When Reasoning Beats Scale: A 1.5B Reasoning Model Outranks 13B LLMs as Discriminator

Md Fahim Anjum*

Department of Neurology University of California San Francisco San Francisco, CA 94143 fahim.anjum@ucsf.edu

Abstract

Large Language Models (LLM) with reasoning capabilities offer a promising path for improving candidate evaluation in planning frameworks, but their relative performance against traditional non-reasoning models remains largely underexplored. In this study, we benchmark a distilled 1.5B parameter reasoning model (DeepSeek-R1) against several state-of-the-art non-reasoning LLMs within a generator-discriminator LLM planning framework for the text-to-SQL task. For this, we introduce a novel method for extracting soft scores from the chain-of-thought (CoT) outputs from reasoning that enables fine-grained ranking of candidates. Our central hypothesis is that reasoning models are more effective discriminators than non-reasoning LLMs. Our results show that distilled DeepSeek-R1-1.5B achieves up to 87% higher F1 and 3.7% better discrimination accuracy than CodeLlama-7B, as well as 3.7% higher execution accuracy than CodeLlama-13B, despite having significantly fewer parameters. Furthermore, we find that there is a limit to the logical capabilities of reasoning models, and only providing more context or allowing more compute budget for reasoning is not enough to improve their discrimination performance. Finally, we demonstrate that, unlike non-reasoning LLMs, reasoning models find generation more challenging than discrimination and may underperform as generators compared to smaller non-reasoning LLMs. Our work highlights the potential of reasoning models as discriminators in agentic frameworks, far outweighing their capabilities as generators, offering insights into their optimal role within LLM planning infrastructures.

1 Introduction

Planning is a fundamental part of intelligent decision-making. Significant research efforts have focused on developing approaches to enable effective and human-like planning, including multiphase and multi-agent frameworks [21, 41, 17, 25]. The recent integration of large language models (LLMs) into such systems has spurred significant interest in LLM-based planning frameworks to enhance the multi-step problem-solving abilities using structured planning, which have shown strong performance across a variety of tasks [14, 15, 43, 44, 51, 7, 21, 41]. Despite differences in implementation and design, most of these frameworks follow a three-stage process: (1) searching for potential action sequences, (2) predicting expected outcomes, and (3) selecting the sequence that maximizes the expected utility [25, 16]. Specifically, LLMs first generate candidate actions (generation), evaluate their consequences using themselves, internal simulations, or another model (discrimination), and finally integrate these outcomes for optimal solutions (planning). This structure aligns closely with how LLMs approach multi-step tasks such as mathematical reasoning [35], multi-hop question answering [44], and code generation [42]. Performance of conventional non-reasoning

^{*}Alternate email: dr.fahim.anjum@gmail.com

LLMs in such frameworks has been studied comprehensively [5]. However, recent developments on LLMs designed for inference-time reasoning via internal deliberation show promising performance across various tasks. Augmented with chain-of-thought (CoT) reasoning [34], these models offer a structured approach to complex tasks. Large reasoning models such as OpenAI o-series [19], Gemini 2.0 Flash Thinking [8], DeepSeek-R1 [6], LIMO [45], and Qwen QwQ-32B [23] have shown impressive stepwise reasoning and state-of-the-art (SOTA) performance across complex benchmarks, far surpassing non-reasoning LLMs [40]. Notably, DeepSeek-R1 [6] has achieved SOTA performance on par with OpenAI's models despite low training cost. With logical and accurate outputs, it consistently outperforms Qwen and LLaMA series models, even under quantization and distillation [49, 38]. Despite the success of reasoning models over traditional non-reasoning LLMs, only a handful of studies have explored their integration into planning or multi-agent frameworks [38, 26, 3]. To the best of our knowledge, there has been no systematic investigation into how reasoning models compare to conventional non-reasoning LLMs within the generator-discriminator LLM planning architecture.

In this work, we examine the role of a reasoning model within a generator-discriminator LLM planning agentic framework applied to the text-to-SQL generation task. For this, we use a distilled 1.5B model of DeepSeek-R1 and hypothesize that reasoning models may serve as more effective discriminators and conduct a systematic investigation guided by the following research questions:

- (1) How does incorporating a reasoning LLM as the discriminator influence the accuracy and overall performance of the agentic framework?
- (2) What is the impact of inference-time CoT compute budget and contextual augmentation on the discrimination performance?
- (2) How does a reasoning model perform in the generator role, and is it better at generation than discrimination?

To this end, we analyze and comprehensively evaluate DeepSeek-R1's discrimination as well as generation abilities within our LLM planning framework and compare the performance with several SOTA non-reasoning LLMs. An additional challenge arises in the extraction of soft scores from CoT reasoning outputs. Traditional non-reasoning LLMs can easily provide logit-based scores for discrimination, but in reasoning models, the arbitrarily long and unstructured nature of the CoT trace complicates soft score extraction. To address this, we propose a novel approach for deriving soft scores from the reasoning outputs, enabling more refined discrimination.

