LegalAgentBench: Evaluating LLM Agents in Legal Domain

Haitao Li[1*†], **Junjie Chen**[1*†] **Jingli Yang**[2], **Qingyao Ai**[1‡], **Jia Wei**[2], **Youfeng Liu**[2], **Kai Lin**[3]
Yueyue Wu[1‡] **Guozhi Yuan**[4], **Yiran Hu**[5], **Wuyue Wang**[6], **Yiqun Liu**[1], **Minlie Huang**[1]
[1]Department of Computer Science and Technology, Tsinghua University, [2]Zhipu AI,
[3]Shanghai Amarsoft Enterprise Credit Information Service Co.,Ltd,
[4]Central South University,[5]University of Waterloo, [6]University of Notre Dame
liht22@mails.tsinghua.edu.cn

Abstract

With the increasing intelligence and autonomy of LLM agents, their potential applications in the legal domain are becoming increasingly apparent. However, existing general-domain benchmarks cannot fully capture the complexity and subtle nuances of real-world judicial cognition and decision-making. Therefore, we propose **LegalAgentBench**, a comprehensive benchmark specifically designed to evaluate LLM Agents in the Chinese legal domain. LegalAgentBench includes 17 corpora from real-world legal scenarios and provides 37 tools for interacting with external knowledge. We designed a scalable task construction framework and carefully annotated 300 tasks. These tasks span various types, including multi-hop reasoning and writing, and range across different difficulty levels, effectively reflecting the complexity of real-world legal scenarios. Moreover, beyond evaluating final success, LegalAgentBench incorporates keyword analysis during intermediate processes to calculate progress rates, enabling more fine-grained evaluation. We evaluated eight popular LLMs, highlighting the strengths, limitations, and potential areas for improvement of existing models and methods. LegalAgentBench sets a new benchmark for the practical application of LLMs in the legal domain, with its code and data available at https://github.com/CSHaitao/LegalAgentBench.

1 Introduction

Recent advances in large language models (LLMs) have significantly increased the field of artificial intelligence (Achiam et al., 2023; Bai et al., 2023; Touvron et al., 2023). With their expansive neural networks and vast training datasets, LLMs have made remarkable strides in various natural language processing tasks, such as text generation and machine translation (Li et al., 2024e,c; Zhao et al., 2024). At the same time, the rapid evolution of LLMs is transforming the traditional legal industry, empowering legal professionals to handle tasks such as legal research, contract drafting, and case analysis with greater efficiency. As a result, LLMs are rapidly becoming indispensable tools in modern legal workflows (Cui et al., 2023; Li et al., 2024d; Guha et al., 2024; Li et al., 2023).

Despite their immense potential, LLMs still face challenges in tackling complex legal issues, as real-world legal tasks often require multistep reasoning and specialized expertise beyond their current capabilities. A promising solution lies in the development of LLM-as-Agent systems (Dorri et al., 2018; Liu et al., 2023b). These agents can engage in step-by-step reasoning and acquire specialized knowledge through iterative interactions with external tools. Their impressive capabilities have attracted significant interest from both academia and industry (Qin et al., 2023; Li et al., 2024a).

Although LLM agents show great promise, the lack of standardized benchmarks to evaluate their performance in legal scenarios is a major challenge. Existing frameworks, such as AgentBench (Liu et al., 2023b) and ToolBench (Qin et al., 2023), are effective in assessing LLM agents in general domains. However, the insights gained from these evaluations often have limited relevance in highly specialized fields such as the legal domain (Li et al., 2024d). Moreover, existing datasets in the legal domain often focus on relatively basic tasks, such as legal case retrieval (Li et al., 2024f,b) or judgment prediction (Li et al., 2024d). In contrast, legal practice is significantly more complex, involving in-depth case analysis, legal reasoning, and comprehensive judgments based on a vast body of laws and precedents. Current datasets and evaluation systems fall short of thoroughly testing these advanced, multidimensional legal capabilities.

To fill this gap, we developed **LegalAgent-**

[*] Equal contributions.
[†] Work done when HT and JJ interned at Zhipu AI.
[‡] Corresponding author

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2322–2344

Bench, a comprehensive benchmark designed to evaluate the capabilities of LLM agents in the Chinese legal domain. Based on real-world legal needs, it includes 17 specialized corpora and 37 tools to interact with external knowledge. Within this framework, LLM agents coordinate and utilize these tools to address specific legal tasks.

LegalAgentBench is highlighted in the following three aspects:

- **Focus on Authentic Legal Scenarios:** LegalAgentBench is the first dataset to evaluate LLM agents in legal scenarios. It requires LLMs to demonstrate a solid understanding of legal principles, enabling them to appropriately select and utilize tools to solve complex legal problems. This represents a significant step forward in advancing the application of LLM agents in legal scenarios.
- **Diverse Task Types and Difficulty Levels:** LegalAgentBench adopts a scalable task construction framework aimed at comprehensively covering various task types and difficulty levels. Specifically, we construct a planning tree based on the dependencies between the corpus and tools, and select tasks through hierarchical sampling and a maximum coverage strategy. Finally, we constructed 300 distinct tasks, including multi-hop reasoning and writing tasks, to comprehensively evaluate the LLM's capabilities.
- **Fine-Grained Evaluation Metrics:** Rather than relying solely on final success rates as evaluation criteria, LegalAgentBench introduces the process rate through the annotation of intermediate steps, enabling fine-grained evaluation. This approach provides deeper insights into an agent's capabilities and identifies areas for improvement beyond the final result.

We evaluated a variety of commercial and open-source LLM agents, identifying several potential weaknesses in their current capabilities. These observations provide valuable insights, inspire new ideas, and pinpoint areas for further research and improvement.

2 Related Work

2.1 LLM Agents

Recently, large language models (LLMs) have achieved significant advancements in their application as intelligent agents. Leveraging the capabilities in natural language understanding and generation, LLM agents can efficiently solve complex tasks by appropriately invoking external tools such as calculators, search engines, and domain-specific APIs (Yao et al.; Wang et al., 2024, 2023; Shinn et al., 2024; Chu et al., 2024).

When faced with complex tasks, LLM agents need to devise appropriate plans and strategies to ensure efficient execution. Chain-of-Thought (CoT) (Wei et al., 2022) reasoning, a pioneering technique in this domain, enhances the reasoning capabilities of agents by decomposing challenging reasoning tasks into smaller, more manageable steps. Building upon this foundation, a series of advanced reasoning strategies have emerged, offering new perspectives for task planning and problem-solving. For instance, ReAct (Yao et al.) decouples reasoning and action, alternating between thinking steps and action steps, thereby significantly improving the planning efficiency for complex tasks.

Furthermore, the capabilities of LLM agents are further enhanced through deep integration with external tools. Methods such as HuggingGPT (Shen et al., 2024) exemplify this approach by positioning LLMs as controllers that decompose complex tasks into subtasks, invoke appropriate specialized models, and integrate the results to produce a final response. The LLM+P (Liu et al., 2023a) approach combines LLMs with a symbolic planner based on the Planning Domain Definition Language (PDDL), leveraging LLMs to formulate problems in PDDL syntax and employing a planning solver to generate solutions.

2.2 Benchmarks on LLM Agents

With the rapid advancement of LLM agents, the demand for evaluating their reasoning and decision-making capabilities in complex tasks has grown significantly. Numerous benchmarks have been developed, providing essential references for the systematic study of LLM agents (Liu et al., 2023b; Zhuang et al., 2023; Ye et al., 2024; Yao et al., 2024; Huang et al., 2023; Xie et al., 2023).

AgentBench (Liu et al., 2023b) is a multidimensional benchmark platform that spans eight distinct environments, including operating systems, databases, and knowledge graphs. It systematically evaluates the performance of LLM agents in multi-turn open-ended generation settings. AgentBoard (Ma et al., 2024) focuses on the systematic evaluation of LLMs in multi-turn interactions.

It introduces fine-grained progress rate metrics and multidimensional analysis tools to comprehensively assess LLMs' multitasking capabilities in partially observable environments, offering a scientific framework for evaluating interactive LLM performance. ToolQA (Zhuang et al., 2023) covers eight domains and 13 tools, with tasks specifically designed to require external tools and reference materials for problem-solving. This avoids reliance solely on the model's internal knowledge, establishing a new benchmark for assessing LLMs' tool-using abilities. T-Eval (Chen et al., 2024) decomposes the tool utilization abilities of LLMs into six dimensions: planning, reasoning, retrieval, understanding, instruction execution, and result evaluation. This framework comprehensively measures LLMs' performance in using external tools across multiple levels.

These benchmarks provide diverse tools for exploring the capability boundaries of LLM agents. However, they primarily focus on evaluating LLMs in general domains, lacking systematic guidance and targeted frameworks for vertical domains such as law and medicine. To address this gap, this paper introduces a dataset specifically designed to evaluate LLM agents in the legal domain, offering valuable insights and support for research and practice.

