: Average performance on different levels' tasks and domains for GPT-4 with the PAL and DFSDT workflow.

4.2 Task Results

RQ1: *How effective are RALLMs across three levels' tasks?*

The evaluation of RALLMs across three levels of tasks, namely Knowledge Seeking (KS), Knowledge Understanding (KU), and Knowledge Application (KA), reveals interesting patterns.

For KS and KA tasks, which assess the model's ability to recall facts and then utilize retrieved knowledge in performing reasoning, the results suggest a potential correlation between a model's rank on KS and KA tasks. Some models, such as the ReAct with GPT-4-1106, demonstrate superior performance on both KS and KA tasks, securing the top rank among all RALLMs, which suggests a strong correlation between retrieving relevant facts and applying retrieved knowledge to new problems. On the other hand, some models like PAL with toolllama2-7b struggle with these tasks, indicating potential areas of improvement in retrieval and reasoning.

In the Knowledge Understanding tasks, which evaluate the model's comprehension of the inherent knowledge embedded in texts, the performance varies as well. Some models that are good at KS and KA tasks perform not as well on KU tasks. For example, ReAct with GPT-4 only ranks the third on aminer's KU task, suggesting that these models may have difficulty interpreting or understanding the underlying knowledge in a given text. However, models with PAL workflow show a strong understanding performance. For example, PAL with GPT-4 performs the best on the KU task of aminer domain. In Figure **??**, we observe that this system can also perform well on KU tasks of Wiki domain, proving its ability on understanding.

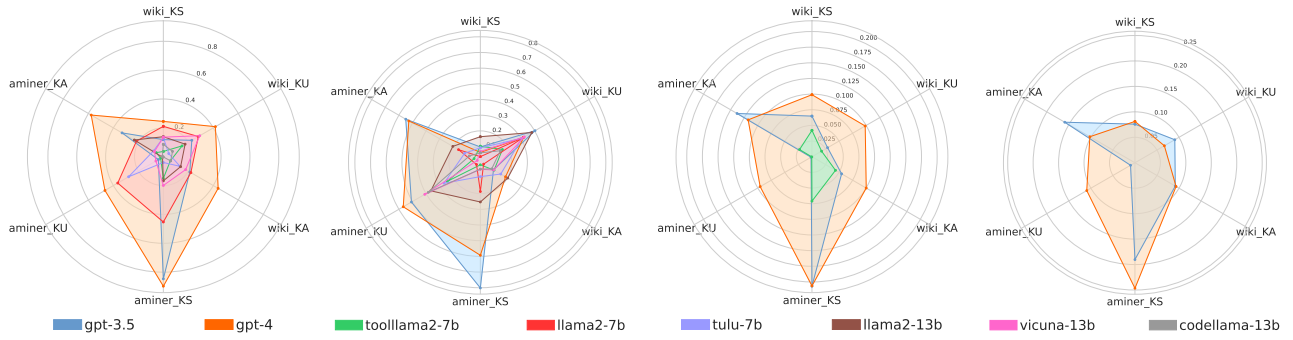
In summary, the effectiveness of RALLMs across the three levels of tasks varies significantly. While some models excel in certain tasks, they may falter in other tasks, highlighting the complexity and challenge of developing systems that can effectively retrieve, understand, and apply domain-specific knowledge.

.

4.3 Domain Results

RQ2: *How effective are RALLMs on wiki and aminer domain?*

The effectiveness of RALLMs also varies significantly across different domains. In the Aminer domain, which has over 69 million scholar profiles and 290 million publications, the performance of RALLMs is varied. Some models, like ReAct with GPT-4-1106, perform well, indicating a strong ability to handle domain-specific knowledge. However, other models may encounter challenges within



(a) ReAct (b) PAL (c) DFSDT (d) FC

Figure 4: Radar map of single system's performance on all tasks for different 4 workflows.

this domain. As illustrated in Table 3, models using the ReAct workflow generally secure higher performance ranks for the Wikipedia domain than for the Aminer domain. This implies that while these models are adept at handling knowledge from a broader domain, they might struggle with the specific types of knowledge and data present within the Aminer domain.

Overall, while some RALLMs demonstrate strong performance across both the Wikipedia and Aminer domains, others struggle in one or both domains. This highlights the importance of developing models that can effectively handle both broad, open-domain knowledge and domain-specific knowledge.

4.4 System Comparison

RQ3: *Which RAG workflow and LLM combination is the best?*

From the results, it appears that the combination of the ReAct workflow with the GPT-4-1106 LLM performs exceptionally well across both the Wikipedia and Aminer domains, as well as across all three levels of tasks. This combination seems to provide a strong balance of fact retrieval, knowledge understanding and application, making it a robust choice for a variety of tasks and domains.

