Legal-R1: Agentic Retrieval for Evidence-Based Legal Reasoning

Anonymous ACL submission

Abstract

Legal question answering requires answers that are not only plausible, but also grounded in legally valid and citable evidence. Standard retrieval-augmented generation often relies on one-shot semantic matching, which is insufficient when legal evidence must be selected according to authority, applicability, and diversity across heterogeneous sources. We introduce **Legal-R1**, an agentic retrieval framework for evidence-based legal reasoning. Legal-R1 decomposes legal reasoning into a multi-agent workflow, where a Lawyer agent learns when to retrieve, which legal source to query, what evidence to seek, and when the collected evidence is sufficient. A Legal Assistant agent retrieves from normative legal sources and precedent cases with authority-aware and diversity-aware selection, while a Judge agent generates the final answer from the complete evidence trajectory. We post-train only the Lawyer agent with masked supervised fine-tuning and trajectory-aware reinforcement learning. Experiments on multiple Chinese legal benchmarks show that Legal-R1 improves over general-purpose, legal-specific, and retrieval-augmented baselines.[1]

1 Introduction

Legal question answering requires more than fluent generation. A trustworthy legal answer must be grounded in legal evidence that is valid, applicable, and citable. This evidence is often distributed across heterogeneous sources, including laws, judicial interpretations, and precedent cases. It must also satisfy domain-specific constraints such as authority hierarchy, temporal validity, jurisdictional fit, and applicability to the target facts. These requirements make legal question answering a problem of evidence construction rather than plain information lookup.

[1]We will release the code, model, and data at https://anonymous.4open.science/r/Legal-R1-EC03.

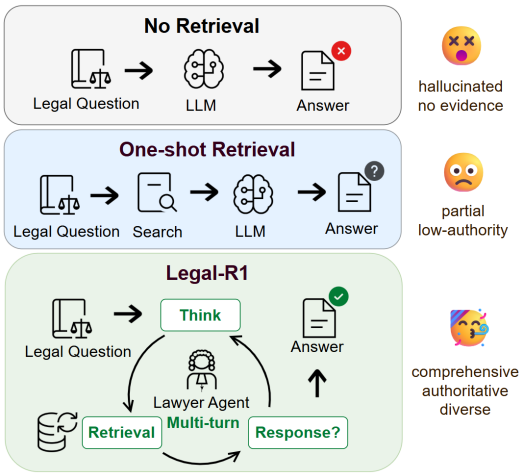

Figure 1: Motivation of Legal-R1. Legal-R1 replaces one-turn semantic matching with multi-turn evidence seeking over norms and cases.

Existing large language models still struggle with this setting. Parametric legal knowledge can be incomplete, outdated, or unsupported by explicit citations. Retrieval-augmented generation improves grounding by retrieving external documents, but standard pipelines usually perform one-shot semantic matching. This is not sufficient for legal reasoning. A semantically similar case may come from a low-authority court, fail to match the relevant legal issue, or duplicate the same sentencing outcome as other retrieved cases. Legal retrieval therefore cannot be reduced to finding similar text. It must identify legally valid evidence that can justify the answer. Figure 1 illustrates this distinction. General agentic retrieval asks what information is needed to answer a question. Legal agentic retrieval asks what legal norms and precedents are needed to support a verifiable answer. The agent must decide whether retrieval is needed, which legal source to query, what evidence to retrieve, and whether the collected evidence is sufficient.

We propose **Legal-R1**, an agentic retrieval

framework for evidence-based legal reasoning. Legal-R1 decomposes legal reasoning into three roles. The *Lawyer agent* plans multi-turn evidence seeking. It identifies missing legal evidence, chooses a retrieval action, formulates source-specific queries, and decides when to stop. The *Legal Assistant agent* executes retrieval over heterogeneous legal knowledge bases. It searches legal norms, including laws and judicial interpretations, and retrieves precedent cases with authority-aware and diversity-aware selection. The *Judge agent* generates the final answer from the legal question and the complete retrieval trajectory. This workflow makes evidence collection explicit and encourages answers to be grounded in comprehensive, authoritative, and diverse legal evidence.

We train only the Lawyer agent. This design focuses learning on the core difficulty of legal agentic retrieval, namely evidence-seeking decisions. It also decouples retrieval planning from answer generation, so the trained Lawyer agent can be paired with different Judge agents in downstream applications. To train the Lawyer agent, we synthesize multi-turn retrieval trajectories from a structured legal QA dataset. We first use masked supervised fine-tuning to teach the retrieval protocol while preventing the model from memorizing retrieved evidence. We then apply trajectory-aware reinforcement learning to improve source selection, query construction, and stopping behavior.

Experiments on Chinese legal benchmarks validate the effectiveness of Legal-R1. Compared with existing RAG-based legal systems, Legal-R1 achieves stronger performance on LexEval and UCL-Bench, showing that actively planning retrieval is more effective than attaching a fixed retriever to an LLM. Ablation and retrieval-quality analyses further show that post-training improves evidence-seeking behavior and that authority-aware and diversity-aware precedent reranking improves the reliability of retrieved evidence. These results support our central claim that legal reasoning benefits from learning an explicit evidence-seeking policy.

Our contributions are summarized as follows:

- We introduce a multi-agent workflow for evidence-based legal reasoning, where the Lawyer, Legal Assistant, and Judge agents separately handle evidence planning, legal retrieval, and answer generation. This workflow supports comprehensive, authoritative, and diverse evidence grounding.
- We propose Legal-R1, an agentic legal retrieval framework that post-trains the Lawyer agent to learn an autonomous evidence-seeking policy, including whether to retrieve, which source to query, what evidence to retrieve, and when to stop.
- We conduct experiments and ablations on multiple Chinese legal benchmarks, demonstrating that Legal-R1 improves legal reasoning over retrieval-augmented baselines and that its gains come from both learned retrieval policies and legal-evidence-aware reranking.

2 Related Work

Retrieval-Augmented Generation Retrieval-Augmented Generation (RAG) mitigates factual hallucinations by dynamically fetching external knowledge, evolving from basic embedding lookups to hybrid frameworks that merge lexical and semantic search (Lewis et al., 2020; Karpukhin et al., 2020). To handle massive document pools, recent paradigms employ large language models as re-rankers to prioritize relevant passages (Sun et al., 2023; Nogueira et al., 2020). However, conventional RAG systems treat documents as independent text blocks, relying on surface-level semantic overlaps while completely ignoring domain-specific structured meta-attributes like judicial hierarchies or sentencing variations (Gao et al., 2023). In contrast, we propose a specialized judicial reranking mechanism integrating an authority hierarchy score and a punishment severity index (PSI). By incorporating these domain priors into a modified maximal marginal relevance formulation (Carbonell and Goldstein, 1998), our approach simultaneously optimizes both statutory authority and retrieval diversity.

Tool-Augmented Agents Tool-augmented agents empower language models to autonomously execute external APIs, reason through complex environments, and solve multi-step tasks (Yao et al., 2022; Schick et al., 2023). Recent breakthroughs leverage post-training reinforcement learning—such as Group Relative Policy Optimization (GRPO)—to scale test-time compute and cultivate long-term thinking behaviors for autonomous error correction (Shao et al., 2024; Guo et al., 2025). Despite their success, existing frameworks heavily rely on deterministic, mathematically

verifiable environments where outcomes are definitively checked by code interpreters or symbolic solvers (Cobbe et al., 2021). When applied to non-deterministic, knowledge-dense fields like law, they suffer from severe supervision bottlenecks due to the absence of absolute verification functions (Hendrycks et al., 2021). Our approach bridges this gap through a hierarchical multi-agent workflow embedded in a continuous deliberation loop. We adapt GRPO to this non-deterministic tool-use paradigm via a multi-tier dense reward function anchored by a fine-tuned critique model (Ouyang et al., 2022).

LLMs and Agents in the Legal Domain The deployment of large language models in computational law has transformed legal natural language processing (Xiao et al., 2018). Existing efforts either prompt general models with legal instructions or fine-tune them on domain-specific corpora (Fei et al., 2024). Recently, multi-agent legal pipelines have emerged to simulate role-differentiated behaviors, alongside synthetic data pipelines to compile large-scale training trajectories (Yue et al., 2023; Wang et al., 2023). Nevertheless, standard legal models function within rigid, single-turn setups that struggle with fluid reasoning, while automated data generation remains highly susceptible to training-test contamination and option overlaps (Xu et al., 2024). Our work addresses these gaps by introducing an end-to-end automated data synthesis and purification pipeline. This framework implements strict algorithmic safeguards, including fingerprint-based contamination filtering and heuristic negative sampling for distractors, ensuring data purity and legal logical uniqueness.