Our work evaluates LLMs' ability to discriminate correct and incorrect SQL queries by re-labeling oracle-generated outputs, measuring both classification and ranking. Discriminator performance is tested in a naive setting and an enhanced setting that filters by executability via environment-based validation. We further test whether LLMs can select correct queries from dynamically generated candidates. We analyze DeepSeek-R1's reasoning quality through lexical, structural, and semantic diversity and examine three key factors influencing its discrimination performance: (1) increased test-time compute budget, (2) prompt augmentation with database schema context, and (3) the use of binary decisions vs. logit-based soft scores. Finally, we evaluate LLMs' performance as generators in the full LLM planning pipeline. Our contributions in this work are as follows:

- (1) We provide a systematic comparison of reasoning and non-reasoning LLMs within a generator-discriminator LLM planning framework for the text-to-SQL task.
- (2) We introduce a novel use of reasoning models as discriminators by deriving soft scores from their outputs for more fine-grained candidate ranking.
- (3) We show that DeepSeek-R1 excels at discrimination, outperforming larger non-reasoning models. In particular, we find that a 1.5B distilled DeepSeek-R1 model achieves 87% higher F1 as well as 3.7% better discrimination accuracy than CodeLlama-7B and 3.7% higher execution accuracy than CodeLlama-13B. Furthermore, logit-based soft scoring vs. binary true/false discrimination yields minimal differences (< 1.5%).
- (4) We find that reasoning models have limited logical capacity and adding more context or compute budget yields diminishing returns (e.g., <0.4% gain beyond 1024 tokens), while extremely low budgets severely degrade performance (<2% accuracy, >94% failure).

(5) Finally, we show that while DeepSeek-R1 is a strong discriminator, it underperforms as a generator, even compared to smaller non-reasoning LLMs. We find that, similar to humans, generation is more challenging than discrimination for reasoning models [9], in contrast to non-reasoning LLMs [5, 36, 10, 31].

In summary, our work investigates the role of DeepSeek-R1 within an LLM planning framework for text-to-SQL task. Through this study, we seek to advance our understanding of reasoning-based LLMs and inform future efforts in designing robust, efficient, and scalable autonomous systems.

2 Related Work

In a generator-discriminator LLM planning framework, discrimination is often seen as easier than generation for human and AI agents [9]. However, West et al. hypothesized that generative models such as LLMs may lack discrimination abilities to match their generation capabilities [36]. This aligns with findings from other studies, which suggest that non-reasoning LLMs may misidentify correct self-generated plans as incorrect without external feedback [10, 31]. Huang et al. observed that self-correction performance gains may depend on high-quality external feedback [10] Chen et al. focused on non-reasoning LLMs' discrimination abilities and their impact on several planning methods and found a strong correlation between discrimination accuracy and overall task performance [5].

In the context of test-time compute for reasoning models, Zeng et al. observed an inverse scaling phenomenon where accuracy declines as CoT length increases [47]. They also found that correct solutions are consistently shorter than incorrect ones, suggesting longer CoTs may actually harm performance. Other studies have shown that exceeding a model's inherent reasoning capacity with longer CoTs leads to error accumulation as well as performance degradation [11, 4, 39] and simply increasing inference-time compute budget does not guarantee performance improvement [20, 33]. Moreover, recent studies have found that optimal reasoning length depends on the model's capability as well as on the complexity of the task. [40, 27, 2]

Unlike these prior studies that focus either on non-reasoning models for planning or reasoning models' CoT without any agentic framework, our work investigates the role of a reasoning model within the LLM planning framework and examines the impact of inference-time compute budget on performance.

3 Experimental Setup

3.1 Task and Dataset

We evaluate the text-to-SQL task using the Spider dataset [46]. Following prior work [5], we use a resource-efficient subset of 400 examples from the Spider development set, uniformly sampled across difficulty levels (100 each from easy, medium, hard, and extra-hard categories). The full Spider development set consists of 1,034 examples, with 24% easy (248), 43.1% medium (446), and 32.9% hard or extra hard (340). The random sampling procedure of the examples is described in [5].

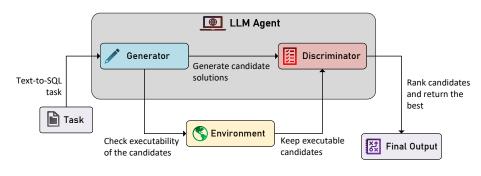
3.2 Language Models

We use *DeepSeek-R1-Distill-Qwen-1.5B* [29] as the primary reasoning model, referred to as 'Distill-R1' throughout this work. To compare against non-reasoning architectures, we select a diverse set of open-source LLMs spanning various sizes: *TinyLlama-1.1B-Chat-v1.0* [48], *DeepSeek-Coder-1.3B-Instruct* [28], *Stable-Code-3B* [22], *CodeLlama-7B-Instruct*, and *CodeLlama-13B-Instruct* [24]. For brevity, we omit the 'Instruct' suffix in model names.