However, it's important to note that while this combination outperforms others in this evaluation, the "best" combination may vary depending on the specific task or domain at hand. For example, other combinations, such as PAL with GPT-4-1106 or ReAct with Llama2-7B-chat, also show strong performance in certain tasks or domains, where PAL with GPT-4-1106 even surpasses ReAct with GPT-4-1106 on the understanding task of aminer domain.

Therefore, while the ReAct and GPT-4-1106 combination appears to be the best overall, other combinations may be more suitable for specific tasks or domains. This underscores the importance of considering both the task and the domain when choosing a RAG workflow and LLM combination.

5 ANALYSIS

To discover the underlying characteristics of RALLMs beyond evaluation performance, we delve into a multifaceted analysis on the compatibility between RAG workflows and LLMs, the error types, and the trade-off between effectiveness and efficiency.

5.1 Matching Analysis

RQ4: *Which LLM best matches each RAG workflow?*

To understand the synergies between LLMs and RAG workflows, we compare the performances of all LLMs within each RAG workflow. The results, visualized in Figure 4, display the average performance of all RALLMs on different tasks, with each sub-figure representing a different workflow. Our findings are as follows:

(1) Within the ReAct workflow [53], GPT-4-1106 emerges as the best matching LLM across all six task types and domains, leading with a significant margin over other LLMs.

(2) However, within the PAL workflow [10], GPT-3.5-Turbo-1106 achieves results comparable to GPT-4-1106 across all levels. While the GPT series models generally outperform other LLMs, they do not surpass Llama2-13b on the three task levels within the Wikipedia domain. This observation suggests that the PAL workflow may enhance the performance of smaller 13b-size LLMs, such as Llama2-13b, more than larger LLMs like the GPT series.

(3) Regarding the DFSDT workflow [32], which requires a specific tool-using format akin to FC, only GPT-3.5-Turbo, GPT-4 and ToolLLaMA2-7B [32] can generate valid results. Among these, GPT-4-1106 outperforms the other two LLMs by a large margin.

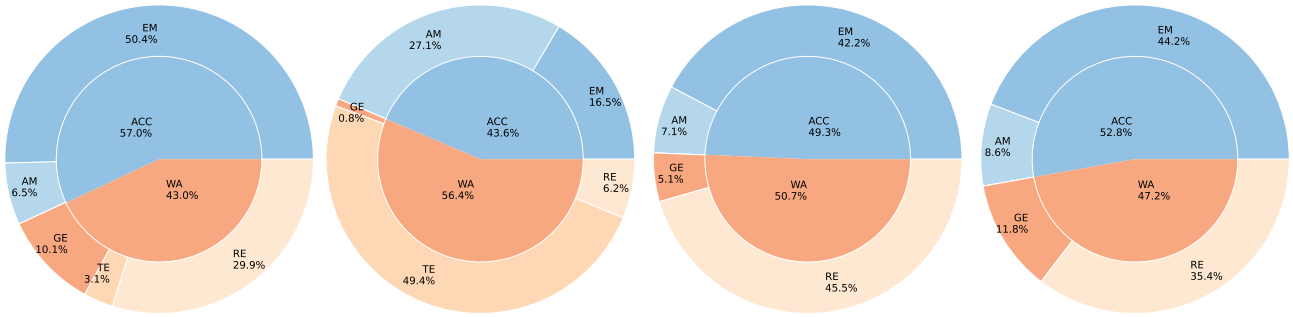
(4) Surprisingly, within the FC workflow, GPT-4 only surpasses GPT-3.5-Turbo on KU and KS tasks within the Aminer domain, while achieving similar or worse results on other tasks. This anomaly prompted us to analyse the error types associated with GPT-4.

5.2 Error Analysis

RQ5: *What types of errors does GPT-4 make across different workflows?*

In Section 3.4, we defined various error types that RALLM systems might encounter. Here, we aim to identify the distribution of these error types in GPT-4 across different RAG workflows.

Figure 5 visualizes the distribution of GPT-4's error types across the four RAG workflows. Notably, the PAL workflow exhibits the highest proportion of Answer Match errors among all workflows at 27.1%, while its Exact Match answers only account for 16.5%. This suggests that the knowledge retrieved by PAL does not contribute significantly to reasoning, and most correct answers originate from the inherent knowledge of the LLMs. Additionally, the tool-using



(a) ReAct (b) PAL (c) DFSDT (d) FC

Figure 5: Error distribution of GPT-4-preview-1104 with different 4 workflows on all tasks.