3 Method

3.1 Problem Formulation

Given a legal question q, the task is to generate an evidence-grounded answer a based on a heterogeneous legal knowledge base $\mathcal{K}$. In our setting, $\mathcal{K}$ consists of legal norms and judicial precedents.

$$\mathcal{K} = \mathcal{K}_{norm} \cup \mathcal{K}_{case},$$

where $\mathcal{K}_{norm}$ contains laws, regulations, and judicial interpretations, and $\mathcal{K}_{case}$ contains criminal judgment cases.

Legal evidence is not judged by relevance alone. A high-quality answer should be grounded in evidence that is *comprehensive*, *authoritative*, and *diverse*. It should cover the key legal issues in q, rely on valid and authoritative legal sources, and include analogous precedents that reflect meaningful variation in fact patterns and judicial outcomes. The central challenge is therefore to decide what evidence should be retrieved, from which legal source, and when the collected evidence is sufficient for answering.

3.2 Agentic Legal Retrieval

We solve the evidence-based legal task through three components: a Lawyer Agent, a Legal Research Agent, and a Judge Agent. The Lawyer Agent serves as the central planner: it analyzes the user question, identifies missing legal evidence, and actively coordinates a multi-turn retrieval process. At each turn, it issues a source-specific retrieval request to the Legal Research Agent, which searches heterogeneous legal sources and returns the retrieved evidence. The Lawyer Agent then updates its reasoning state based on the evidence and decides whether additional retrieval is needed. Once the Lawyer Agent judges that the collected evidence is sufficient, the Judge Agent synthesizes the retrieval trajectory into the final evidence-grounded answer.

3.2.1 Lawyer Agent

Given q, the Lawyer agent identifies the legal issues, determines what type of evidence is missing, and constructs a multi-turn retrieval trajectory.

$$\tau = (r_1, u_1, s_1, e_1, \ldots, r_T, u_T, s_T, e_T)$$

where r_j is the reasoning state, u_j is the retrieval action, s_j is the search query, and e_j is the retrieved evidence at turn j. The action space is defined as

$$u_j \in \{\textsc{Statute}, \textsc{Precedent}, \textsc{Stop}\}$$

Here, STATUTE requests legal provisions or judicial interpretations, PRECEDENT requests analogous criminal cases, and STOP indicates that the current evidence is sufficient for answering. The Lawyer agent therefore decides whether the next step should clarify the legal basis, compare precedents, refine a failed query, or terminate retrieval. After each returned evidence unit, it performs a rethink step to assess whether the evidence resolves the current uncertainty and whether the trajectory should continue with the same source, switch to another source, or stop.

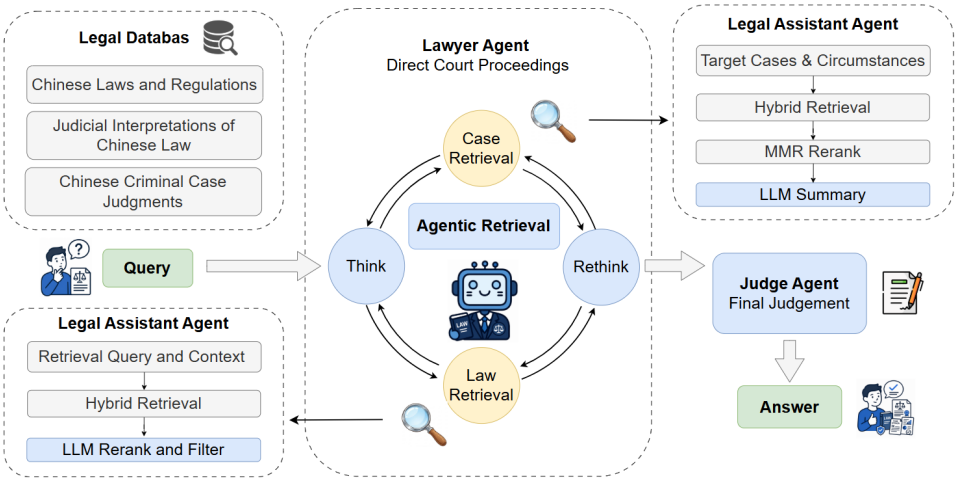

Figure 2: Overview of the proposed agentic legal retrieval workflow.

3.2.2 Legal Assistant Agent

The Legal Assistant agent executes the Lawyer agent's retrieval requests over $\mathcal{K}$ and returns compact evidence units.

Intent-Aware Statute Retrieval. For STATUTE, the Lawyer agent provides a search query and a short retrieval intent or context, such as locating a sentencing range, identifying the elements of an offense, or finding an exception clause. The Legal Assistant agent first retrieves candidate provisions from $\mathcal{K}_{norm}$ using the query. It then reranks the candidates with an LLM-based normative reranker $\mathcal{L}_{norm}$ conditioned on the retrieval intent, so that the returned provisions match the specific legal role requested by the Lawyer agent.

Authority-Aware Diverse Precedent Retrieval. For PRECEDENT, the Legal Assistant agent retrieves a candidate set C from $\mathcal{K}_{case}$ using the target fact description and optional circumstances. Since semantic similarity alone may return low-authority or outcome-redundant cases, we adapt Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) to select precedents with both court authority and sentencing-outcome diversity. For each candidate $d_i \in C$, the base score is

$$\text{Base}(d_i) = \alpha \cdot \text{Rel}(d_i) + (1 - \alpha) \cdot \text{Auth}(d_i)$$

where $\text{Rel}(d_i)$ is the retrieval relevance score, $\text{Auth}(d_i)$ is the normalized court-authority score, and α controls their trade-off. Given selected cases S, the final MMR-style score is

$$M(d_i \mid S) = \text{Base}(d_i) - \beta \cdot \max_{d_j \in S} \text{Sim}_{\text{PSI}}(d_i, d_j)$$

where β controls the diversity penalty and $\text{Sim}_{\text{PSI}}(d_i, d_j) = 1 - \frac{|\text{PSI}_i - \text{PSI}_j|}{d_{\max} + \epsilon}$ measures similarity in sentencing severity. Here, $d_{\max}$ normalizes PSI distance and ϵ ensures numerical stability. Section 4.1 details the authority annotation and PSI construction. After selection, an LLM-based case summarizer $\mathcal{L}_{case}$ condenses each precedent into a compact evidence unit with key facts, court level, judgment outcome, and sentencing information.

3.2.3 Judge Agent

The Judge agent directly generates the final evidence-grounded answer. We denote the answer-generation LLM as $\mathcal{L}_{Judge}$. Given the legal question q and the evidence collected along the retrieval trajectory τ, the Judge agent produces

$$a = \mathcal{L}_{Judge}(q, \tau)$$

The prompt instructs $\mathcal{L}_{Judge}$ to answer based on the retrieved legal norms and precedents, and the full prompt is provided in the appendix A.

3.3 Lawyer Agent Training

We train the Lawyer agent to learn an active evidence-seeking policy for legal retrieval. It learns when to retrieve, what evidence to seek, how to revise source-specific queries, and when to stop. Training only the Lawyer agent decouples evidence collection from answer generation, allowing the learned policy to be paired with different Judge agents in downstream applications.

Trajectory Synthesis. We synthesize tool-use trajectories by simulating the inference-time interaction between the Lawyer agent and the Legal Assistant agent. Given a legal question q, a teacher model $\mathcal{L}_{Teacher}$ plays the Lawyer agent and generates a reasoning state r_j, a retrieval action u_j,

and a source-specific query s_j. When $u_j \neq$ STOP, the Legal Assistant agent uses s_j to invoke the corresponding retrieval module and returns evidence e_j. The teacher conditions on the updated context to decide whether to refine the query, switch the evidence source, or stop retrieval. The process terminates when $\mathcal{L}_{Teacher}$ emits STOP and produces the final answer a. Each training instance records the complete evidence-seeking process.

$$D = \{(q^{(i)}, r_1^{(i)}, u_1^{(i)}, s_1^{(i)}, e_1^{(i)}, \ldots, r_{T_i}^{(i)}, u_{T_i}^{(i)}, s_{T_i}^{(i)}, e_{T_i}^{(i)}, a^{(i)})\}_{i=1}^{N}$$

where D is the synthesized trajectory dataset, N is the number of training instances, and T_i is the number of retrieval turns in instance i.