3 LegalAgentBench

3.1 Overview

LegalAgentBench consists of three key components: the environment, tools, and tasks. LLM agents rely on these tools to interact with the environment and tackle specific tasks. We define the environment as a text-based corpora, where both the observation and action spaces are represented in natural language. Specifically, the environment consists of 17 distinct corpora, 14 of which are tabular databases for lookup, while 3 are document collections for retrieval. To facilitate interaction with the environment, we provide 37 specialized tools for tasks such as text retrieval, database operations, and mathematical computations.

In addition, LegalAgentBench includes 300 tasks of varying difficulty levels and types. Figure 1 illustrates a task example. The key_answer represents the keywords in the answer, used to evaluate the success rate of the final result. The key_middle refers to the keywords in the intermediate steps of solving the task, providing a more granular evaluation. These keywords are derived from the observations returned by successful tool calls. The Path refers to the correct solution path for solving the task. A longer path signifies a higher level of difficulty associated with the task.

- **Query:** I would like detailed information about the court system in Beijing. Could you please tell me which courts are located in the area where the Beijing First Intermediate People's Court is situated?
- **Answer:** The Beijing First Intermediate People's Court and the Beijing Shijingshan District People's Court are located in this area.
- **Key_answer:** ["Beijing First Intermediate People's Court", "Beijing Shijingshan District People's Court"]
- **Key_middle :** ["Shijingshan District, Beijing"]
- **Path:** get_court_info----get_court_info_list

Figure 1: A task example in LegalAgentBench (translated from Chinese).

Upon receiving the task, the LLM agent gradually selects the appropriate tools and obtains corresponding feedback. After receiving feedback, the LLM agent updates its internal state based on the observation and then determines the next action. The agent's actions typically involve selecting the appropriate tool and inputting the necessary parameters, or adjusting the query strategy based on the feedback. Each action generates a new observation, and this cycle continues until the agent completes the task or reaches the predefined termination condition.

3.2 Corpora and Tools

In LegalAgentBench, we developed 17 real-world datasets, including 14 tabular databases designed for lookups and 3 document repositories intended for retrieving relevant information. Table 1 presents the basic information and a brief overview of these corpora. More details can be found in Appendix B.

Specifically, we gather publicly available real-world data, encompassing information about courts, law firms, listed companies, and their associated legal cases. Additionally, we compile legal knowledge, articles, and guiding cases to create comprehensive and searchable corpora. These corpora are sourced from real-world scenarios and can evolve over time, mitigating the risk of LLM overfitting. Due to the sensitivity of the legal domain, we discuss the license and ethical considerations in Appendix A.

To obtain information from these reference corpora, we designed 37 tools that are available to the LLM Agents. These tools are primarily divided into the following categories:

ID	Corpus	Format	Size	Brief overview
1	CompanyInfo	Tabular Database	695	Basic information of listed companies
2	CompanyRegister	Tabular Database	10,125	Registration information of listed companies
3	SubCompanyInfo	Tabular Database	9,433	Investment information of listed companies
4	LegalDoc	Tabular Database	24,372	Legal cases involving listed companies
5	LegalAbstract	Tabular Database	1,200	Summary information of legal cases
6	CourtInfo	Tabular Database	3,413	Basic information of courts
7	CourtCode	Tabular Database	3,348	Levels and administrative division codes of courts
8	LawfirmInfo	Tabular Database	4,768	Basic information of law firms
9	LawfirmLog	Tabular Database	101	Service information of law firms
10	AddrInfo	Tabular Database	19,533	Province, city, and district corresponding to the address
11	RestrictionCase	Tabular Database	46	Cases involving restrictions on high consumption
12	FinalizedCase	Tabular Database	119	Cases closed upon final enforcement
13	DishonestyCase	Tabular Database	13	Cases involving dishonest judgment debtors
14	AdministrativeCase	Tabular Database	443	Cases involving administrative penalty
15	LegalKonwledge	Retrieval Corpus	26,951	Knowledge from legal books
16	LegalArticle	Retrieval Corpus	55,347	Legislatively enacted legal articles
17	LegalCases	Retrieval Corpus	2,370	Officially published guiding cases

Table 1: Basic information of the corpora.

- **Text Retrievers:** These tools are responsible for retrieving content relevant to a given query from document repositories. We use Embedding-3 [1] as the default retriever. Additionally, users can integrate more advanced retrieval models to further improve query results. LegalAgentBench includes three text retrievers. For each retriever, the LLM can use the *Number* parameter to specify the number of relevant documents to be returned.

- **Mathematical Tools:** These tools perform basic arithmetic operations such as addition, subtraction, multiplication, and division. Additionally, they can handle more complex tasks like sorting data, and computing maximum or minimum values. There are five mathematical tools in LegalAgentBench.

- **Database Tools:** These tools interact with specific databases to extract content from particular columns based on predefined queries. They allow the LLM agents to access structured data, providing tailored responses that match the criteria defined in the query. There are 28 database tools in LegalAgentBench. For each Database Tool, the LLM can use the parameter *Column* to control the attributes returned from the corresponding tabular database.

- **System Tools:** The System tool currently only includes *Finish*, which parses the execution feedback and returns the answer to complete the task.

Due to space constraints, detailed descriptions of each tool, including their inputs, outputs, and usage examples, are provided in Appendix C.

[1] https://bigmodel.cn/

3.3 Tasks

3.3.1 Task Definition

LegalAgentBench employs a unified framework to standardize all tasks, offering a formalized definition of the agent's interaction process in legal scenarios. At each time step t, the agent performs an action a_t, receives an observation o_t, and updates its current state s_t accordingly. This iterative process continues until the task is successfully completed or the maximum iteration limit T is reached.

s_t represents the agent's perceptual state at time t, encompassing the acquired contextual information and the environmental state. The state s_t is updated according to the following formula:

$$s_{t+1} = u(s_t, a_t, o_t),$$

where u is the state update function, responsible for integrating the agent's actions and feedback into a new perceptual state.

a_t represents the specific action taken by the agent in the state s_t, including tool invocation or information generation. The action a_t is determined by the state s_t and the sequence of past observations.

$$a_t = \pi(s_t, o_1, o_2, ..., o_{t-1}),$$

where π is the agent's decision-making policy.

o_t is the feedback information received by the agent after invoking a tool, including successful results or error messages.

$$o_t = f(a_t)$$

Here, f is the feedback function. If a_t is a valid action, it returns the result of the operation; otherwise, it returns an error message.

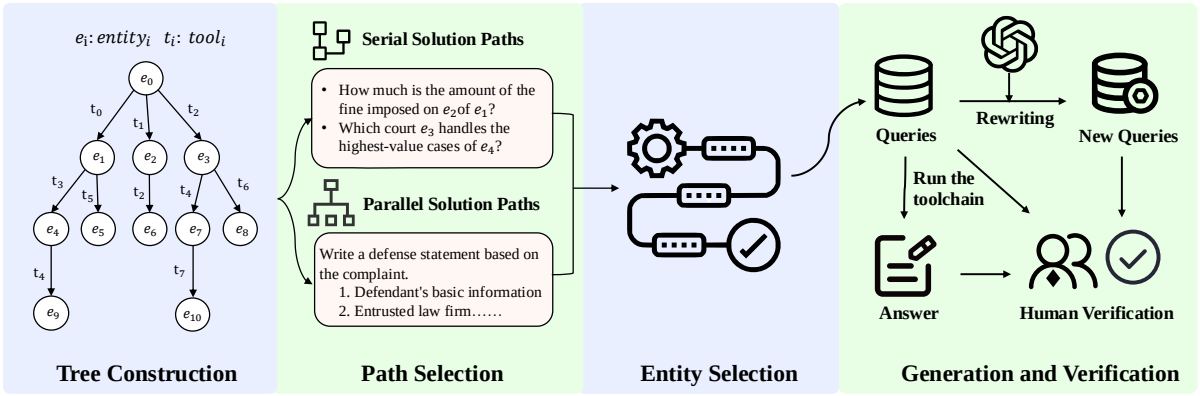

Figure 2: The overview of the task construction process in LegalAgentBench.

3.3.2 Task Construction

During the task generation phase, our primary goal is to leverage corpus and tools to construct a diverse set of questions. The core principle is to ensure that tasks encompass a wide range of difficulty levels and types, enabling a comprehensive evaluation. In this section, we propose a scalable task construction process that can be extended to incorporate new knowledge bases and tools. The entire process is illustrated in Figure 2.