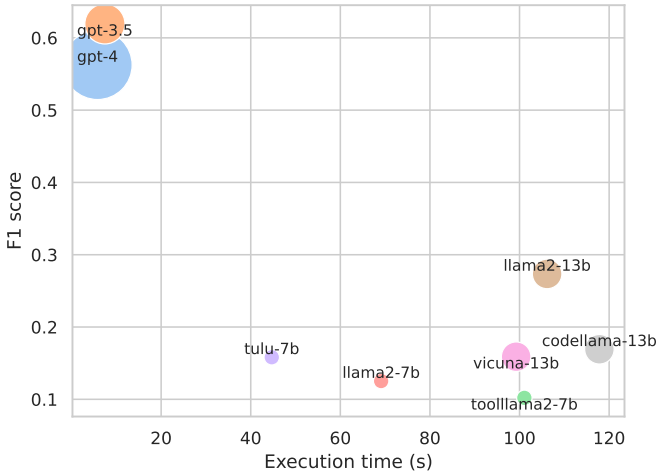


Figure 6: Average time cost and task performance of different LLMs with PAL workflow on aminer domain for deployment.

error rate for PAL is also the highest, reaching 49.4%, indicating that the API calling process is often interrupted, preventing successful retrieval of domain knowledge. As PAL workflow executes the API calling code only once, it cannot revise the code when encountering a tool execution error, which is critical in other workflows [31, 53] and explains why GPT-4 performs poorly with PAL.

Other RAG workflows, while more robust in their tool-using manners, require more inference time as they can interact with our query APIs for feedback and can execute multiple rounds of queries. Consequently, the ReAct, DFSDT, and FC workflows yield fewer tool-using errors than PAL, enabling them to retrieve domain knowledge more successfully. This results in fewer Answer Match errors and a greater number of Exact Match answers.

5.3 Deployment Analysis

RQ6: *Which system offers the best practical performance (both in terms of efficiency and effectiveness) within the specific domain?*

We are interested in assessing the practical performance of these RALLM systems, specifically in terms of execution time and the F1 score of the answers. For this study, we report the results of eight LLMs using the PAL workflow, as PAL's simple-turn interaction with the environment allows for faster execution compared to other workflows. Following the evaluation process of ToolBench [31], we evaluate efficiency using a single thread on a GPU (Nvidia A100) for each open-source LLM, and one single thread for each LLM called via the OpenAI API. The analysis results for the Aminer domain are illustrated in Figure 6, with additional results provided in Appendix C. Our findings are as follows:

GPT-4 and GPT-3.5 achieve superior F1 scores on Aminer and require less execution time than other open-source LLMs. This efficiency is likely due to the fact that while each open-source LLM is run on a single GPU, the GPT models are invoked through an API, which may leverage a cloud server with multiple GPUs.

When comparing only open-source LLMs, we observe that although Llama-13b ranks among the best in terms of F1 score, it is significantly slower than other LLMs, taking approximately 105 seconds per query in the Aminer domain. Among all the models, Tulu-7b strikes a commendable balance between efficiency (40-50 seconds per query) and effectiveness (F1 0.18). Intriguingly, as depicted in Figure 6, we find that open-source LLMs with larger parameters can markedly improve the F1 score while compromising on efficiency. This trade-off can guide developers in selecting the most suitable RALLM for their specific domain applications.

6 CONCLUSION

In this work, we presented the R-Eval toolkit, a comprehensive evaluation platform designed to address the existing gap in the systematic evaluation of Retrieval-Augmented Large Language Models (RALLMs). By offering a user-friendly, modular, and extensible interface, R-Eval facilitates the comparison and analysis of various RAG workflows and LLMs. Our study using this toolkit revealed significant variations in the performance of RALLMs across different tasks and domains, underscoring the importance of task- and domain-specific selection of RAG workflows and LLMs. As the field continues to evolve, we believe that R-Eval will benefit both researchers and industry professionals with a pivotal role in shaping the future of RALLMs and their domain-specific applications.

REFERENCES

[1] Sarabjot S Anand, David A Bell, and John G Hughes. 1995. The role of domain knowledge in data mining. In *Proceedings of the fourth international conference on Information and knowledge management*. 37–43.

[2] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712* (2023). https://arxiv.org/abs/2303.12712

[3] Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022. KQA Pro: A Dataset with Explicit Compositional Programs for Complex Question Answering over Knowledge Base. In *Proceedings of ACL*. 6101–6119. https://doi.org/10.18653/v1/2022.acl-long.422

[4] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, Vancouver, Canada, 1870–1879. https://doi.org/10.18653/v1/P17-1171

[5] Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. 2023. INSTRUCTEVAL: Towards Holistic Evaluation of Instruction-Tuned Large Language Models. *arXiv preprint arXiv:2306.04757* (2023). https://arxiv.org/abs/2306.04757

[6] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2023. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217* (2023).