Evidence-Masked Supervised Fine-Tuning. We use supervised fine-tuning to teach the Lawyer agent the protocol of agentic legal retrieval. Each sample in D is serialized into a token sequence $x = (x_1, \ldots, x_L)$. We optimize the causal language modeling loss with an evidence mask:

$$\mathcal{L}_{\text{SFT}} = -\sum_{t=1}^{L} m_t \cdot \log p_\theta(x_t \mid x_{<t}),$$

where $m_t = 0$ for tokens belonging to retrieved evidence e_j and $m_t = 1$ otherwise. This masking prevents the model from learning to reproduce retrieved statutes or cases, and instead focuses training on reasoning states, retrieval actions, and query revision.

Trajectory-Aware Reinforcement Learning. Starting from the SFT model, we use Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to further train the Lawyer agent. For each question q, the current model samples a retrieval trajectory $\hat{\tau}$ and a final answer $\hat{a}$. The reward compares them with the synthesized trajectory τ and reference answer a in D. It encourages the Lawyer agent to collect useful evidence, use valid queries and formats, and avoid unnecessary retrieval turns. The total reward is

$$\begin{aligned} R(\hat{\tau}, \hat{a} \mid q, \tau, a) = & w_1 R_{\text{acc}}(\hat{a}, a) + w_2 R_{\text{align}}(\hat{\tau}, \tau) \\ & + w_3 R_{\text{gain}}(\hat{e}, a) + w_4 R_{\text{query}}(\hat{s}) \\ & + R_{\text{fmt}}(\hat{\tau}, \hat{a}) - \rho(\hat{\tau}, \hat{a}) - \eta\, \text{KL} \end{aligned}$$

where $\hat{e}$ and $\hat{s}$ denote the evidence and search queries in $\hat{\tau}$, R_{acc} scores the final answer, R_{align} compares the sampled trajectory with the synthesized trajectory, R_{gain} checks whether the retrieved evidence supports the answer, R_{query} evaluates query quality, R_{fmt} checks the tool-use format, and ρ penalizes verbosity and repeated retrieval loops. The weights $w_1, \ldots, w_4$ balance these terms. KL is the GRPO regularization term and η controls its weight.

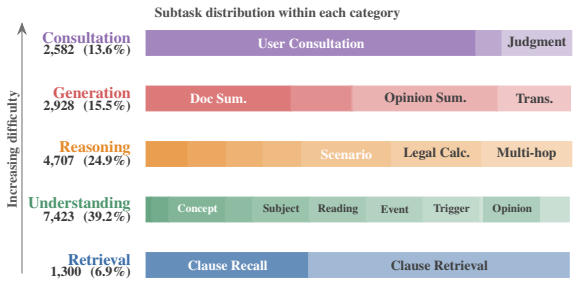

Figure 3: Distribution of the training data across major legal task categories and fine-grained subtasks. Detailed statistics are provided in Appendix D.

4 Data Construction

The dataset utilized in this project consists of two components: a legal knowledge base $\mathcal{K}$ and training data for legal agent D.

4.1 Legal Knowledge Base

Normative Legal Corpus. We construct $\mathcal{K}_{norm}$ from the National Database of Laws and Regulations[2]. Each statutory article or judicial interpretation is treated as an individual retrieval unit and is annotated with its law title, chapter, section, article number, and effective date. The processed corpus contains 296 active laws with 43,117 articles and 555 judicial interpretations with 6,370 articles.

Criminal Precedent Corpus. We construct $\mathcal{K}_{case}$ from LeCaRDv2[3], which contains 53,277 criminal judgment records. Each case is retained as a precedent-level retrieval unit and is associated with structured attributes including the court level, case facts, judgment outcome, and sentencing result. This corpus supports precedent retrieval for analogous facts and sentencing comparison.

Authority and Sentencing Annotations. To support authority-aware and diversity-aware precedent retrieval, we add two annotations to $\mathcal{K}_{case}$. First, cases are grouped by the level of the adjudicating court, including high people's courts, intermediate people's courts, and primary people's courts or other local tribunals, and each group is mapped to

[2] https://flk.npc.gov.cn/index
[3] https://github.com/THUIR/LeCaRDv2

Model	Retrieval	Scale	Lawbench		LexEval		Avg
			Trad. Score	LLM Judge	Trad. Score	LLM Judge	
LawGPT	✗	7B	11.33	13.76	9.49	9.18	10.94
Luwen	✗	7B	30.06	32.77	14.03	16.97	23.45
LegalOne	✗	8B	23.25	41.02	35.30	43.79	35.84
Qwen3-8B	✗	8B	24.70	40.20	18.04	28.47	27.85
A2search	✓	7B	46.82	49.14	35.23	37.94	42.28
R-Search	✓	7B	43.01	47.44	45.05	46.85	45.59
Qwen3-8B-Embedding	✓	8B	32.92	43.25	35.79	41.40	38.34
Llama-Embed-Nemotron	✓	8B	32.23	42.28	36.45	41.77	38.19
DeepSeek-V4-Flash	✗	284B	62.40	68.77	61.81	72.90	66.47
GPT-5.4-Mini	✗	–	54.83	62.83	50.50	59.32	56.87
Legal-R1	✓	8B	57.36	63.97	56.54	63.28	60.28

Table 1: Model Performance Comparison on LexEval,and Lawbench Benchmarks. Judge by DeepSeek-V4-Flash.

an authority score. Second, the Punishment Severity Index (PSI) maps heterogeneous sentencing outcomes into a numerical severity value. Main penalties are converted into day-equivalent values and adjusted by penalty type and suspension factors, while supplementary penalties are added as extra severity terms:

$$\text{PSI} = D \cdot \gamma \cdot \phi + \sum S_{\text{sup}}$$

where D is the day-equivalent main penalty, γ is the penalty-type coefficient, ϕ is the suspension factor, and S_{sup} denotes supplementary penalties such as fines or deprivation of political rights. Detailed authority scores and PSI examples are provided in Appendix C.

4.2 Training Data Construction

We construct the post-training dataset through an automated data synthesis and purification pipeline. The data are collected from established legal benchmarks and authoritative examination resources, including CAIL (Xiao et al., 2018), DISC-LawLLM (Yue et al., 2023), CrimeKgAssitant[4] , and the National Unified Legal Professional Qualification Examination[5], and are further complemented with domain-specific synthetic problem sets. We then converts heterogeneous legal materials into task-specific QA pairs while applying multiple quality-control mechanisms. Specifically, the pipeline first performs task-driven applicability gating to filter source texts that are unsuitable for a given task, then removes potentially contaminated samples through N-gram fingerprint matching against held-out benchmarks, and finally applies automatic leakage detection and rule-based distractor construction to improve the reliability of open-ended and multiple-choice questions. The resulting dataset contains 18,940 high-quality structured QA instances, covering five major legal domains and 29 fine-grained subcategories. Figure 3 shows the distribution of the training data across these categories. The detailed distribution and construction details are provided in Appendix D.

5 Experiments

5.1 Setup

Benchmark. We evaluate our method on LexEval (Li et al., 2024) and LawBench (Fei et al., 2024), two widely used Chinese legal benchmarks. LexEval consists of 23 tasks and 14,150 questions covering six legal cognitive abilities, while LawBench includes 20 tasks across three judicial cognitive dimensions, ranging from multiple-choice questions to practical applications such as reading comprehension, criminal amount calculation, and legal consultation.

Evaluation Metrics. We evaluate the proposed method from two perspectives: *performance* and *efficiency*. Performance is measured by benchmark default scores and an *LLM-as-a-judge* score, while efficiency is assessed by retrieval turns, execution time, tool-invocation time, and token consumption.