Planning Tree Construction: To better organize and structure the task, we first construct a planning tree based on the call relationships between the available tools. The root node of the tree represents the unknown entity, which typically corresponds to the starting point of the task. Each branch of the tree corresponds to a tool that can be utilized by the entity, while the child nodes contain the information obtained after invoking these tools. For each child node, subsequent tools may be called progressively, forming deeper branches. In this planning tree, each path represents a potential solution pathway, and the information at the leaf nodes corresponds to the final requirements of the task.

Path Selection: We select different paths from the planning tree to construct tasks that cover a variety of types and difficulty levels. In the planning tree, the depth of each branch represents the complexity of the task solution, while the breadth of all branches reflects the diversity of task types. Therefore, we extract solution paths layer by layer to ensure coverage of different difficulty levels. Moreover, we minimize overlap between solution paths for tasks of different difficulty levels, expanding the coverage of all branches to encompass a broader range of task types. We ultimately developed serial solution paths ranging from 1-hop to 5-hop.

In addition to using tools sequentially, our solution paths also incorporate the parallel use of tools. We introduce the task of writing a defense document as a typical example. In this scenario, the LLM must query basic information about the plaintiff, defendant, and their lawyer based on a template, while simultaneously retrieving fundamental legal knowledge and relevant articles to formulate a defense against the complaint. In Appendix D, we provide examples for better understanding.

Entity Selection: After determining the solution path, we select entities to formulate the complete question. It is important to note that not all initial entities can complete the intended solution path, as there may be cases where the tool call returns no result. To address this issue, we iterate through all possible entities and select two successfully executed entities for each path.

Question Rewriting: After the above steps, we obtain the appropriate entities and can automatically generate multi-hop questions such as "What is $entity_3$ of $entity_2$ of $entity_1$?". However, this type of question not only deviates from actual usage habits but also directly exposes the solution path to the model. To better align with real-world scenarios and obscure the solution path, we rephrase the questions using GPT-4 to make them more flexible and in line with human needs. The specific prompt can be found in Appendix D.

Answers Generation: For each question, since the entities and the toolchain used in the solution path are known, we can programmatically extract the answer from the reference corpora through the appropriate parameterized toolchain. This enables the automatic generation of correct answers, even for complex multi-hop reasoning tasks.

Human Verification: We manually validate all questions, solution paths, and answers in the tasks. This involves revising incompatible queries and tools, refining questions that deviate from everyday

Attribute	1-hop	2-hop	3-hop	4-hop	5-hop	Writing	ALL
# Task	80	80	60	40	20	20	300
Avg. length per query	88.29	87.90	99.37	118.33	110.25	1059.95	160.65
Avg. length per answer	74.20	40.84	45.53	63.48	86.20	678.75	99.24
# Avg. key_answer per query	1.88	1.44	1.20	1.40	2.25	10.25	2.14
Avg. length of key_answer	10.59	5.94	6.07	6.59	6.93	12.58	9.28
# Avg. key_middle per query	0.13	1.45	2.87	4.78	5.60	6.20	2.42
Avg. length of key_middle	9.20	9.72	10.95	11.35	11.25	7.21	10.24

Table 2: Detailed Statistics of LegalAgentBench Tasks.

usage, and addressing queries related to specialized tools. Detailed guidelines for the validation process are provided in Appendix G.

3.3.3 Task Evaluation

Given the high accuracy requirements in the legal domain, we primarily evaluate performance using keyword matching to calculate the success rate. Specifically, we extract keywords from tool call results and record them as key_answer. The overlap between the agent's output and these keywords serves as a measure of its capabilities.

However, relying solely on the success rate fails to capture subtle differences, as it cannot distinguish between tasks that are nearly completed and those that are unfinished. To address this issue, we provide keywords not only for the final answers but also for the intermediate solution steps. Using these keywords, we calculate the progress rate, enabling a more fine-grained evaluation of the agent's performance at various stages of task completion. We provide the detailed calculation for the success rate and progress rate in Appendix E.

3.3.4 Task Statistics

After careful human verification, LegalAgentBench includes a total of 300 tasks across 6 different types. Table 2 presents the detailed statistics of the tasks. Overall, LegalAgentBench has a well-balanced distribution of difficulty and task types, making it an effective tool for evaluating the capabilities of agents in the legal domain.

4 Experiment

4.1 Experiment Setup

4.1.1 Baselines

We evaluated eight well-known LLMs on LegalAgentBench: GLM-4 (GLM et al., 2024), GLM-4-Plus (GLM et al., 2024), LLaMA3.1-8B-instruct (Touvron et al., 2023), Qwen-max (Bai et al., 2023), Claude-sonnet (claude-3-5-sonnet-20241022), GPT-3.5 (gpt-3.5-turbo-1106), GPT-4o-mini (gpt-4o-mini-2024-07-18), and GPT-4o (gpt-4o-2024-08-06). Except for LLaMA3.1-8B-Instruct, other LLMs are evaluated through API calls. To ensure the reproducibility of the results, we set the temperature for all LLMs to 0.

For each LLM, we implemented three different methods. 1) **Plan-and-Solve** (Wang et al., 2023): Outline a complete plan and execute it step by step. 2) **Plan-and-Execute** (Topsakal and Akinci, 2023): Develop a multi-step plan and complete it sequentially. After completing a task, the LLM can reassess the plan and make appropriate adjustments. 3) **ReAct** (Yao et al.): Perform reasoning incrementally through the "thought-action-observation" process, integrating reasoning and tool usage.

When given a task, the model first determines which tools are needed, and then uses the selected tools to gradually solve the task. When the LLM outputs *Finish* or reaches the maximum iteration limit $T = 10$, it summarizes the current trajectory and provides the final answer. We included three examples for each process to guide the model in using the tools and following the specified output format. Additional implementation details are available in Appendix E.

4.1.2 Metrics

We apply three evaluation metrics, **Success Rate**, **Process Rate**, and **BERT-Score** (Zhang et al., 2019), to comprehensively evaluate the performance. Specifically, the success rate calculates the proportion of key_answer elements included in the LLM's answer. The process rate further incorporates key_middle, measuring the ratio of key_middle and key_answer elements present in the answer. Moreover, BERTScore is applied to compute the textual similarity between the generated answer and the reference answer, thus assessing the quality and accuracy of the output. In addition to these metrics, we also report the number of tokens consumed by the LLMs as a reference for efficiency. Due to space limitations, we report only the success rate in the main result. More results can be found in the Appendix F.

Model	Method	1-hop	2-hop	3-hop	4-hop	5-hop	Writing	ALL	Tokens
GLM-4	P-S	0.7588	0.4229	0.1806	0.3708	0.1750	0.5778	0.4509	5,100,468
	P-E	0.7838	0.4042	0.2056	0.3083	0.1600	0.5469	0.4461	9,849,924
	ReAct	0.8787	0.6771	0.4167	0.3875	0.2433	0.5937	0.6057	11,920,863
GLM-4-Plus	P-S	0.8519	0.4667	0.4167	0.3583	0.1167	0.7522	0.5406	5,657,827
	P-E	0.8419	0.5000	0.3667	0.3458	0.1167	0.7679	0.5363	9,422,692
	ReAct	0.9131	0.8104	0.6417	0.6167	0.4300	0.7659	0.7499	11,739,861
LLaMa3.1-8B	P-S	0.3406	0.1333	0.0333	0.0375	0.1083	0.2382	0.1612	9,279,701
	P-E	0.3510	0.1042	0.0250	0.0500	0.0667	0.3484	0.1607	13,649,741
	ReAct	0.6019	0.1542	0.0750	0.0708	0.0600	0.0872	0.2359	50,661,127
Qwen-max	P-S	0.8469	0.4958	0.4083	0.3792	0.2333	0.4836	0.5381	4,800,345
	P-E	0.8594	0.5896	0.3583	0.4083	0.3017	0.5539	0.5695	9,884,307
	ReAct	0.9062	0.7917	0.6333	0.5833	0.6083	0.6662	0.7422	11,473,873
Claude-sonnet	P-S	0.8137	0.6354	0.4833	0.3750	0.2400	0.8395	0.6051	7,100,962
	P-E	0.8700	0.6771	0.5333	0.4667	0.4717	0.8610	0.6703	13,566,119
	ReAct	0.8950	0.6979	0.4750	0.4792	0.4567	0.6567	0.6579	32,878,858
GPT-3.5	P-S	0.4906	0.2396	0.0500	0.1000	0.0667	0.0333	0.2247	5,007,391
	P-E	0.4906	0.2062	0.0667	0.0625	0.0500	0.0405	0.2135	9,597,807
	ReAct	0.6421	0.2854	0.1167	0.1000	0.1333	0.0852	0.2986	11,357,664
GPT-4o-mini	P-S	0.7117	0.3375	0.2750	0.2583	0.1250	0.6314	0.4196	5,482,556
	P-E	0.7444	0.3771	0.3250	0.2417	0.1417	0.6681	0.4503	10,861,492
	ReAct	**0.9333**	0.6500	0.4000	0.4208	0.2583	0.6087	0.6161	13,332,418
GPT-4o	P-S	0.7856	0.4437	0.2750	0.1875	0.2250	0.7971	0.4760	4,153,333
	P-E	0.8106	0.3979	0.3167	0.2167	0.2767	**0.8643**	0.4906	7,261,238
	ReAct	0.9262	**0.8396**	**0.7500**	**0.6417**	**0.6117**	0.6541	**0.7908**	11,206,957

Table 3: The success rate of different baselines on LegalAgentBench. P-S represents the Plan-and-Solve method, and P-E represents the Plan-and-Execute method. The best results are highlighted in bold.