[7] Yaxin Fan, Feng Jiang, Peifeng Li, and Haizhou Li. 2023. Grammargpt: Exploring open-source llms for native chinese grammatical error correction with supervised fine-tuning. In *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer, 69–80.

[8] Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, Fengzhe Zhou, Zhuo Han, Songyang Zhang, Kai Chen, Zongwen Shen, and Jidong Ge. 2023. LawBench: Benchmarking Legal Knowledge of Large Language Models. *arXiv preprint arXiv:2309.16289* (2023).

[9] Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Christian Petersen, Alexis Chevalier, and Julius Berner. 2023. Mathematical capabilities of chatgpt. *arXiv preprint arXiv:2301.13867* (2023). https://arxiv.org/abs/2301.13867

[10] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning*. PMLR, 10764–10799.

[11] Neel Guha, Julian Nyarko, Daniel E Ho, Christopher Ré, Adam Chilton, Aditya Narayana, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N Rockmore, et al. 2023. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. *arXiv preprint arXiv:2308.11462* (2023).

[12] Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active Retrieval Augmented Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, 7969–7992. https://doi.org/10.18653/v1/2023.emnlp-main.495

[13] Qiao Jin, Yifan Yang, Qingyu Chen, and Zhiyong Lu. 2023. Genegpt: Augmenting large language models with domain tools for improved access to biomedical information. *ArXiv* (2023).

[14] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406* (2022).

[15] Jan Kocoń, Igor Cichecki, Oliwier Kaszyca, Mateusz Kochanek, Dominika Szydło, Joanna Baran, Julita Bielaniewicz, Marcin Gruza, Arkadiusz Janz, Kamil Kanclerz, et al. 2023. ChatGPT: Jack of all trades, master of none. *arXiv preprint arXiv:2302.10724* (2023). https://arxiv.org/abs/2302.10724

[16] David R Krathwohl. 2002. A revision of Bloom's taxonomy: An overview. *Theory into practice* 41, 4 (2002), 212–218.

[17] Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115* (2022).

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

[19] Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models. https://arxiv.org/abs/2305.11747

[20] Jianquan Li, Xidong Wang, Xiangbo Wu, Zhiyi Zhang, Xiaolong Xu, Jie Fu, Prayag Tiwari, Xiang Wan, and Benyou Wang. 2023. Huatuo-26M, a Large-scale Chinese Medical QA Dataset. *arXiv preprint arXiv:2305.01526* (2023).

[21] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 3102–3116. https://doi.org/10.18653/v1/2023.emnlp-main.187

[22] Yangning Li, Shirong Ma, Xiaobin Wang, Shen Huang, Chengyue Jiang, Hai-Tao Zheng, Pengjun Xie, Fei Huang, and Yong Jiang. 2023. EcomGPT: Instruction-tuning Large Language Model with Chain-of-Task Tasks for E-commerce. *arXiv preprint arXiv:2308.06966* (2023).

[23] Zonglin Li, Ruiqi Guo, and Sanjiv Kumar. 2022. Decoupled context processing for context augmented language modeling. *Advances in Neural Information Processing Systems* 35 (2022), 21698–21710.

[24] Jiongnan Liu, Jiajie Jin, Zihan Wang, Jiehan Cheng, Zhicheng Dou, and Ji-Rong Wen. 2023. RETA-LLM: A Retrieval-Augmented Large Language Model Toolkit. *arXiv preprint arXiv:2306.05212* (2023).

[25] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172* (2023).

[26] Yikang Liu, Ziyin Zhang, Wanyang Zhang, Shisen Yue, Xiaojing Zhao, Xinyuan Cheng, Yiwen Zhang, and Hai Hu. 2023. ArguGPT: evaluating, understanding and identifying argumentative essays generated by GPT models. *arXiv preprint arXiv:2304.07666* (2023). https://arxiv.org/abs/2304.07666

[27] Zhiwei Liu, Weiran Yao, Jianguo Zhang, Le Xue, Shelby Heinecke, Rithesh Murthy, Yihao Feng, Zeyuan Chen, Juan Carlos Niebles, Devansh Arpit, et al. 2023. Bolaa: Benchmarking and orchestrating llm-augmented autonomous agents. *arXiv preprint arXiv:2308.05960* (2023).

[28] Allen Newell. 1982. The knowledge level. *Artificial intelligence* 18, 1 (1982), 87–127.

[29] OpenAI. 2023. GPT-4 technical report. *arXiv preprint arxiv:2303.08774* (2023). https://arxiv.org/pdf/2303.08774.pdf

[30] Hao Peng, Xiaozhi Wang, Shengding Hu, Hailong Jin, Lei Hou, Juanzi Li, Zhiyuan Liu, and Qun Liu. 2022. COPEN: Probing Conceptual Knowledge in Pre-trained Language Models. In *Proceedings of EMNLP*. 5015–5035. https://aclanthology.org/2022.emnlp-main.335

[31] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. *arXiv preprint arXiv:2307.16789* (2023).