Implementation Details. We train Legal-R1 from Qwen3-8B using SFT followed by GRPO. The SFT stage runs for 3 epochs on D_{SFT} with a learning rate of 5×10^{-6} and a batch size of 32. The GRPO stage uses a learning rate of 1×10^{-6}, a batch size of 32, 8 rollouts, and $\eta = 0.08$. Training is conducted on four NVIDIA H20 GPUs. At inference time, Legal-R1 serves as the lawyer model that performs multi-turn reasoning and retrieval. A hybrid retriever and an auxiliary Qwen3-8B assistant model refine the retrieved evidence, while another Qwen3-8B judge model generates the final evidence-grounded answer. Full deployment details are provided in

[4] https://github.com/liuhuanyong/CrimeKgAssitant

[5] https://www.moj.gov.cn/jgsz/jgszzsdw/zsdwgjsfkszx/

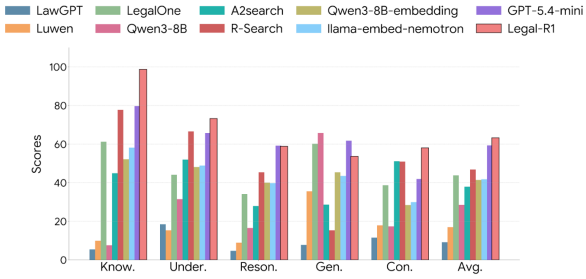

Figure 4: Model Performence on Lexeval's Five Categories.

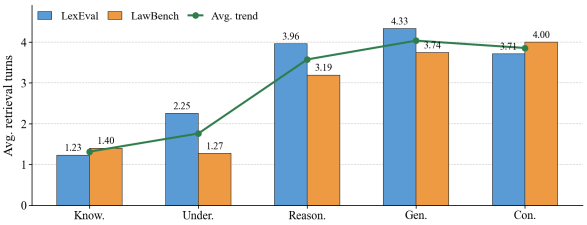

Figure 5: Average Retrieval Turn for Different Complexity Tasks on Lawbench and Lexeval Benchmark.

Appendix E.

Baselines. We compare our model against three categories of baseline LLMs: (1) General LLMs, represented by DeepSeek-V4-Flash(DeepSeek-AI, 2026),and GPT-5.4-Mini(OpenAI, 2026); (2) Legal-Specific LLMs, including LawGPT(Zhou et al., 2024), LegalOne(Li et al., 2026), and Luwen(Wu et al., 2026); and (3) RAG-based LLMs, which include A2search(Zhang et al., 2025) and R-Search(Zhao et al., 2025).

During evaluation, we configure each baseline LLM with its native prompt template (if available). For RAG-enabled baselines, we integrate their respective RAG components and load the same retrieval knowledge base used by our agentic retrieval framework. We then execute a unified evaluation pipeline: the primary models and the judge model (Qwen3-8B) are deployed via vLLM to generate responses across all benchmarks and compute the final scores. Concurrently, we track key inference metrics, including the number of retrieval turns, execution time (latency), and token consumption.

5.2 Results and Discussion

Main Results We evaluate various baselines on legal tasks across Lawbench and LexEval, as shown in Table 1. The empirical results indicate that while traditional legal-specific models and RAG-augmented methods show competitive performance, they are consistently outperformed by our approach. Our method, Legal-R1, performs exceptionally well across all datasets and evaluation metrics, achieving a top-tier overall average score of 60.28%. Specifically, at the 8B parameter scale, Legal-R1 significantly outperforms the strongest RAG baseline, R-Search, by 14.69 points in overall average. Furthermore, our 8B method even surpasses the larger proprietary model, GPT-5.4-Mini, by 3.41 points. Legal-R1 ranks second only to the massive DeepSeek-V4-Flash (284B), demonstrating its remarkable effectiveness and parameter efficiency in complex legal reasoning.

Efficiency As illustrated in Figure 5, our adaptive retrieval mechanism demonstrates exceptional efficiency. For less complex tasks (*Knowledge*, *Understanding*), the system conservatively limits retrieval to an average of 1.2–1.7 turns, directly leveraging the LLM's internal knowledge base to minimize computational overhead. In contrast, for highly complex tasks (*Reasoning*, *Generation*, *Consultation*), the retrieval frequency scales up to 3.1–4.3 turns, enabling multi-hop information synthesis. Crucially, this dynamic adjustment trend remains highly consistent across both LawBench and LexEval benchmarks, proving that our model achieves a Pareto-optimal balance between performance and computational cost across varying task complexities.

Ablation Study To verify the effectiveness of each component in our framework, we conduct an ablation study on Lawbench and LexEval, as reported in Table 2.

First, the results underscore that incorporating external knowledge is paramount for legal reasoning; equipping the vanilla Qwen3-8B with hybrid retrieval ('w/ hybrid') drastically boosts the overall average score from 27.85% to 55.91%. Second, domain-specific adaptation without retrieval ('w/ post-train') also yields a substantial improvement (+18.03 points on average), validating the necessity of legal domain fine-tuning. Finally, our complete method, Legal-R1 , integrates domain post-training, the multi-stage RAG pipeline, and GRPO reinforcement learning to achieve the highest average score of 60.28%. This final leap demonstrates that these components provide orthogonal benefits, culminating in optimal synergistic performance in complex legal scenarios.

Case Study To intuitively illustrate the autonomous multi-turn reasoning and retrieval capabilities of Legal-R1, we conduct a qualitative case

Model	Retrieval	Scale	Lawbench		LexEval		Avg
			Trad. Score	LLM Judge	Trad. Score	LLM Judge	
Qwen3-8B	✗	8B	24.70	40.20	18.04	28.47	27.85
Qwen3-8B w/ post-train	✗	8B	26.24	38.98	55.51	62.78	45.88
Qwen3-8B w/ hybrid	✓	8B	52.02	59.85	50.52	61.49	55.91
Qwen3-8B w/ rerank	✓	8B	53.28	57.97	50.12	57.31	54.67
Legal-R1 w/ hybrid	✓	8B	57.08	63.69	55.47	61.96	59.55
Legal-R1 w/ rerank	✓	8B	57.36	63.97	56.54	63.28	60.28

Table 2: Ablation study: Model Performance Comparison on LexEval, Lawbench and average Benchmarks. Judge by DeepSeek-V4-Flash.

Case Study: Autonomous Multi-Turn Retrieval	
Query	Will a debtor automatically be blacklisted if they fail to repay a debt within the court's 25-day deadline?
Trajectory	💡 **Thinking Step 1:** I need to locate the foundational statutory criteria for blacklisting. ↪ 🔍 **Retrieval 1:** *Article 1*.↪ 💡 **Thinking Step 2:** Article 1 hinges on "fulfillment capability." I must retrieve its specific definition. ↪ 🔍 **Retrieval 2:** *Article 2*. ↪ 💡 **Thinking Step 3:** I need to search for procedural exemptions regarding the blacklisting process. ↪ 🔍 **Retrieval 3:** *Article 3*. ↪ 💡 **Thinking Step 4:** Let me invoke a precedent search to validate how courts apply these articles in practice.↪ 🔍 **Retrieval 4:** *Precedent Case*.
Answer	No, blacklisting is not automatic; according to Articles 1, 2, and 3, it strictly depends on whether the debtor willfully refuses to comply despite having the financial capability, provided they do not meet any legal exemptions. 👍

Table 3: A case study of Legal-R1's autonomous multi-turn retrieval, with the full trajectory provided in Table 6 in the appendix.

Model	Retrieval	Avg. Score
Qwen3-8B	✓	54.67
Qwen3-8B-SFT	✓	59.98
Legal-R1	✓	60.28

Table 4: Ablation study of training progression based on overall average scores.

study on a representative real-world legal consultation. As analyzing in Table 3, given a user query regarding enforcement penalties, the model does not rely on static memories but independently initiates a sequence of precise retrieval loops to search for statutory criteria and validation procedures for dishonest judgment debtors. Notably, it strategically executes cross-references between general rules (Article 1) and exception clauses (Article 3), while invoking a legal precedent to ground its contextual understanding in judicial practice. Ultimately, through this systematic deliberation, Legal-R1 synthesizes an accurate and legally sound judgment, demonstrating its robust reliability in professional legal applications.

SFT vs RL We isolate the distinct contributions of supervised fine-tuning (SFT) and reinforcement learning (RL) within our framework, as presented in Table 4. Transitioning from the raw RAG pipeline ('w/ rerank') to the SFT stage ('Qwen3-8B-SFT w/ rerank') yields a prominent performance leap of 5.31 percentage points, demonstrating that SFT is vital for injecting foundational legal expertise and task-specific alignment. Building upon this optimized baseline, Legal-R1 leverages GRPO reinforcement learning to further elevate the overall average score to a peak of 60.28%. This continuous gain validates that RL successfully refines the model's underlying reasoning policy, capitalizing on the specialized knowledge acquired during SFT to unlock maximum precision.