4.2 Main Results

The performance comparison between different LLMs and baselines on LegalAgentBench is shown in Table 3. More experimental results can be found in Appendix F. Based on the experimental results, we draw the following observations.

- **Comparing Different LLMs.** In experiments across different LLMs, GPT-3.5 and LLaMA3.1-8B demonstrated poor performance, with success rates on LegalAgentBench below 30%. This may stem from their limited ability to effectively use tools, restricting their problem-solving capacity in complex legal tasks. GLM4, GLM4-Plus, Qwen-Max, and GPT-4o-mini consumed tokens at similar levels. However, GLM4-Plus and Qwen-Max demonstrated superior performance. Claude-Sonnet also achieved competitive results, delivering the best performance under both P-S and P-E methods. However, it often required more tokens compared to other LLMs. Under the ReAct method, GPT-4o achieved the best performance with relatively fewer tokens, reaching a success rate of 79.08%. Overall, LegalAgentBench effectively differentiates the capabilities of various LLMs, with those demonstrating stronger tool usage and logical reasoning achieving superior performance.

- **Comparing different methods.** By comparing different methods, we found that ReAct typically produces better results for multi-hop questions. However, this advantage comes with higher token consumption, suggesting that allowing more time for reasoning could improve performance. Additionally, we found that when LLMs are limited in capability, the P-E method doesn't always outperform P-S. LLMs like GLM-4, LLaMa3.1-8B, and GPT-3.5 show similar trends. This may be due to plan updates increasing the context length, which reduces the effectiveness of the attention mechanism. The performance gap between different reasoning methods for the same LLM can reach 65%, emphasizing that effective methods better utilize the LLM's potential. Additionally, when designing effective reasoning methods, it is also crucial to balance model capability, reasoning time, and token consumption.

- **Comparing different types of queries.** As shown in Table 3, for multi-hop questions, all

baselines showed decreased performance as the number of hops increased. The best performance achieved 93% success on the 1-hop question, but only 61% on the 5-hop question. This indicates that the questions in LegalAgentBench cover a wide range of difficulty levels. For the Writing task, we observed that ReAct performed poorly compared to other methods. We believe this may be due to the step-by-step update mechanism, which repeatedly attempts to solve individual steps when errors occur. In tasks like writing, which require parallel processing, this approach may overlook other potential paths that could lead to other answers. This also highlights that the diverse types of questions in LegalAgentBench effectively assess the potential of different methods.

4.3 Analysis

In this section, we analyze the unique challenges faced by LLM agents in LegalAgentBench and the potential improvements.

Lack of specialized legal knowledge. The terminology and concepts in the legal domain are highly specialized, and without sufficient legal knowledge, LLMs may struggle to generate accurate reasoning paths. For example, LMs often fail to distinguish between filing time and trial time or to interpret the specific meanings of different parts of a case number, all of which are crucial for arriving at the correct answer. Although we have provided a legal knowledge corpora, in practice, we find that LLMs often fail to recognize the necessity of consulting relevant legal knowledge. When handling legal issues, LLMs tend to rely on existing patterns and linguistic knowledge, overlooking the precision and details required in the legal domain. In the future, legal knowledge should be integrated into LLMs more systematically and in-depth to ensure accuracy and reliability in complex legal contexts. Moreover, LLMs should also intelligently recognize when to access external legal databases, enabling more flexible and accurate decision-making.

Insufficient understanding of legal articles and case law. The resolution of some legal issues relies on articles and case law, but LLMs may have significant limitations in this regard. During reasoning, LLMs often struggle to accurately interpret the scope and logic of legal articles. Even when retrievers successfully identify relevant articles and cases, LLMs may still encounter difficulties in understanding the judicial interpretations and practical applications of these referenced materials, especially in complex legal scenarios. In the future, to enhance the effectiveness of LLMs in the legal domain, it is essential to strengthen their ability to deeply understand and apply legal articles and cases.

Other Error Types In addition to the above challenges, LLM agents commonly encounter the following errors on LegalAgentBench: 1) Argument Errors: LLM agents fail to provide the correct argument when invoking tools. 2) Planning Errors: LLM agents generate incorrect planning paths or use inappropriate tools due to hallucinations or insufficient knowledge. 3) Exceeding Length Limitations: The encoded interaction history, observations, and tool usage plans exceed the length limit, preventing the LLM agents from resolving the task. 4) Getting Stuck in Loops: LLM agents may repeatedly attempt to solve the same problem, ultimately reaching the maximum iteration limit. Overall, LegalAgentBench provides a comprehensive and reliable evaluation platform, highlighting that LLM agents still have significant room for improvement in solving complex legal problems.

5 Conclusion

In this paper, we introduced LegalAgentBench, a benchmark designed to evaluate the performance of LLM agents in the legal domain. To achieve this, we collected real-world data to construct 17 databases and 37 tools. Additionally, we proposed a scalable task construction framework that comprises six steps: planning tree creation, path selection, entity verification, question rewriting, answer generation, and human verification. This framework is versatile and can be extended to incorporate new knowledge bases and tools, enabling the construction of tasks with varying types and difficulty levels. We conducted extensive experiments on LegalAgentBench, providing an in-depth analysis of the strengths, weaknesses, and potential improvement directions for existing models and methods. Looking ahead, we aim to expand LegalAgentBench to support additional languages and legal systems, fostering the development of legal LLMs worldwide.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Zehui Chen, Weihua Du, Wenwei Zhang, Kuikun Liu, Jiangning Liu, Miao Zheng, Jingming Zhuo, Songyang Zhang, Dahua Lin, Kai Chen, et al. 2024. T-eval: Evaluating the tool utilization capability of large language models step by step. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9510–9529.

Zhumin Chu, Qingyao Ai, Yiteng Tu, Haitao Li, and Yiqun Liu. 2024. Pre: A peer review based large language model evaluator. *arXiv preprint arXiv:2401.15641*.

Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *arXiv preprint arXiv:2306.16092*.

Ali Dorri, Salil S Kanhere, and Raja Jurdak. 2018. Multi-agent systems: A survey. *Ieee Access*, 6:28573–28593.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Neel Guha, Julian Nyarko, Daniel Ho, Christopher Ré, Adam Chilton, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel Rockmore, Diego Zambrano, et al. 2024. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. *Advances in Neural Information Processing Systems*, 36.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, et al. 2023. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*.

Haitao Li, Qingyao Ai, Jia Chen, Qian Dong, Yueyue Wu, Yiqun Liu, Chong Chen, and Qi Tian. 2023. Sailer: structure-aware pre-trained language model for legal case retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1035–1044.

Haitao Li, Qingyao Ai, Jia Chen, Qian Dong, Zhijing Wu, Yiqun Liu, Chong Chen, and Qi Tian. 2024a. Blade: Enhancing black-box large language models with small domain-specific models.

Haitao Li, Qingyao Ai, Xinyan Han, Jia Chen, Qian Dong, Yiqun Liu, Chong Chen, and Qi Tian. 2024b. Delta: Pre-train a discriminative encoder for legal case retrieval via structural word alignment. *arXiv preprint arXiv:2403.18435*.

Haitao Li, Junjie Chen, Qingyao Ai, Zhumin Chu, Yujia Zhou, Qian Dong, and Yiqun Liu. 2024c. Calibraeval: Calibrating prediction distribution to mitigate selection bias in llms-as-judges. *arXiv preprint arXiv:2410.15393*.

Haitao Li, You Chen, Qingyao Ai, Yueyue Wu, Ruizhe Zhang, and Yiqun Liu. 2024d. Lexeval: A comprehensive chinese legal benchmark for evaluating large language models. *arXiv preprint arXiv:2409.20288*.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. 2024e. Llms-as-judges: A comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*.

Haitao Li, Yunqiu Shao, Yueyue Wu, Qingyao Ai, Yixiao Ma, and Yiqun Liu. 2024f. Lecardv2: A large-scale chinese legal case retrieval dataset. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2251–2260.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023a. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023b. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn llm agents. *arXiv preprint arXiv:2401.13178*.