[32] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789* (2023).

[33] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[34] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950* (2023).

[35] Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2023. Ares: An automated evaluation framework for retrieval-augmented generation systems. *arXiv preprint arXiv:2311.09476* (2023).

[36] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652* (2023).

[37] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[38] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 990–998.

[39] SM Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. 2024. A Comprehensive Survey of Hallucination Mitigation Techniques in Large Language Models. *arXiv preprint arXiv:2401.01313* (2024).

[40] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arxiv:2302.13971* (2023). https://arxiv.org/pdf/2302.13971.pdf

[41] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics* 10 (2022), 539–554. https://doi.org/10.1162/tacl_a_00475

[42] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Toronto, Canada, 10014–10037. https://doi.org/10.18653/v1/2023.acl-long.557

[43] Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. 2023. A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation. *arXiv preprint arXiv:2307.03987* (2023).

[44] Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, et al. 2023.

Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *arXiv preprint arXiv:2310.07521* (2023).
[45] Xidong Wang, Guiming Hardy Chen, Dingjie Song, Zhiyi Zhang, Zhihong Chen, Qingying Xiao, Feng Jiang, Jianquan Li, Xiang Wan, Benyou Wang, et al. 2023. Cmb: A comprehensive medical benchmark in chinese. *arXiv preprint arXiv:2308.08833* (2023).
[46] Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. 2023. How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources. *arXiv preprint arXiv:2306.04751* (2023).
[47] Yuanchun Wang, Jifan Yu, Zijun Yao, Jing Zhang, Yuyang Xie, Shangqing Tu, Huihui Yuan, Jingyao Zhang, Bowen Huang, Yuanyao Li, Juanzi Li, and Jie Tang. 2024. SoAy: A Service-oriented APIs Applying Framework of Large Language Models. https://github.com/RUCKBReasoning/SoAy.
[48] Yuanhao Wu, Juno Zhu, Siliang Xu, Kashun Shum, Cheng Niu, Randy Zhong, Juntong Song, and Tong Zhang. 2023. RAGTruth: A Hallucination Corpus for Developing Trustworthy Retrieval-Augmented Language Models. *arXiv preprint arXiv:2401.00396* (2023).
[49] Qianqian Xie, Weiguang Han, Xiao Zhang, Yanzhao Lai, Min Peng, Alejandro Lopez-Lira, and Jimin Huang. 2023. PIXIU: A Large Language Model, Instruction Data and Evaluation Benchmark for Finance. *arXiv preprint arXiv:2306.05443* (2023).
[50] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. 2023. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. *arXiv preprint arXiv:2304.13712* (2023). https://arxiv.org/abs/2304.13712
[51] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *Proceedings of EMNLP*. 2013–2018. https://doi.org/10.18653/v1/D15-1237
[52] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of EMNLP*. 2369–2380. https://doi.org/10.18653/v1/D18-1259
[53] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629* (2022).
[54] Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A Large-Scale Document-Level Relation Extraction Dataset. In *Proceedings of ACL*. 764–777. https://doi.org/10.18653/v1/P19-1074
[55] Jifan Yu, Xiaozhi Wang, Shangqing Tu, Shulin Cao, Daniel Zhang-Li, Xin Lv, Hao Peng, Zijun Yao, Xiaohan Zhang, Hanming Li, et al. 2023. KoLA: Carefully Benchmarking World Knowledge of Large Language Models. *arXiv preprint arXiv:2306.09296* (2023).
[56] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685* (2023).
[57] Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. ToolQA: A Dataset for LLM Question Answering with External Tools. *arXiv preprint arXiv:2306.13304* (2023).

A IMPLEMENTATION DETAILS

We have implemented an environment for each domain with several query APIs to retrieve the domain knowledge.

On Aminer domain, we have these APIs:

• **searchPerson**. The searchPerson function, which is based on the scholar entities' information in Aminer, receives the name, organization and interest of this intended scholar and returns the detailed information including person's id, citation number and publication number via fuzzy match.

• **searchPublication**. The searchPublication function, which is based on the publication entities' information in Aminer, receives the publication information and returns the related information including publication's id, title and publication year via fuzzy match.

• **getCoauthors**. The getCoauthors function, which is based on the relation information between scholar entities, receives the person id then returns the input scholar's coauthors and their detailed information including id, name and relation via exact match.