6 Conclusion

In this paper, we introduced Legal-R1, an agentic legal large language model for evidence-based question answering. Our framework couples a multi-turn Lawyer Agent with a multi-stage RAG pipeline via a Legal Assistant Agent. This pipeline uses hybrid retrieval followed by authority- and diversity-aware reranking, and leverages LLM filtering and summarization to distill compact legal evidence units. To optimize the central retrieval policy, Legal-R1 undergoes a streamlined two-stage training paradigm of Evidence-Masked Supervised Fine-Tuning (SFT) and Trajectory-Aware Reinforcement Learning via GRPO.

Empirical results on the LexEval and LawBench benchmarks demonstrate the exceptional efficacy and efficiency of our approach. At the 8B parameter scale, Legal-R1 achieves a state-of-the-art overall average score of 60.28%, surpassing the proprietary GPT-5.4-Mini. Furthermore, our model maintains high efficiency by dynamically scaling its retrieval turns according to task complexity, successfully achieving a Pareto-optimal balance between computational cost and performance.

Limitations

While Legal-R1 demonstrates robust and highly accurate performance in autonomous legal reasoning, it is not without minor limitations. Primarily, the multi-turn agentic retrieval process—although dynamically adjusted for task complexity—inherently introduces a slight inference latency compared to standard, single-pass generation models. While this computational overhead is justified by the significant gains in accuracy and legal grounding, it may require further pipeline optimization (e.g., parallelized evidence fetching or KV-cache optimization) before deployment in extremely time-sensitive, real-time legal consultation environments. Additionally, the current model and knowledge base are deeply tailored to the Chinese legal system; extending this framework to support multi-jurisdictional or multilingual legal reasoning remains a promising avenue for future work.

References

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI. 2026. DeepSeek-V4-Pro. https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro. Accessed: 2026-05-25.

Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, Fengzhe Zhou, Zhuo Han, Alan Huang, Songyang Zhang, Kai Chen, Zhixin Yin, Zongwen Shen, and 1 others. 2024. Lawbench: Benchmarking legal knowledge of large language models. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 7933–7962.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, Haofen Wang, and 1 others. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1):32.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 6769–6781.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Haitao Li, Yifan Chen, Shuo Miao, Qian Dong, Jia Chen, Yiran Hu, Junjie Chen, Minghao Qin, Yueyue Wu, Yujia Zhou, and 1 others. 2026. Legalone: a family of foundation models for reliable legal reasoning. *arXiv preprint arXiv:2602.00642*.

Haitao Li, You Chen, Qingyao Ai, Yueyue Wu, Ruizhe Zhang, and Yiqun Liu. 2024. Lexeval: A comprehensive chinese legal benchmark for evaluating large language models. *Advances in Neural Information Processing Systems*, 37:25061–25094.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the association for computational linguistics: EMNLP 2020*, pages 708–718.

OpenAI. 2026. Introducing GPT-5.4 mini and nano. https://openai.com/index/introducing-gpt-5-4-mini-and-nano/. Accessed: 2026-05-25.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 14918–14937.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508.

Yiquan Wu, Yuhang Liu, Yifei Liu, Ang Li, Siying Zhou, and Kun Kuang. 2026. Luwen technical report. *arXiv preprint arXiv:2604.06737*.

Chaojun Xiao, Haoxi Zhong, Zhipeng Guo, Cunchao Tu, Zhiyuan Liu, Maosong Sun, Yansong Feng, Xianpei Han, Zhen Hu, Heng Wang, and 1 others. 2018. Cail2018: A large-scale legal dataset for judgment prediction. *arXiv preprint arXiv:1807.02478*.

Cheng Xu, Shuhao Guan, Derek Greene, M Kechadi, and 1 others. 2024. Benchmark data contamination of large language models: A survey. *arXiv preprint arXiv:2406.04244*.

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*.

Shengbin Yue, Wei Chen, Siyuan Wang, Bingxuan Li, Chenchen Shen, Shujun Liu, Yuxuan Zhou, Yao Xiao, Song Yun, Xuanjing Huang, and 1 others. 2023. Disc-lawllm: Fine-tuning large language models for intelligent legal services. *arXiv preprint arXiv:2309.11325*.

Fengji Zhang, Xinyao Niu, Chengyang Ying, Guancheng Lin, Zhongkai Hao, Zhou Fan, Chengen Huang, Jacky Keung, Bei Chen, and Junyang Lin. 2025. AΘ2 search: Ambiguity-aware question answering with reinforcement learning. *arXiv preprint arXiv:2510.07958*.

Qingfei Zhao, Ruobing Wang, Dingling Xu, Daren Zha, and Limin Liu. 2025. R-search: Empowering llm reasoning with search via multi-reward reinforcement learning. *arXiv preprint arXiv:2506.04185*.

Zhi Zhou, Jiang-Xin Shi, Peng-Xiao Song, Xiao-Wen Yang, Yi-Xuan Jin, Lan-Zhe Guo, and Yu-Feng Li. 2024. Lawgpt: A chinese legal knowledge-enhanced large language model. *arXiv preprint arXiv:2406.04614*.

A Prompts Used in the Agentic Retrieval Workflow

This appendix provides the prompts used by the LLM-based components in the proposed workflow.

A.1 Normative Evidence Reranking Prompt

This prompt is used by the LLM-based normative reranker $\mathcal{L}_{norm}$. Given a retrieval intent and candidate legal provisions from $\mathcal{K}_{norm}$, the model ranks the candidates according to their relevance to the requested legal role, such as offense elements, sentencing ranges, or exception clauses. The detailed template implementation is illustrated in Figure 7.

A.2 Precedent Evidence Summarization Prompt

This prompt is used by the case summarizer $\mathcal{L}_{case}$. Given a selected precedent, the model compresses it into an evidence unit that preserves the key facts, court level, judgment outcome, and sentencing information. The complete template configuration is detailed in Figure 8.

A.3 Judge Answer Generation Prompt

This prompt is used by the Judge agent $\mathcal{L}_{Judge}$. Given the legal question and the collected retrieval trajectory, the model generates the final answer based only on the retrieved legal norms and precedents, while avoiding unsupported claims. The structured implementation of this template is presented in Figure 9.

B Full Trajectory on Case Study

B.1 Case Study

To intuitively illustrate the autonomous multi-turn reasoning and retrieval capabilities of Legal-R1, we conduct a qualitative case study on a representative real-world legal consultation. As analyzing in Figure 10, given a user query regarding enforcement penalties, the model does not rely on static memories but independently initiates a sequence of precise retrieval loops to search for statutory criteria and validation procedures for dishonest judgment debtors. Notably, it strategically executes cross-references between general rules (Article 1) and exception clauses (Article 3), while invoking a legal precedent to ground its contextual understanding in judicial practice. Ultimately, through this systematic deliberation, Legal-R1 synthesizes an accurate and legally sound judgment, demonstrating its robust reliability in professional legal applications.

C Legal Knowledge Base Annotation Details

C.1 Authority Tier

We use the level of the adjudicating court as a proxy for the authority of a precedent case. Since judgments from higher-level courts generally carry stronger reference value, each case is mapped to one of three authority tiers and receives a tier-specific score for precedent reranking:

- **Level 1** consists of cases adjudicated by high people's courts. These cases receive an authority score of 1.1 and include 195 cases in LeCaRDv2.
- **Level 2** consists of cases adjudicated by intermediate people's courts. These cases receive an authority score of 1.05 and include 3,731 cases in LeCaRDv2.
- **Level 3** consists of cases adjudicated by primary people's courts or other local tribunals. These cases receive an authority score of 1.0 and include 49,351 cases in LeCaRDv2.

These scores provide a lightweight prior rather than a hard filter. Cases from lower courts can still be selected when they are more relevant or contribute distinct sentencing evidence.

C.2 Punishment Severity Index

To systematically evaluate judicial outcomes, we develop the Punishment Severity Index (PSI), a computational framework that converts heterogeneous sentencing descriptions into a continuous numerical scale.This ensures that linguistic variations in judicial texts (e.g., "three years" vs. "36 months") are standardized into uniform computational inputs.