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*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2024. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Oguzhan Topsakal and Tahir Cetin Akinci. 2023. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, volume 1, pages 1050–1056.

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*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*.

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

Xiaohui Xie, Qian Dong, Bingning Wang, Feiyang Lv, Ting Yao, Weinan Gan, Zhijing Wu, Xiangsheng Li, Haitao Li, Yiqun Liu, and Jin Ma. 2023. T2ranking: A large-scale chinese benchmark for passage ranking. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 2681–2690, New York, NY, USA. Association for Computing Machinery.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Junjie Ye, Guanyu Li, Songyang Gao, Caishuang Huang, Yilong Wu, Sixian Li, Xiaoran Fan, Shihan Dou, Qi Zhang, Tao Gui, et al. 2024. Tooleyes: Fine-grained evaluation for tool learning capabilities of large language models in real-world scenarios. *arXiv preprint arXiv:2401.00741*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2024. A survey of large language models.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools. *Advances in Neural Information Processing Systems*, 36:50117–50143.

A Discussion

In this section, we discuss limitations, potential impacts, license, and legal and ethical considerations.

A.1 Limitation

We acknowledge several limitations in this study and aim to address them in future work. First, the current dataset is primarily constructed in Chinese, which limits its applicability in broader multilingual contexts. We plan to release an updated version in future iterations that will support English. Second, the task design primarily covers statutory law system, and its performance in case law systems requires further exploration. It may not fully capture the diversity of legal knowledge and systems across different countries. In future work, we will enrich the task types, broaden the scope, and incorporate legal scenarios from more countries and regions, thereby enhancing the applicability and comprehensiveness of our research.

A.2 Broader Impact

LegalAgentBench aims to promote the application of LLM agents in the legal field and help legal professionals better understand and evaluate the performance of LLM agents through a standardized evaluation framework. By providing reliable evaluation methods and construction processes, LegalAgentBench not only facilitates the application of LLM agents in legal tasks but also offers valuable insights for building benchmarks in other vertical domains. The widespread use of LLM agents in the legal industry may influence the way legal professionals work, change how they use technological tools, and drive adjustments in legal education and practice. We are particularly focused on the far-reaching impact of LLM agents on the legal industry, ensuring that their application always adheres to principles of social justice and the rule of law. To guarantee fairness and transparency, the datasets and evaluation methods of LegalAgentBench will undergo rigorous ethical reviews and be subject to broad stakeholder involvement to ensure their impartiality.

It is important to note that LegalAgentBench does not advocate for the complete replacement of legal professionals by LLM agents but seeks to reduce the burden on legal personnel and enhance efficiency through supportive tools. The uniqueness and complexity of legal judgment require rich expertise and human insight, qualities that LLM agents cannot fully replace. Our goal is to provide legal professionals with a standardized evaluation tool to help them make more informed decisions in practical applications, understanding when, where, and how to effectively use LLM agents. However, the evaluation results of LegalAgentBench should be used as a reference only, as applications in real-world legal scenarios still require further in-depth evaluation to ensure the legality and reasonableness of decisions. We believe that LegalAgentBench will contribute to the development of a more just and efficient legal system and promote the responsible application of AI technology in the legal field.

A.3 License

In this section, we clarify the copyright and licensing status of LegalAgentBench to ensure that users can utilize this dataset in a legal and compliant manner.

In LegalAgentBench, all tabular databases is sourced from authentic, publicly available resources. We have obtained explicit copyright permissions to incorporate these databases into the benchmark. Additionally, the retrieval corpora are derived from publicly accessible legal materials that comply with relevant legal and ethical standards. These resources are made available in accordance with applicable norms for open access to legal information, ensuring that their inclusion in the benchmark does not raise any legal or ethical concerns. While the copyright for these resources remains with the respective government agencies, they have been publicly released and authorized for public use. Users are expected to adhere to the relevant laws, regulations, and provisions established by the respective government agencies when utilizing this data

The entire dataset is released under the MIT License. If you believe that LegalAgentBench contains content that infringes on your copyrighted work, please feel free to contact us at any time to request its removal.

A.4 Legal and Ethical Considerations

The construction and release of LegalAgentBench adhere to strict legal and ethical standards to ensure its responsible use in research and development. All data in LegalAgentBench has undergone rigorous privacy screening and anonymization processes. Any personal or sensitive information has been removed to comply with applicable data protection laws and ethical research guidelines, ensur-

ing the dataset can be used without compromising individual privacy or security. To prevent potential harm, the datasets in LegalAgentBench have been carefully curated and filtered to exclude discriminatory, explicit, violent, or offensive content. An ethical review by legal experts further ensures that the benchmark minimizes risks related to security, safety, discrimination, surveillance, deception, harassment, human rights, bias, and fairness. By addressing these considerations, LegalAgentBench aims to provide a legally compliant and ethically sound foundation for advancing AI capabilities in the legal domain, while upholding the principles of fairness, transparency, and social responsibility.

B Corpus Details

Table 4 provides detailed information about the corpora in LegalAgentBench, it lists the keys for 14 different tabular databases along with the number of keys per table, ranging from 3 to 28. This variation reflects the diversity of scenarios in real-world applications. Additionally, basic examples are provided for the three retrieval corpus to clarify their content.

C Tool Details

Tables 5 and 6 provide detailed information about the tools included in LegalAgentBench. Table 5 outlines the input and output for each tool, while Table 6 describes the basic functionality of each tool.

For each tool, beyond the basic input information, we also provide optional parameters to enhance flexibility in usage. For instance, the Database Tools include the parameter *Columns*, which specifies the keys of the structured data to be returned. Similarly, the text retriever offers a *Number* parameter that allows users to control the number of documents retrieved. To enhance understanding, Table 7 provides a concrete example of a tool call.

D Task Details

Tables 8 and 9 provide concrete examples of the tasks included in LegalAgentBench. Table 8 illustrates a **multi-hop task** along with a human-provided solution path, showcasing the reasoning steps involved. Table 9 presents an example of the **writing task** from the dataset, where the task requires not only querying and retrieving various pieces of information but also processing and organizing the information into a formatted legal defense document. This example highlights the challenges faced by LLM agents when applied to the legal domain, demonstrating the need for advanced reasoning, information synthesis, and adherence to domain-specific requirements.

E More Implementation details

E.1 Prompt

In this section, we present the key prompts used in LegalAgentBench. More details can be found in our code. Table 10 shows the prompt template for question rewriting during the task construction process. We used GPT-4 to rewrite the original questions and manually reviewed the rewritten questions for accuracy. Tables 11, 12 and 13 present the prompts used in different methods.

E.2 Metrics

In this section, we provide a detailed explanation of the calculation for success rate and process rate. Assume there is a dataset $\mathcal{D}$ consisting of N data points, where each data point includes a keyword set $\mathcal{K}_i$ and a model output $\mathcal{O}_i$. The rate s_i for the i-th data point is calculated as:

$$s_i = \frac{|\mathcal{M}_i|}{|\mathcal{K}_i|}$$

where $\mathcal{M}_i = \{k \in \mathcal{K}_i | k \text{ appears in } \mathcal{O}_i\}$. The notation $|\cdot|$ represents the number of elements in a set. When the keyword set $\mathcal{K}_i$ = key_answer, s_i represents the success rate. When $\mathcal{K}_i$ = key_answer $\cup$ key_middle, s_i represents the progress rate. We report the average of all tasks in the experimental results.

F More Evaluation Result

Tables 14 and 15 report the progress rate and BertScore of different LLMs and methods on LegalAgentBench. Table 14 highlights the progress rate as a more fine-grained evaluation metric. We observed that GLM-4-Plus surpasses GPT-4o in progress rate across all tasks. This suggests that, although GLM-4-Plus may not perform as well as GPT-4o in terms of final outcomes, it achieves a higher degree of intermediate completion in many tasks. This provides a more comprehensive and fine-grained perspective for analyzing model performance. In Table 15, BERTScore measures the semantic similarity between the model's output

and the answer. However, the differences between baselines become smaller, making it challenging to effectively distinguish the performance of different baselines. Given the high accuracy requirements in the legal domain, we recommend using success rate and progress rate to evaluate model performance.

G Guidelines for Expert-Annotation

To ensure the quality and reliability of the dataset during its construction, we conducted human verification on LegalAgentBench. To guide the validation process and maintain consistency, we provided the following annotation guidelines:

Validation of Answer Accuracy: Annotators must independently call the relevant tools to generate correct answers, ensuring that the dataset reflects accurate and reliable outputs. Incorrect answers should be carefully reviewed and corrected to maintain the integrity of the dataset. This process involves cross-verifying outputs with authoritative sources, re-running tool-based evaluations where necessary, and documenting any adjustments made to ensure transparency and reproducibility.