• **getPersonInterest**. The getPersonInterest function, which is based on the property information of scholar entities in Aminer, receives the scholar's id and returns a list of the person's interested research topics via exact match.

• **getPublication**. The getPublication function, which is based on the property information of publication entities in Aminer, receives the publication's id and returns its detailed information including the publication's abstract, author list and the number of citation via exact match.

• **getPersonBasicInfo**. The getPersonBasicInfo function, which is based on the scholar entities' property information in Aminer, receives the person's id of this intended scholar and returns the detailed information including person's name, gender, organization, position, short bio, education experience and email address via exact match. In fact, these information consists of the person's profile.

• **getPersonPubs**. The getPersonPubs function, which is based on the relation information between publication entities and scholar entities in Aminer, receives the person's id, and returns the detailed information including the publication's id, title, citation number and the authors' name list via exact match.

On Wikipedia domain, we have these APIs:

• **Search**. The Search function, which is based on the entities' page information in Wikipedia, receives the entity's name, and returns the page's abstract via fuzzy match while storing the other sections' information in the document store for the further usage. If there is no matching Wikipedia entity, the API will provide a list of possibly related entities for continuous searching.

• **Lookup**. The Lookup function, which is based on the previous stored entity information via search API on Wikipedia, receives the keyword and returns a list of relevant text segments from the document store via fuzzy match.

• **Finish**. This is a special function for stopping the searching process on Wikipedia.

Table 4: Hyper-parameters for open-source LLMs' inference in our evaluation.

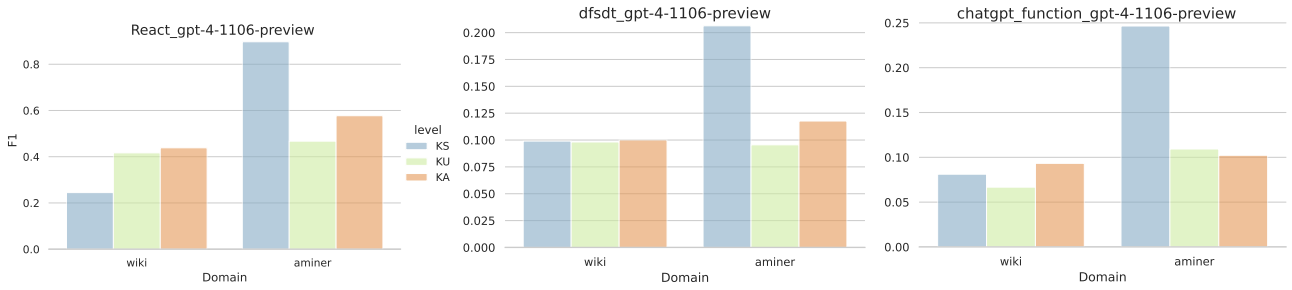
Process	Parameter	Value
Tokenization	max_length	2048
	truncation	True
	skip_special_tokens	True
Decoding	num_beams	1
	do_sample	False
	temperature	0
	stop sequence	</s>
	max_new_tokens	128

B EVALUATION DETAILS

In our study, we evaluate different RALLMs on various datasets and domains under a one-shot setting. On each task, we will give RALLMs an example of how to interact with the domain API to retrieve the knowledge and reasoning on it. For the fairness of comparison, different RALLMs share the same example (query and answer) while the retrieval and reasoning processes are customized

Table 5: Performance of different systems for each task on Wikipedia domain.