The core of the PSI(Punishment Severity Index) is based on a weighted temporal-equivalent model. Main penalties are first converted into a day-equivalent baseline (D), which is then adjusted by a Severity Coefficient (α) and a Probation Adjustment Factor (ϕ). Specifically, fixed-term imprisonment serves as the baseline ($\alpha = 1.0$), while criminal detention and public surveillance are assigned lower weights ($\alpha = 0.8$ and 0.3, respectively) to reflect their lower degree of liberty deprivation. Furthermore, if a suspended sentence is detected, a multiplier of $\phi = 0.2$ is applied. For extreme non-temporal penalties, such as life imprisonment or the death penalty, the index assigns discrete, high-order ordinal values (e.g., $15,000$ to $100,000$) to maintain the hierarchical integrity of the penal system.

Finally, the index incorporates Supplementary Penalties through additive weighting. To account for the diminishing marginal impact of monetary sanctions, fines and property forfeitures are integrated using a logarithmic scaling function: $S_{fine} = 20 \times \log_{10}(\text{amount} + 1)$. Other sanctions, including the deprivation of political rights and deportation, are assigned fixed weights based on their statutory duration or severity. The total PSI is the cumulative sum of these components:

$$PSI = (D \cdot \alpha \cdot \phi) + \sum S_{supplementary}$$

This integrated metric provides a robust basis for subsequent statistical analysis and cross-case comparisons of sentencing severity.

D Details of Training Data Construction

This appendix provides additional details on the construction of the post-training dataset used in our legal agent model. The goal of data construction is to obtain high-quality, evidence-grounded, and tool-use-oriented legal QA instances that can improve the model's ability to understand legal facts, invoke external tools, retrieve relevant legal knowledge, and generate reliable answers.

D.1 Data Sources and Task Coverage

Our post-training dataset consists of 18,940 structured QA instances, covering 5 major legal domains and 29 fine-grained subcategories, detailed in Table 6. The raw materials are collected from a combination of public legal NLP benchmarks, authoritative legal examination resources, and domain-specific synthetic problem sets. Specifically, the primary data sources include the Chinese AI and Law Challenge (CAIL), DISC-LawLLM, CrimeKgAssitant, and the National Unified Legal Professional Qualification Examination. These resources provide diverse legal materials, including statutory provisions, case facts, legal reasoning questions, criminal law problems, civil law problems, procedural law questions, and examination-style multiple-choice questions.

To improve task diversity, we organize the dataset according to a task-oriented configuration profile. Each task type is associated with a dedicated task description, input format, output format, and generation prompt. This design allows the

Penalty Case	Calculation Process	PSI Result
死刑 (Death penalty)	$100,000$	100,000.0
无期徒刑，剥夺政治权利终身 (Life imprisonment with lifelong deprivation of political rights)	$15,000 + 2,000$	17,000.0
有期徒刑十五年 (Fixed-term imprisonment of fifteen years)	$(15 \times 12 \times 30) \times 1.0$	5,400.0
有期徒刑三年，缓刑四年 (Fixed-term imprisonment of three years with a four-year suspension of sentence)	$(3 \times 12 \times 30) \times 1.0 \times 0.2$	216.0
拘役六个月，并处罚金一万元 (Criminal detention of six months, with a concurrent fine of 10,000 RMB)	$(6 \times 30) \times 0.8 + 20 \times \log_{10}(10,000 + 1)$	224.0
管制一年 (Public surveillance for one year)	$(1 \times 12 \times 30) \times 0.3$	108.0
驱逐出境 (Deportation)	$1,000$	1,000.0
免予刑事处罚 (Exemption from criminal punishment)	0	0.0

Table 5: Examples of PSI Calculation for Different Penalties.

pipeline to convert heterogeneous source materials into structured training instances under a unified schema. The resulting QA pairs cover both open-ended verifiable questions and legal selection tasks, enabling the model to learn different forms of legal reasoning, including fact understanding, legal norm matching, punishment prediction, legal element identification, and multi-step evidence-grounded answering.

D.2 Overview of the Automated Construction Pipeline

To construct the dataset at scale while maintaining data quality, we design an end-to-end automated synthesis and purification pipeline. Given raw legal materials, the pipeline first determines whether each source text is suitable for a target task, then synthesizes candidate QA pairs, and finally applies a series of automatic quality-control procedures. The overall pipeline consists of four major stages: task-driven applicability gating, fingerprint-based contamination filtering, automated answer-leakage suppression, and rule-driven distractor design.

Formally, let $\mathcal{D}_{raw}$ denote the collection of raw legal materials, and let $\mathcal{T}$ denote the set of target task types. For each source text $x \in \mathcal{D}_{raw}$ and each task type $\tau \in \mathcal{T}$, the pipeline first evaluates whether x contains sufficient information to support task τ. If the source text passes this applicability check, the model generates one or more candidate QA instances. Each candidate instance is then validated by contamination detection and task-specific quality filters. Only instances that pass all filtering steps are retained in the final post-training dataset $\mathcal{D}_{train}$.

D.3 Task-Driven Two-Stage Applicability Gating

A major challenge in legal data synthesis is that not every source text is suitable for every downstream task. For example, a statutory article may be appropriate for legal norm identification but may not contain sufficient factual information for punishment prediction. Similarly, a case fact description may support legal issue analysis but may not provide enough explicit evidence for a multiple-choice question about procedural requirements. Directly generating QA pairs from mismatched source materials can therefore introduce unanswerable questions, hallucinated answers, or weakly grounded supervision signals.

To address this issue, we implement a task-driven two-stage applicability gating mechanism before QA synthesis. In the first stage, the pipeline performs a coarse task-material matching check based on metadata, source type, and task description. This step filters out clearly incompatible combinations, such as applying a sentencing-oriented task to a source text that contains only abstract legal definitions.

In the second stage, the pipeline performs a model-based feasibility pre-vetting. Given a source

Main Category	Sub-category	Count
Knowledge Retrieval	Clause Recall	500
	Clause Retrieval	800
	Subtotal	*1,300*
Understanding	Case Similarity	101
	Dispute Identification	300
	Concept Comprehension	1,000
	Marriage Assessment	454
	Subject Identification	1,000
	Reading Comprehension	1,000
	Event Detection	1,000
	Trigger Extraction	1,000
	Relation Understanding	68
	Plaintiff Opinion	1,000
	Document Correction	500
	Subtotal	*7,423*
Reasoning	Charge Prediction	460
	Charge with law Prediction	426
	Amount Calculation	401
	Sentence Prediction	420
	Scenario Analysis	1,000
	Legal Calculation	1,000
	Multi-hop Q&A	1,000
	Subtotal	*4,707*
Generation	Document Summarization	1,000
	Indictment Generation	420
	Contract Generation	8
	Opinion Summarization	1,000
	Term Translation	500
	Subtotal	*2,928*
Consultation	User Consultation	2,000
	Evidence Opinion	158
	Case Judgment	424
	Subtotal	*2,582*
Total		**18,940**

Table 6: Distribution of Dataset Categories and Subcategories.

text and the target task description, the model is instructed to explicitly judge whether the material contains sufficient factual predicates, legal entities, or normative conditions required by the task. The model must output a binary decision together with a brief justification. If the material lacks key factual disclosures, contains insufficient legal context, or cannot support a verifiable answer, the corresponding task is skipped and recorded as a discarded instance.

This two-stage design filters low-quality candidates before generation rather than after generation. It reduces unnecessary synthesis costs and prevents the pipeline from producing superficially fluent but legally unsupported QA pairs.

D.4 Fingerprint-Based Data Contamination Filtering

To reduce the risk of benchmark leakage, we incorporate a fingerprint-based contamination filtering module. Before running the data synthesis pipeline, all designated evaluation benchmarks are cleaned and normalized to construct a global fingerprint library. The normalization process removes redundant whitespace, standardizes punctuation, and unifies common formatting variants. Each benchmark sample is then decomposed into a set of textual fingerprints based on N-gram spans.

During data construction, every candidate source text is compared against the fingerprint library before it is used for QA synthesis. If the source text contains an N-gram segment that overlaps with the held-out benchmark fingerprints above a predefined threshold, the sample is treated as potentially contaminated and is removed from the generation process. This step is applied at the source-text level rather than only at the final QA-pair level, because a generated question may paraphrase a contaminated source and become difficult to detect after synthesis.