Validation of Query Rewriting: Annotators must ensure that rewritten queries preserve the meaning and intent of the original query. Specifically, they must verify that the answers obtained from the original and rewritten queries are identical, thereby ensuring semantic equivalence and logical consistency. In addition, any rewritten query that does not align with practical, everyday usage should be modified to ensure naturalness and usability.

Verification of Relevant Legal Provisions: Certain tasks, such as Writing tasks, require citing legal knowledge, articles or cases. Annotators must verify that all references are accurate, up-to-date, and relevant to the task at hand. This ensures the legal soundness of the dataset and enhances its applicability in real-world scenarios.

Handling Doubts and Uncertainties: If annotators encounter doubts or uncertainties during validation, they are required to consult official documents, legal texts, or terminological glossaries associated with the relevant classification system. Collaboration with legal experts is strongly encouraged to resolve ambiguities and clarify issues, ensuring that the dataset remains precise and unambiguous.

Review and Quality Control: A robust review mechanism is established to maintain high-quality annotations. Senior annotators regularly cross-check and review the annotations, correcting simple errors and refining complex cases. Each annotation undergoes multiple rounds of manual verification to ensure accuracy and consistency. In cases where disagreements arise among annotators, collaborative discussions are held to reach consensus, with the final decision documented to ensure transparency.

Feedback Mechanism: To promote continuous improvement, a feedback mechanism is in place, allowing annotators to provide insights and suggestions regarding the annotation guidelines. This iterative refinement ensures that the guidelines remain effective, up-to-date, and aligned with the evolving requirements of the dataset.

Ethical Considerations: Ensure that all annotations are conducted with integrity and impartiality, maintaining high standards of accuracy and fairness. Take proactive measures to avoid any biases or conflicts of interest that could compromise the quality or objectivity of the annotations.

By adhering to these guidelines, we aim to produce a high-quality dataset that is not only accurate and reliable but also capable of supporting meaningful advancements in legal AI applications.

ID	Corpus	Key	# Key
1	CompanyInfo	Company Name, Company Abbreviation, English Name, Associated Security, Company Code, Former Abbreviation, Market, Industry, Date of Establishment, Listing Date, Legal Representative, General Manager, Board Secretary, Postal Code, Registered Address, Office Address, Telephone Number, Fax, Official Website, Email, Included Indices, Main Business, Business Scope, Company Profile, Par Value per Share (CNY), Initial Public Offering (IPO) Price (CNY), Net Proceeds from IPO (CNY), Lead Underwriter for IPO	**28**
2	CompanyRegister	Company Name, Registration Status, Unified Social Credit Code, Legal Representative, Registered Capital, Date of Establishment, Company Address, Contact Number, Contact Email, Registration Number, Organization Code, Number of Insured Persons, Primary Industry Category, Secondary Industry Category, Tertiary Industry Category, Former Name, Company Profile, Business Scope	**18**
3	SubCompanyInfo	Full Name of the Related Listed Company, Relationship to the Listed Company, Listed Company's Shareholding Percentage, Listed Company's Investment Amount, Company Name	**5**
4	LegalDoc	Related Company, Title, Case Number, Document Type, Plaintiff, Defendant, Plaintiff's Law Firm, Defendant's Law Firm, Cause of Action, Amount Involved (CNY), Judgment Outcome, Date, File Name	**13**
5	LegalAbstract	File Name, Case Number, Text Summary	**3**
6	CourtInfo	Court Name, Court Head, Date of Establishment, Court Address, Court Province, Court City, Court County, Court Contact Number, Court Official Website	**9**
7	CourtCode	Court Name, Administrative Level, Court Level, Court Code, Zoning Code, Rank	**6**
8	LawfirmInfo	Law Firm Name, Law Firm Unique Code, Law Firm Head, Law Firm Registered Capital, Date of Establishment, Law Firm Address, Law Firm Province, Law Firm City, Law Firm County, Contact Phone Number, Contact Email, Law Firm Profile, Law Firm Registration Authority	**13**
9	LawfirmLog	Law Firm Name, Business Volume Ranking, Served Listed Companies, Listed Companies with Violations During Reporting Period, Listed Companies Under Investigation During Reporting Period	**5**
10	AddrInfo	Address, Province, City, County	**4**
11	RestrictionCase	Restricted High-Consumption Enterprise Name, Case Number, Legal Representative, Applicant, Amount Involved (CNY), Executing Court, Filing Date, Restriction Publication Date	**4**
12	FinalizedCase	Finalized Company Name, Case Number, Person Subject To Execution, Suspected Applicant for Enforcement, Unfulfilled Amount (CNY), Executor Target (CNY), Executing Court, Filing Date, Finalized Date	**8**
13	DishonestyCase	Dishonest Executed Company Name, Case Number, Dishonest Executed Person, Suspected Applicant for Enforcement, Amount Involved (CNY), Executing Court, Filing Date, Publication Date	**9**
14	AdministrativeCase	Administrative Penalty Company Name, Case Number, Facts, Penalty Outcome, Penalty Amount (CNY), Penalizing Authority, Penalty Date	**8**
15	LegalKonwledge	**EXAMPLE**: Tax Law, Tax Collection and Administration Law, Legal Liability (1) If a taxpayer fails to pay taxes within the prescribed period or if a withholding agent...	**-**
16	LegalArticle	**EXAMPLE**: Article 1082 of the Civil Code of the People's Republic of China During the woman's pregnancy, within one year after childbirth, or within six months after termination of pregnancy, the man shall not file for divorce...	**-**
17	LegalCases	**EXAMPLE**: Application for Recognition of a Civil Judgment by Kaohsiung County Court in Taiwan: Kang vs. Huang. (1) Basic Facts: On June 1, 2004, mainland resident Kang...	**-**

Table 4: More details about the corpus of LegalAgentBench.

ID	Tool	Input	Return
1	get_company_info	Company name or abbr. or code	CompanyInfo
2	get_company_register	Company name	CompanyRegister
3	get_company_register_name	Unified social credit code	Company name
4	get_sub_company_info	Company name	SubCompanyInfo
5	get_sub_company_info_list	Company name	List[SubCompanyInfo]
6	get_legal_document	Case number	LegalDoc
7	get_legal_abstract	Case number	LegalAbstract
8	get_legal_document_company_list	Company name	List[LegalDoc]
9	get_legal_document_lawfirm_list	Law firm name	List[LegalDoc]
10	get_court_info	Court name	CourtInfo
11	get_court_info_list	Province, city, and county	List[CourtInfo]
12	get_court_code	Court name or code	CourtCode
13	get_lawfirm_info	Law firm name	LawfirmInfo
14	get_lawfirm_info_list	Province, city, and county	List[LawfirmInfo]
15	get_lawfirm_log	Law firm name	LawfirmLog
16	get_address_info	Specific address	AddrInfo
17	get_restriction_case	Case number	RestrictionCase
18	get_restriction_case_company_list	Company name	List[RestrictionCase]
19	get_restriction_case_court_list	Court name	List[RestrictionCase]
20	get_finalized_case	Case number	FinalizedCase
21	get_finalized_case_company_list	Company name	List[FinalizedCase]
22	get_finalized_case_court_list	Court name	List[FinalizedCase]
23	get_dishonesty_case	Case number	DishonestyCase
24	get_dishonesty_case_company_list	Company name	List[DishonestyCase]
25	get_dishonesty_case_court_list	Court name	List[DishonestyCase]
26	get_administrative_case	Case number	AdministrativeCase
27	get_administrative_case_company_list	Company name	List[AdministrativeCase]
28	get_administrative_case_court_list	Court name	List[AdministrativeCase]
29	legal_knowledge_retriever	Query text	Relevant knowledge
30	legal_article_retriever	Query text	Relevant articles
31	legal_case_retriever	Query text	Relevant cases
32	get_sum	List[num]	Summation results
33	get_subtraction	Minuend, subtrahend	Subtraction results
34	get_multiplication	List[num]	Multiplication results
35	get_division	Dividend, divisor	Division results
36	get_rank	List[num]	Sorting results
37	finish	-	Final results

Table 5: The input and output of the tools.