Workflow	LLM	wiki KS			wiki KU				wiki KA				
		1-1	1-2	Rank	2-1	2-2	2-3	Rank	3-1	3-2	3-3	3-4	Rank
ReAct	gpt-4-1106	23.1	25.8	1st	33.7	51.3	40.0	1st	59.7	64.6	27.3	23.8	1st
PAL	gpt-3.5-turbo	10.4	9.1	10th	12.7	65.0	43.0	2nd	12.1	9.4	5.1	12.0	13th
PAL	gpt-4-1106	7.9	4.6	17th	18.6	30.0	47.0	5th	30.3	12.1	14.2	17.5	5th
ReAct	llama2-7b-chat	21.2	20.9	2nd	2.0	55.0	26.9	9th	31.5	26.8	11.5	18.3	2nd
PAL	llama2-13b	19.1	13.5	3rd	1.3	67.0	45.4	3rd	25.0	18.5	8.0	29.0	4th
ReAct	gpt-3.5-turbo	11.2	12.5	7th	9.3	27.0	32.0	10th	43.1	30.1	7.6	3.8	3rd
ReAct	vicuna-13b	14.9	12.3	5th	4.3	57.1	26.0	8th	25.2	28.0	4.3	14.0	6th
PAL	tulu-7b	10.4	8.1	11th	2.7	34.8	54.2	7th	15.1	22.7	7.5	14.8	7th
PAL	vicuna-13b	8.4	5.4	15th	1.0	49.9	46.4	4th	11.9	11.2	2.9	13.3	12th
ReAct	llama2-13b	16.2	11.5	4th	0.2	30.0	22.0	11th	20.1	24.6	4.0	6.5	8th
PAL	llama2-7b-chat	4.0	3.2	21th	1.3	60.3	33.4	6th	2.9	2.4	0.9	3.3	21th
PAL	codellama-13b	6.0	6.0	18th	0.4	11.7	32.0	14th	6.6	12.9	4.5	9.5	16th
PAL	toolllama2-7b	9.9	10.6	8th	2.2	40.5	6.1	12th	8.5	19.5	3.5	8.9	10th
ReAct	tulu-7b	14.7	11.8	6th	2.0	5.0	6.0	19th	18.0	23.3	1.4	10.5	9th
DFSDT	gpt-4-1106	10.8	9.0	9th	9.0	8.2	12.2	15th	10.7	18.4	7.9	3.1	11th
FC	gpt-4-1106	8.2	8.0	13th	9.9	3.3	6.8	18th	10.9	15.0	8.1	3.3	14th
FC	gpt-3.5-turbo	8.0	7.2	14th	2.1	4.0	21.0	16th	12.4	15.1	5.7	3.4	15th
ReAct	toolllama2-7b	5.8	1.5	20th	0.0	44.0	1.0	13th	4.4	12.1	0.0	6.0	18th
DFSDT	gpt-3.5-turbo	6.8	6.2	16th	0.9	4.0	3.7	20th	7.6	10.1	2.9	1.3	19th
ReAct	codellama-13b	7.7	9.5	12th	0.0	13.0	9.0	17th	5.7	9.0	0.0	8.7	17th
DFSDT	toolllama2-7b	4.5	4.0	19th	0.9	4.1	0.3	21th	1.8	8.7	2.1	4.8	20th



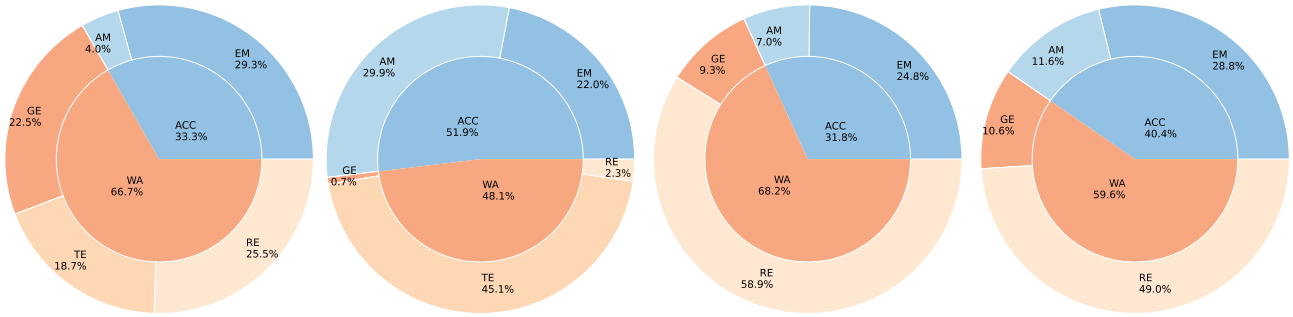
(a) ReAct (b) DFSDT (c) FC

Figure 7: Average performance of GPT-4 with different 3 workflows on all tasks.

for each RAG workflow. Note that we also provide an example pool with at least 1.8k cases for each task, as shown in Table 1, where the user of our toolkit can adjust the number of used examples to fit their own few-shot settings.

The models participating in the evaluation fall into two categories: closed-source models that generate responses via API calls, and open-source models that are directly deployed for inference, with a temperature parameter set to 0. The other hyper-parameters of open-source models are summarized in Table 4. To load open-source models, we employ the widely adopted *PyTorch* and *transformers* libraries. The evaluation experiments are executed on an Ubuntu 20.04.4 server, furnished with 112 Intel Xeon(R) Platinum 8336C CPU cores, and complemented by graphic cards incorporating 8 NVIDIA A100 SXM 80GB GPUs. Furthermore, the software environment includes CUDA version 11.4, Python version 3.9.17, PyTorch version 2.1.2, and the transformers library version 4.28.1.

Given that all tasks are arranged in open-ended generative question-answering formats, we choose F1 score as evaluation metrics to reflect the model's genuine performance. The metric is specifically tailored to the characteristics of different task levels and is used for post-processing the answers. For tasks that primarily use F1 as the evaluation metric, we employ a relaxed version of F1 (token match rate). Specifically, after tokenizing the model's predicted results and the reference answers using the GPT2Tokenizer [33], we calculate whether each token in the prediction appears in the corresponding position of the gold standard.