The contamination filter is designed to be conservative. Its purpose is not to remove all topically similar legal materials, since different legal cases may naturally involve similar statutes or legal concepts. Instead, it aims to eliminate near-duplicate or directly overlapping textual evidence from held-out benchmarks, thereby reducing the possibility that evaluation performance is inflated by memorization.

D.5 Automated Syntax Parsing and Answer Leakage Suppression

For open-ended verifiable QA tasks, it is important that the question does not reveal the ground-truth answer. This is especially critical in agentic retrieval settings, where the model is expected to decide whether to invoke tools, retrieve legal evidence, and reason from the retrieved information. If the answer is already exposed in the question, the instance becomes trivial and no longer provides useful supervision for retrieval or legal reasoning.

We therefore design an automated answer-leakage suppression module. Given a generated QA pair (q, a), the module performs two layers of validation. First, it checks whether the answer string a appears directly in the question q. This direct substring matching is effective for detecting

explicit leakage, such as questions that already contain the target charge, statute number, sentencing result, or legal conclusion.

Second, the module applies regular-expression-based pattern matching to identify sensitive answer-bearing values. For example, in punishment-related questions, the parser detects sentencing expressions, penalty types, durations, fines, suspension conditions, and other legally meaningful values. If such values appear in the question and match the expected answer format, the instance is flagged as answer-leaking and discarded. This rule is particularly useful when the answer is not copied verbatim but appears in a slightly different surface form.

After leakage filtering, the remaining questions are more suitable for evaluating and training the model's ability to infer answers from legal evidence rather than relying on superficial cues in the question itself.

D.6 Rule-Driven Heuristic Distractor Design for Legal Selection Tasks

Multiple-choice legal questions require high-quality distractors. If distractor options are too obviously wrong, the task becomes overly simple; if they are semantically overlapping with the correct answer, the question may become ambiguous or legally invalid. To avoid these issues, we do not rely solely on unconstrained model generation for option construction. Instead, we design a rule-driven heuristic distractor generation module.

Given a source text and the corresponding correct answer, the module constructs negative options through two main corruption strategies.

Value Swapping. The first strategy creates distractors by swapping authentic values across legal entities or factual roles. For example, if a case description contains multiple actors, amounts, dates, or sentencing results, the algorithm reassigns a correct value from one entity to another incorrect entity. Since the values are drawn from the original text, the generated distractors remain plausible at the surface level. However, because the entity-value alignment is corrupted, the distractors are legally incorrect.

Wrong Label Assignment. The second strategy constructs distractors by assigning correct textual entities to incorrect legal labels. For example, a factual element that corresponds to one legal role, charge, or statutory condition may be deliberately mapped to another label. This creates options that are lexically grounded in the source material but invalid under legal logic.

After generating candidate distractors, the module applies validity checks to ensure that each distractor is distinct from the correct answer and from other options. The correct option and the distractors are then randomly shuffled and mapped to an answer index, such as A, B, C, or D. This randomization prevents the model from exploiting positional bias during training.

D.7 Quality Control and Final Dataset Assembly

After the above filtering and validation steps, all retained QA instances are converted into a unified training format. Each instance contains the task instruction, question, answer, optional choices, supporting evidence, and task metadata. For tool-use-oriented tasks, the instance may additionally include tool invocation annotations, intermediate evidence, or retrieval-oriented supervision signals.

The final dataset assembly process follows three principles. First, each instance should be answerable from its associated evidence or legal knowledge source. Second, the question should not contain direct answer leakage or shortcut cues. Third, the answer should be legally valid, concise, and consistent with the intended task type. Instances that violate any of these principles are removed.

Through this automated construction pipeline, we obtain a post-training dataset that is diverse in legal domain coverage, structured in task format, and reliable in supervision quality. The resulting dataset supports the model's post-training for legal question answering, legal retrieval, tool invocation, and evidence-grounded reasoning.

E Implementation and Deployment Details

This section provides the full training, deployment, and inference details of Legal-R1, including hardware allocation, auxiliary model configuration, and the multi-turn retrieval pipeline.

Using the proposed method, we train our model, Legal-R1, based on Qwen3-8B. The model is fine-tuned on D_{SFT} for 3 epochs with a learning rate of 5e-6 and a batch size of 32. Subsequently, we employ GRPO for reinforcement learning (RL) with a learning rate of 1e-6, a batch size of 32, rollout number of 8, and η set to 0.08. Regarding the hardware configuration, the training is conducted on

a single node equipped with four NVIDIA H20 GPUs (141GB VRAM each). The workloads are distributed across the GPUs to optimize resource utilization: GPU 3 hosts the hybrid retrieval module (combining BM25 and text2vec), consuming approximately 8 GB of VRAM.

Furthermore, two separate Qwen3-8B instances are deployed as auxiliary models: a rerank model utilized for filtering and summarization in the RAG pipeline, and a reward judge model employed to calculate reward functions during the RL phase; these two models are allocated 40% (approx. 56.4 GB) and 50% (approx. 70.5 GB) of GPU 3's VRAM, respectively. The Fully Sharded Data Parallel (FSDP) execution and the rollout process of the RL phase are distributed across GPU 0, GPU 1, and GPU 2.

Prior to inference, the hybrid retriever is initialized, and three distinct model instances are launched via the vLLM engine: (i) the *legal assistant model* (Qwen3-8B), which handles text summarization, filtering, and reranking within the RAG pipeline; (ii) the *judge model* (Qwen3-8B), responsible for rendering the final judgment; and (iii) the *lawyer model* (Legal-R1), which drives the reasoning and tool-invocation processes.

During the inference phase, the user query is first fed into the lawyer model. The lawyer model executes reasoning steps and generates explicit retrieval requests, which are dispatched to the hybrid retriever. The retriever performs an initial coarse-grained retrieval to obtain a candidate document pool. Subsequently, the legal assistant model is invoked to perform fine-grained re-ranking and filtering before returning the refined documents to the lawyer model. This thought-retrieval loop persists iteratively until the lawyer model determines that sufficient information has been gathered. Finally, the complete trajectory—comprising the initial query, the intermediate chain-of-thought, and the retrieved documents—is forwarded to the judge model, which synthesizes the information to generate a well-grounded and contextually accurate final response.

F The Use of Large Language Models

Using an LLM to help with paper writing During the preparation of this work, the authors utilized Large Language Models for language polishing, improving the structural clarity of the manuscript, and refining the formal expression of individual sentences. The use of Large Language Models did not influence the substantive content of the study and served solely as a writing aid.

G Potential Risks

While Legal-R1 significantly enhances the accuracy of legal question answering through its active evidence-seeking framework, we acknowledge several potential risks associated with its deployment:

- **Risk of Hallucination and Misguidance:** As an LLM-based system, Legal-R1 may still suffer from subtle hallucinations or generate plausible-sounding but incorrect legal reasoning in highly complex or edge cases. If users, especially non-professionals, rely entirely on the model's outputs for real-world litigation or legal decisions without professional verification, it could lead to severe legal or financial consequences.
- **Over-Reliance on the System:** The structured nature and highly confident tone of the multi-turn trajectories generated by Legal-R1 might inadvertently lead users to treat it as an absolute legal authority, reducing human legal practitioners' critical oversight.
- **Jurisdictional and Temporal Limitations:** The current model and its underlying heterogeneous knowledge base are strictly tailored to the contemporary Chinese legal framework. Applying this model to foreign jurisdictions, alternative legal systems (e.g., common law), or outdated legal contexts could produce entirely invalid results.

To mitigate these risks, we explicitly emphasize that Legal-R1 is designed strictly as a *supportive tool* to assist legal professionals in research and evidence collection, rather than a replacement for qualified legal counsel. We recommend that any downstream deployment of this system incorporate clear user disclaimers and human-in-the-loop validation mechanisms to ensure safety and compliance.

Case Study: Autonomous Multi-Turn Retrieval and Reasoning Trajectory

[User Query]
My friend borrowed 164,000 RMB this January, failed to repay, and was sued. We won the lawsuit, and the court ordered repayment within 25 days. If they still do not pay, will they automatically be blacklisted as a dishonest judgment debtor?