ID	Tool	Description
1	get_company_info	Query the corresponding listed company information in the [CompanyInfo] based on [Company Name, Company Abbreviation, or Company Code].
2	get_company_register	Query the corresponding company registration information in the [CompanyRegister] based on [Company Name].
3	get_company_register_name	Query the corresponding company name in the [CompanyRegister] based on [Unified Social Credit Code].
4	get_sub_company_info	Query the corresponding parent company and investment information in the [SubCompanyInfo] based on [Company Name].
5	get_sub_company_info_list	Query all subsidiary companies invested by the parent company in the [SubCompanyInfo] based on [Company Name].
6	get_legal_document	Query the corresponding judgment document information in the [LegalDoc] based on [Case Number].
7	get_legal_abstract	Query the text summary of the case in the [LegalAbstract] based on [Case Number].
8	get_legal_document_company_list	Query the corresponding judgment document information in the [LegalDoc] based on [Company Name].
9	get_legal_document_lawfirm_list	Query the corresponding judgment document information in the [LegalDoc] based on [Law Firm Name].
10	get_court_info	Query the relevant court information in the [CourtInfo] based on [Court Name].
11	get_court_info_list	Query the relevant court information in the [CourtInfo] based on [Province, City, County].
12	get_court_code	Query the relevant court information in the [CourtCode] based on [Court Name or Court Code].
13	get_lawfirm_info	Query the relevant law firm information in the [LawfirmInfo] based on [Law Firm Name].
14	get_lawfirm_info_list	Query the relevant law firm information in the [LawfirmInfo] based on [Province, City, County].
15	get_lawfirm_log	Query the service records of the law firm in the [LawfirmLog] based on [Law Firm Name].
16	get_address_info	Query the province, city, and county of the address in the [AddrInfo] based on [Specific Address].
17	get_restriction_case	Query the relevant high-consumption restriction case information in the [RestrictionCase] based on [Case Number].
18	get_restriction_case_company_list	Query the relevant high-consumption restriction case information in the [RestrictionCase] based on [Company Name].
19	get_restriction_case_court_list	Query the relevant high-consumption restriction case information in the [RestrictionCase] based on [Executing Court Name].
20	get_finalized_case	Query the relevant finalized case information in the [FinalizedCase] based on [Case Number].
21	get_finalized_case_company_list	Query the relevant finalized case information in the [FinalizedCase] based on [Finalized Company Name].
22	get_finalized_case_court_list	Query the relevant finalized case information in the [FinalizedCase] based on [Executing Court Name].
23	get_dishonesty_case	Query the relevant dishonesty enforcement case information in the [DishonestyCase] based on [Case Number].
24	get_dishonesty_case_company_list	Query the relevant dishonesty enforcement case information in the [DishonestyCase] based on [Company Name].
25	get_dishonesty_case_court_list	Query the relevant dishonesty enforcement case information in the [DishonestyCase] based on [Executing Court Name].
26	get_administrative_case	Query the relevant administrative penalty case information in the [AdministrativeCase] based on [Case Number].
27	get_administrative_case_company_list	Query the relevant administrative penalty case information in the [AdministrativeCase] based on [Company Name].
28	get_administrative_case_court_list	Query the relevant administrative penalty case information in the [AdministrativeCase] based on [Penalizing Authority].
29	legal_knowledge_retriever	Retrieve relevant legal knowledge based on [Query text].
30	legal_article_retriever	Retrieve relevant legal articles based on [Query text].
31	legal_case_retriever	Retrieve relevant legal cases based on [Query text].
32	get_sum	Perform [Summation] based on [List[num]].
33	get_subtraction	Perform [Subtraction] based on [Minuend, Subtrahend].
34	get_multiplication	Perform [Multiplication] based on [List[num]].
35	get_division	Perform [Division] based on [Dividend, Divisor].
36	get_rank	Perform [Sorting] based on [List[num]].
37	finish	Summarize existing information to generate an answer.

Table 6: The descriptions of the tool functions.

Example: Use get_restriction_case_company_list to query the amounts involved in all cases participated in by **Jiangsu Yanning New Material Technology Development Co., Ltd**.

Call Tool:

get_restriction_case_company_list(
    identifier="Jiangsu Yanning New Material Technology Development Co., Ltd.",
    columns=["Amount Involved (CNY)"])

Return:

[{'Amount Involved (CNY)': 686550}, {'Amount Involved (CNY)': 385353}, {'Amount Involved (CNY)': 17875}, {'Amount Involved (CNY)': 2456446}]

Table 7: A concrete example of a tool call. There are four cases related to the specified company. Since the *columns* parameter specifies the Amount Involved (CNY) attribute, each dictionary in the returned list contains only the Amount Involved (CNY) key. If columns=[], all keys in the corresponding table will be returned.

Task: What is the total amount of restricted high consumption for 91320115773957541H?

Step:

Firstly, since the provided Unified Social Credit Code "91320115773957541H" cannot be directly used to query the total amount of restricted high consumption, we use the **get_company_register_name** tool to obtain the company name associated with this code.

Result of Action:

The query reveals that the company name is Jiangsu Yanning New Material Technology Development Co., Ltd.

Secondly, we use the **get_restriction_case_company_list** tool to query all restricted high consumption cases involving this company, based on the company name retrieved in the previous step.

Result of Action:

The query returns the amounts involved in all cases as follows: [{'Amount Involved (CNY)': 686550}, {'Amount Involved (CNY)': 385353}, {'Amount Involved (CNY)': 17875}, {'Amount Involved (CNY)': 2456446}].

Thirdly, we use the **get_sum** tool to calculate the total of the amounts obtained in the previous step.

Result of Action:

The calculation yields a total of 3546224 CNY.

Finally, we summarize the information and answer the original question.

Answer: The total amount of restricted high consumption for 91320115773957541H is 3546224 CNY.

Table 8: An example of the **multi-hop task.**

Task: PersonA has filed a lawsuit against CompanyX. The specific content of the complaint is as follows: [Complaint Content]. CompanyX has engaged Law Firm A for legal representation. As the head of Law Firm A, please draft a defense statement in response to the complaint based on the specified defense statement format: [Defense Statement Format].

Step:
Firstly, we need to retrieve the address, legal representative, Unified Social Credit Code, and contact number of CompanyX.
Call Tool: get_company_register

Secondly, we need to retrieve the head and contact number of Law Firm A.
Call Tool: get_court_info

Thirdly, we need to retrieve relevant legal knowledge, legal article, and legal case
Call Tools: legal_knowledge_retriever, legal_article_retriever, legal_case_retriever

Finally, we generate the answer following the specified defense statement format.

Answer:
Defendant: [Company Name], [Address], [Legal Representative], [Unified Social Credit Code], [Contact Number]
Authorized Litigation Representative: [Law Firm Name], [Law Firm Head], [Law Firm Contact Number]
Plaintiff: [Name]
[Specific Content of the Defense Statement]

Table 9: An example of the **writing task.**

The prompt used in question rewriting.

You are an advanced question rewriter. Your task is to rewrite the given questions to make them more relevant to real-life scenarios and sound more natural. Please adhere to the following requirements:
1. **Preserve the Core Inquiry**: Do not change the core content or the focus of the original question. Ensure the user's intent is not misunderstood.
2. **Introduce Misleading Context**: Add potentially misleading context to obscure the true purpose of the query.
3. **Maintain Logical Coherence**: The rewritten question should align with daily usage scenarios, have smooth language, and be logically coherent.
4. **Relate to Legal Needs**: Whenever possible, ensure the question remains tied to legal requirements.
5. **Ensure Clarity**: Retain the critical points of the inquiry so that the answers to the original question and the rewritten question remain consistent.

Table 10: The prompt template for question rewriting during the task construction process.

Stage	The Prompt used in the Plan-and-Solve method.
Plan	Solve a Question-Answering Task. Please understand the question and develop a step-by-step plan to solve it. Start your output with the title "Plan:" and follow it with a list of steps. Each step should begin with "Step n:", where n is the current step number (1, 2, 3, ...). The plan should include several distinct steps, and completing these steps sequentially will yield the correct answer. Ensure the plan is sufficiently detailed to complete the task accurately, without skipping any steps or adding unnecessary ones. The final step should always be: "Based on the above steps, please answer the original question." At the end of the plan, include the line "End of Plan." You can use the following tools: {tools} Relevant data tables and their fields (any field appearing in the table can be used as a value in the columns parameter): {table_used_prompt} Note: Your task is to develop the plan and output it as requested. Do not execute the plan! Here are some examples: {examples} (Examples End) Question: {question}
Solve	Given a single-step plan, please output the specific action you intend to execute based on the plan. "Action" is specified using a JSON block, which includes an action key (tool name) and an action_input key (tool input). Valid values for the action key include: "Final Answer" or {tool_names}. You may use the following tools: {tools} Relevant data tables and their available fields (any field in the table can be used as a value in the columns parameter): {table_used_prompt} Each "Action" can only call one tool at a time. If multiple tools need to be called, split them into separate steps. The output of an "Action" must strictly follow the JSON format below and be parsable by Python's json.loads function: “‘json {{ "action": TOOL_NAME, "action_input": INPUT }} ”’ Here are some examples: {examples} (Examples End) Plan you need to execute: {plan} Steps you have already completed: {scratchpad} Action:

Table 11: The prompt used in the Plan-and-Solve method.