(a) ReAct (b) PAL (c) DFSDT (d) FC

Figure 8: Error distribution of GPT-3.5-turbo-1104 with different 4 workflows on all tasks.

C EXTRA EXPERIMENT RESULTS

C.1 Wiki Domain Results

RQ7: *How effective are RALLMs on wiki domain?*

The performance of different RALLMs on Wikipedia domain is demonstrated in Table 5. In the Wikipedia domain, which contains over 6.6 million English articles, some models with ReAct workflow, such as ReAct with GPT-4-1106, demonstrate strong performance across all three levels of tasks and win the first place on all these tasks. To illustrate the performance difference among RAG workflows, we display the average performance of GPT-4-1106 model with ReAct, DFSDT and FC workflow in Figure 7. The DFSDT and FC perform badly on the three levels' tasks of Wikipedia domain, where their F1 scores are all below 0.1, which is much lower than the ReAct workflow's scores even though they are all based on GPT-4-1106. This suggests that the ReAct workflow is effective at retrieving, understanding, and applying knowledge in a broad open-domain context.

C.2 Error Analysis for GPT-3.5-turbo

RQ8: *What kinds of errors does gpt-3.5 make on different workflows?*

As shown in Figure 8, the error distribution of gpt-3.5-turbo is different from gpt-4 in Figure 5. We find that the PAL workflow with gpt-3.5-turbo has a larger Answer Match and EM portion than the PAL workflow with gpt-4. While their AM portion are similar (29.9% vs. 27.1%), gpt-3.5-turbo has a significantly larger EM portion (22.0%) than gpt-4 (16.5%). However, for other workflows (ReAct, DFSDT, FC), gpt-3.5-turbo perfoms much worse than gpt-4. The main error gpt-3.5-turbo encountered is RE (reasoning error), especially on DFSDT and FC. These results reflect that gpt-3.5-turbo may be more dependant on the retrieved knowledge for reasoning while gpt-4 are better at inner knowledge. This provides a crucial insight about the difference between gpt-3.5-turbo and gpt-4 on their response and error types with different workflows.

C.3 Deployment Analysis for Wiki Domain

RQ9: *Which system provides the best practical performance (efficiency and effectiveness) on wiki domain?*

We are also interested in evaluating the practical performance of these RALLM systems on Wikipedia domain in terms of execution time and answer F1 score. We report the results of 8 LLMs with PAL workflow in Figure 9, since PAL has the simple-turn interaction with the environment, making it much faster than other workflow. Surprisingly, we find that although GPT-4 and GPT-3.5 get better F1 scores than other models on Aminer, they are not the best model in the Wikipedia domain.

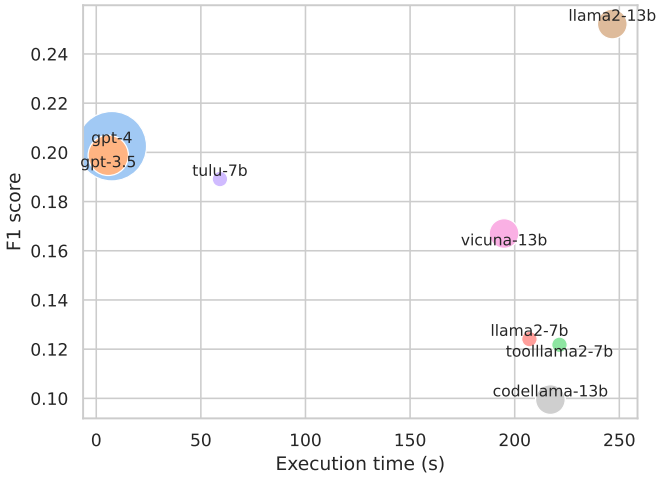


Figure 9: Average time cost and task performance of different LLMs with PAL workflow on wiki domain for deployment.

If we only compare open-source LLMs, we find that although llama-13b is the best on F1 score, which is even higher than GPT-4, it is much slower other LLMs (250s per query on the Wikipedia domain). Among all the models, tulu-7b still achieves a good trade-off between efficiency (50s per query) and effectiveness (0.19 F1 score) on the Wikipedia domain. Interestingly from Figure 9, we find the open-source LLMs that based on Llama2 are generally worse on the efficiency than those Llama-based LLMs. For example, vicuna-13b is fine-tuned on Llama while Llama2-7b is in the Llama2 family and has less parameters than vicuna-13b. But, llama2-7b is slower than vicuna-13b. This finding can help developers choosing the suitable LLM for their domain applications.