[Autonomous Reasoning & Retrieval Trajectory]
Thinking Step 1: To address whether non-repayment automatically leads to being blacklisted, I must first locate the precise statutory criteria and procedures regarding the designation of a "dishonest judgment debtor."
↪ **Retrieval 1 (Statute):** *Several Provisions of the Supreme People's Court on Publishing Information of the List of Dishonest Judgment Debtors, Article 1.*
Thinking Step 2: According to Article 1, "having the capability to fulfill but refusing to fulfill" is a core mandatory prerequisite. I need to retrieve further regulatory interpretations on how the court determines "the capability to fulfill."
↪ **Retrieval 2 (Statute):** *Several Provisions of the Supreme People's Court on Publishing Information of the List of Dishonest Judgment Debtors, Article 2.*
Thinking Step 3: I must now locate procedural guidelines or exemption clauses, particularly regarding situations where execution remains incomplete after the court-ordered deadline.
↪ **Retrieval 3 (Statute):** *Several Provisions of the Supreme People's Court on Publishing Information of the List of Dishonest Judgment Debtors, Article 3.*
Thinking Step 4: To validate the interactive boundaries of "non-fulfillment" and "blacklisting" in real judicial practice, I will invoke a precedent retrieval to observe how local courts adjudicate analogous scenarios.
↪ **Retrieval 4 (Precedent Case):** *The judgment debtor Xiang Mo failed to fulfill the repayment obligation after the court judgment took effect, and still willfully refused execution after receiving the enforcement notice...*

[Final Response Generated by Judge Agent]
Answer: No, failure to repay within 25 days does not inevitably lead to being blacklisted. The critical criterion hinges on whether the debtor "has the capability to fulfill but willfully refuses to do so." If the debtor genuinely lacks the capacity to repay (e.g., assets are already sealed by other creditors, or income is insufficient for basic living costs), they may not be blacklisted. Furthermore, according to Article 3 of the *Provisions*, if the debtor provides sufficient collateral, or if their seized/frozen assets are already adequate to liquidate the debt, they shall not be blacklisted. Therefore, the court evaluates this on a case-by-case basis.
Legal Basis: Articles 1 and 3 of the *Several Provisions of the Supreme People's Court on Publishing Information of the List of Dishonest Judgment Debtors*.

Figure 6: An idealized snapshot of Legal-R1's autonomous thought-retrieval loop and final response synthesis.

Normative Evidence Reranking Prompt

Task Instructions
You are an experienced legal document auditor. Based on the provided [Contextual Information] and [Search Query], please screen the [Candidate Texts] and select the document ID(s) that are semantically most relevant and strictly meet the statutory requirements.
Core Rules (Must Be Strictly Obeyed):
1. Context Priority: If the [Contextual Information] specifies a particular law name or article number (e.g., Article 2), the selected text must match it. If no matching text is found, return [] directly.
2. No Padding: The screening result should contain at most {topk} segments. If only 1 segment matches, return only that segment's ID; if none match, return []. It is strictly forbidden to return irrelevant text just to reach the {topk} limit.
3. Output Format: You are only allowed to output a Python-formatted list of integers, for example: [1] or [1, 2].
4. No Explanation: Do not output any analytical process, thinking content (<think>), or any extra text explanations.
5. Index Alignment: Please carefully read the content of each candidate text. If the [Contextual Information] requests 'Article 2', locate the text that actually describes 'Article 2' and output the corresponding [ID] of that text.
Input Data
- [Search Query]: {query}
- [Contextual Information]: {context}
- [Candidate Texts]: {results}
Screening Result (Output List Only):

Figure 7: The prompt template utilized for normative evidence reranking. Within this template, {topk} denotes the predefined upper bound of retrieval turns, {query} and {context} represent the primary retrieval informational query, and {results} contains the candidate document pool obtained from the initial coarse-grained retrieval stage.

Precedent Evidence Summarization Prompt

Task Instructions
You are an experienced judicial assistant. Please carefully compare the user's [Queried Case Facts] with the retrieved [Candidate Case].
Summarize the "facts," "judicial reasoning," and "judgment outcome" of the candidate case into a concise 150-word case brief.
Briefing Requirements:
1. Summarize the facts of the case, emphasizing the similarities between this case and the [Queried Case Facts].
2. Concisely outline the court's judicial reasoning (especially the underlying logic regarding the determination of key circumstances).
3. Clearly state the final judgment outcome.
Input Data
- [Queried Case Facts]: {fact_query}
- [Candidate Case - Facts]: {doc_fact}
- [Candidate Case - Judicial Reasoning]: {doc_reason}
- [Candidate Case - Judgment Outcome]: {doc_result}
Output (Please directly output the 150-word brief):

Figure 8: The prompt template utilized for precedent evidence summarization. Within this template, {fact_query} denotes the target case query to be investigated, while {doc_fact}, {doc_reason}, and {doc_result} represent the factual narrative, the judicial rationale, and the final adjudication outcome of the retrieved candidate precedent, respectively.

Judge Answer Generation Prompt

Task Instructions
You are a professional and rigorous legal AI assistant. Based on the [Original Query] and the system's [Chain-of-Thought Analysis] provided below, please extract and organize the final answer.
Core Rules:
1. Provide a comprehensive summary answer to the original query that is backed by precise evidence, demonstrates clear and profound logical reasoning, reaches a fully correct conclusion, and is articulated with professional phrasing.
2. You must eliminate all internal tags and machine retrieval artifacts, such as <search>, <syllogism>, and <answer>. Seamlessly integrate the core substance of relevant legal statutes and analogous cases into your final response.
3. Present the final result directly to the user. Do not include machine-perspective pleasantries or boilerplate language like "According to the system's analysis."
Input Data
- [Original Query]: {query}
- [Chain-of-Thought Analysis]: {history}
Final Answer:

Figure 9: The prompt template utilized for judge answer generation. Within this template, {query} denotes the initial legal question inputted by the user, and {history} represents the comprehensive reasoning chain and multi-turn retrieval trajectory generated by the lawyer agent.

Case Study: Autonomous Multi-Turn Retrieval and Reasoning Trajectory

[User Query]
My friend borrowed 164,000 RMB this January, failed to repay, and was sued. We won the lawsuit, and the court ordered repayment within 25 days. If they still do not pay, will they automatically be blacklisted as a dishonest judgment debtor?

[Autonomous Reasoning & Retrieval Trajectory]
Thinking Step 1: To address whether non-repayment automatically leads to being blacklisted, I must first locate the precise statutory criteria and procedures regarding the designation of a "dishonest judgment debtor."
↪ **Retrieval 1 (Statute):** *Several Provisions of the Supreme People's Court on Publishing Information of the List of Dishonest Judgment Debtors, Article 1.*
Thinking Step 2: According to Article 1, "having the capability to fulfill but refusing to fulfill" is a core mandatory prerequisite. I need to retrieve further regulatory interpretations on how the court determines "the capability to fulfill."
↪ **Retrieval 2 (Statute):** *Several Provisions of the Supreme People's Court on Publishing Information of the List of Dishonest Judgment Debtors, Article 2.*
Thinking Step 3: I must now locate procedural guidelines or exemption clauses, particularly regarding situations where execution remains incomplete after the court-ordered deadline.
↪ **Retrieval 3 (Statute):** *Several Provisions of the Supreme People's Court on Publishing Information of the List of Dishonest Judgment Debtors, Article 3.*
Thinking Step 4: To validate the interactive boundaries of "non-fulfillment" and "blacklisting" in real judicial practice, I will invoke a precedent retrieval to observe how local courts adjudicate analogous scenarios.
↪ **Retrieval 4 (Precedent Case):** *The judgment debtor Xiang Mo failed to fulfill the repayment obligation after the court judgment took effect, and still willfully refused execution after receiving the enforcement notice...*

[Final Response Generated by Judge Agent]
Answer: No, failure to repay within 25 days does not inevitably lead to being blacklisted. The critical criterion hinges on whether the debtor "has the capability to fulfill but willfully refuses to do so." If the debtor genuinely lacks the capacity to repay (e.g., assets are already sealed by other creditors, or income is insufficient for basic living costs), they may not be blacklisted. Furthermore, according to Article 3 of the *Provisions*, if the debtor provides sufficient collateral, or if their seized/frozen assets are already adequate to liquidate the debt, they shall not be blacklisted. Therefore, the court evaluates this on a case-by-case basis.
Legal Basis: Articles 1 and 3 of the *Several Provisions of the Supreme People's Court on Publishing Information of the List of Dishonest Judgment Debtors*.

Figure 10: An idealized snapshot of Legal-R1's autonomous thought-retrieval loop and final response synthesis.