Stage	The prompt used in the Plan-and-Execute method.
Replan	Solve a Question-Answering task. Please understand the question and develop a step-by-step plan to solve it. Start your output with the title "Plan:" and follow it with a list of steps. Each step should begin with "Step n:", where n is the current step number (1, 2, 3, ...). The plan should include several distinct steps, and completing these steps sequentially will yield the correct answer. Ensure the plan is sufficiently detailed to complete the task accurately, without skipping any steps or adding unnecessary ones. The final step should always be: "Based on the above steps, please answer the original question." At the end of the plan, include the line "End of Plan." You may use the following tools: {tools} Relevant data tables and their fields (any field in the table can be used as a value in the columns parameter): {table_used_prompt} Note: You are only responsible for creating the plan according to the requirements. Do not execute the plan! Examples: {examples} (Examples End) Question: {question} Your Original Plan: {plan} Steps You Have Already Completed: {scratchpad} Please update the plan based on the situation. If additional steps are needed, list the remaining steps strictly following the format requirements. Retain the already completed steps as they are and do not repeat them. Each step should begin with "Step n:", where n is the current step number (1, 2, 3, ...). The final step should always be: "Based on the above steps, please answer the original question." At the end of the plan, include the line "End of Plan."

Table 12: The prompt used in the Plan-and-Execute method.

The prompt used in the ReAct method.

Solve a Question-Answering Task. The process involves alternating steps of "Thought", "Action", and "Observation".
- "Thought" is used to reason about the next step based on the current situation. Note that you only need to consider the immediate next step.
- "Action" refers to specifying the tool to use through a JSON block containing an 'action' key (tool name) and an 'action_input' key (tool input).
- Valid values for the 'action' key include: '"Final Answer" or {tool_names}.
- You may use the following tools:
{tools}
- Relevant data tables and their available fields (any field in the table can be used as a value in the 'columns' parameter) include:
{table_used_prompt}
- Each "Action" can call only one tool. If multiple tools need to be called, break them into separate steps.
- The output of an "Action" strictly follow the JSON format below and be parsable by Python's 'json.loads' function:
"'json
{{
"action": TOOL_NAME,
"action_input": INPUT
}}
"'
Examples:
{examples}
(Examples End)
Important Notes:
- When outputting "Action," the result must strictly follow the JSON format specified above.
- When outputting "Thought," only reason about the immediate next step, and avoid thinking about multiple steps ahead.
Question: {question}
{scratchpad}

Table 13: The prompt used in the ReAct method.

Model	Method	1-hop	2-hop	3-hop	4-hop	5-hop	Writing	ALL	Tokens
GLM-4	P-S	0.7519	0.5094	0.3502	0.3238	0.2334	0.4989	0.4984	5,100,468
	P-E	0.7477	0.5225	0.4140	0.3080	0.2631	0.4801	0.5121	9,849,924
	ReAct	0.8967	0.7092	0.5103	0.4191	0.2820	0.5179	0.6395	11,920,863
GLM-4-Plus	P-S	0.8231	0.6577	0.5336	0.4791	0.3366	0.6367	0.6304	5,657,827
	P-E	0.8673	0.6531	0.5447	0.4762	0.3156	0.6401	0.6416	9,422,692
	ReAct	0.9323	0.7835	0.6233	**0.5788**	0.4026	0.6273	**0.7280**	11,739,861
LLaMa3.1-8B	P-S	0.3475	0.2498	0.0999	0.0734	0.0859	0.2631	0.2123	9,279,701
	P-E	0.3719	0.2248	0.1247	0.0934	0.1097	0.3334	0.2260	13,649,741
	ReAct	0.6177	0.1573	0.0649	0.0468	0.0581	0.1074	0.2369	50,661,127
Qwen-max	P-S	0.8485	0.6442	0.4579	0.3994	0.2610	0.4557	0.5907	4,800,345
	P-E	0.8979	0.7004	0.5233	0.4141	0.2836	0.5011	0.6384	9,884,307
	ReAct	0.9229	**0.7954**	0.5832	0.5185	0.4908	0.5659	0.7144	11,473,873
Claude-sonnet	P-S	0.8304	0.6987	0.5975	0.4794	0.3195	0.6575	0.6563	7,100,962
	P-E	0.8262	0.7340	0.6458	0.5188	0.4418	0.6776	0.6890	13,566,119
	ReAct	0.8867	0.7356	0.4850	0.4533	0.3379	0.5504	0.6493	32,878,858
GPT-3.5	P-S	0.4994	0.2856	0.1296	0.0905	0.0412	0.0863	0.2558	5,007,391
	P-E	0.5067	0.2731	0.1254	0.0886	0.0496	0.0973	0.2546	9,597,807
	ReAct	0.6344	0.3117	0.1162	0.1018	0.1102	0.0813	0.3019	11,357,664
GPT-4o-mini	P-S	0.7010	0.5371	0.4269	0.3193	0.1637	0.5224	0.5039	5,482,556
	P-E	0.7298	0.5548	0.4525	0.3124	0.2032	0.5547	0.5252	10,861,492
	ReAct	**0.9354**	0.6746	0.4834	0.3652	0.3714	0.5014	0.6329	13,332,418
GPT-4o	P-S	0.6773	0.6137	0.4544	0.3813	0.2878	0.6336	0.5474	4,153,333
	P-E	0.7669	0.6204	0.4772	0.3955	0.2424	**0.6748**	0.5793	7,261,238
	ReAct	0.9344	0.7937	**0.6397**	0.4833	**0.4880**	0.5123	0.7199	11,206,957

Table 14: The process rate of different baselines on LegalAgentBench. P-S represents the Plan-and-Solve method, and P-E represents the Plan-and-Execute method. The best results are highlighted in bold.

Model	Method	1-hop	2-hop	3-hop	4-hop	5-hop	Writing	ALL	Tokens
GLM-4	P-S	0.8511	0.7585	0.6996	0.7289	0.7459	0.7754	0.7678	5,100,468
	P-E	0.8392	0.7389	0.7131	0.7288	0.7270	0.7725	0.7606	9,849,924
	ReAct	0.9086	0.8088	0.7913	0.7691	0.7597	0.7705	0.8208	11,920,863
GLM-4-Plus	P-S	0.8998	0.8036	0.7505	0.7726	0.7805	0.7790	0.8113	5,657,827
	P-E	0.9000	0.8168	0.7453	0.7551	0.7530	0.7791	0.8097	9,422,692
	ReAct	0.9284	**0.8718**	**0.8254**	**0.8267**	**0.8277**	0.7873	**0.8630**	11,739,861
LLaMa3.1-8B	P-S	0.7427	0.6674	0.6356	0.6241	0.6313	0.6640	0.6727	9,279,701
	P-E	0.7278	0.6542	0.6421	0.6327	0.6413	0.7336	0.6730	13,649,741
	ReAct	0.8080	0.6944	0.6583	0.6390	0.6672	0.5931	0.7015	50,661,127
Qwen-max	P-S	0.8581	0.7518	0.7240	0.7296	0.7338	0.7570	0.7708	4,800,345
	P-E	0.8521	0.7664	0.7259	0.7035	0.7300	0.7602	0.7699	9,884,307
	ReAct	0.9033	0.8315	0.8196	0.7676	0.7886	0.7835	0.8337	11,473,873
Claude-sonnet	P-S	0.8566	0.7841	0.7452	0.7246	0.7416	0.7928	0.7855	7,100,962
	P-E	0.8538	0.7937	0.7441	0.7314	0.7560	0.7913	0.7888	13,566,119
	ReAct	0.8722	0.8038	0.7770	0.7510	0.7747	0.7679	0.8053	32,878,858
GPT-3.5	P-S	0.8329	0.7344	0.6961	0.6856	0.6317	0.5332	0.7262	5,007,391
	P-E	0.8325	0.7302	0.7008	0.6544	0.6201	0.5412	0.7216	9,597,807
	ReAct	0.8695	0.7618	0.7182	0.6933	0.6459	0.5125	0.7483	11,357,664
GPT-4o-mini	P-S	0.8945	0.7811	0.7364	0.7498	0.7427	0.7766	0.7954	5,482,556
	P-E	0.8958	0.7884	0.7376	0.7493	0.7372	0.7785	0.7976	10,861,492
	ReAct	**0.9303**	0.8307	0.7917	0.7911	0.7900	0.7679	0.8373	13,332,418
GPT-4o	P-S	0.8966	0.7874	0.7432	0.7387	0.7450	0.7864	0.7983	4,153,333
	P-E	0.8955	0.7822	0.7352	0.7407	0.7489	**0.7964**	0.7962	7,261,238
	ReAct	0.9154	0.8346	0.8199	0.7848	0.8047	0.7792	0.8409	11,206,957

Table 15: The BERT-Score of different baselines on LegalAgentBench. P-S represents the Plan-and-Solve method, and P-E represents the Plan-and-Execute method. The best results are highlighted in bold.