3.3 Framework for LLM Planning

As illustrated in Figure 1, we adopt a generator-discriminator framework [5], where the generator LLM proposes candidate solutions or actions and the discriminator LLM evaluates them. The planning module ranks candidates based on the evaluation outcomes and orchestrates the interaction between the two models. In this section, we detail each of these components and their specific implementation for text-to-SQL parsing.

LLM Agent Framework



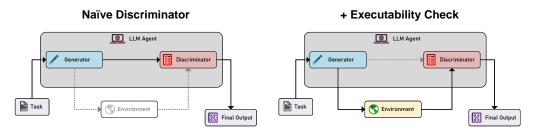


Figure 1: Framework of LLM planning for Text-to-SQL task utilized for this study. For further information, see [5].

3.3.1 Generator

We use the generator LLM to produce candidate SQL queries for user-provided natural language questions via zero-shot prompting. Database tables are included as context, and a fixed prompt template is employed to guide candidate generation. For non-reasoning models, we prompt with the instruction 'Given database schema and a question in natural language, generate the corresponding SQL query.''. For the reasoning model (Distill-R1), the instruction is 'Answer the question by generating a valid sqlite SQL query. Final answer must start with SELECT.'' which allows the model to use reasoning for generating the query. The prompt template in this case is provided in markdown format, facilitating the extraction of the final SQL query generated by the model. Example prompts for both reasoning and non-reasoning models are given in the Appendix (Table 14 and Table 15).

We generate text-to-SQL outputs using the HuggingFace implementation [37], leveraging its model checkpoint and generation function. The hyperparameters are set as follows: $max_length = 300$, temperature = 0.2, and $num_return_sequences = 5$. When a single output is generated without planning or discriminator feedback ($num_return_sequences = 1$), we refer to it as greedy generation.

3.3.2 Discriminator

Given some candidate solutions, we formulate the discrimination task as a binary question answering [12, 13], and generate a discrimination score for each candidate.

Non-reasoning Models: For conventional LLMs, we follow a straightforward scoring approach: the input prompt is passed to the model, and the discrimination score is computed as the token-level probability of generating "Yes" as the next token by applying a softmax function over the final logits [5]. Specifically, we prompt the LLMs with the question "Is the SQL program correct given the utterance?" and extract the softmax probability assigned to "Yes" as the output score. An illustrative example is provided in the Appendix (Table 11).

Reasoning Model: Unlike the non-reasoning models, Distill-R1 utilizes reasoning to obtain a conclusion, and thus obtaining a soft score is not straightforward. For this, we in-

troduce a novel approach: first, we prompt the reasoning model to think and output the final answer in a specific key-value format (JSON). In particular, we prompt the model with the instruction 'Determine whether the SQL query is correct or not for the given question. Provide final answer in JSON format with the key 'correct'.'' An example prompt is provided in the Appendix (Table 12). After the token generation process, the JSON format is identified and parsed to obtain the value for key 'correct' along with its corresponding logit. Then, we normalize the logit via the softmax function to compute the probability ρ for the possible key values ('true' or 'false') and assign the score as ρ for 'true' or $1-\rho$ for 'false'. (Figure 2). If both 'true' and 'false' values are missing, a default score -0.5 is returned.

Reasoning LLM as Discriminator

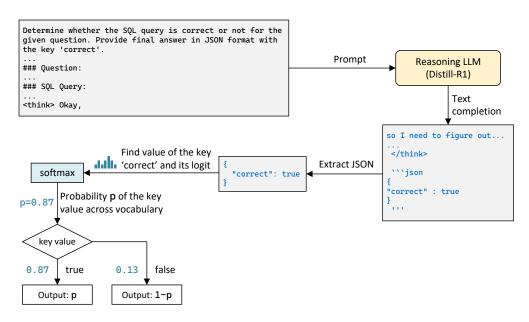


Figure 2: Flowchart illustrating our approach for obtaining soft evaluation scores from a reasoning model (Distill-R1) as a discriminator.

3.3.3 Planning Method

We adopt re-ranking, a simple yet effective planning strategy where the generator first samples a set of candidate queries. These candidates are then individually scored by the discriminator, and the highest-scoring query is returned as the final output. Despite its simplicity, this planning method is widely adopted in code generation [18] and mathematical reasoning tasks [32, 14].

In our experiments, we utilize the discriminator to score the candidates in two configurations [5]:

- 1. *Naive Discriminator*: In this case, the discriminator evaluates the candidates without any additional information.
- 2. *Enhanced with Executability Check*: Here, the candidate SQL queries are tested on an environment to see whether they are executable (run without producing errors). The discriminator only ranks the executable candidates.

3.4 Evaluation Methods

3.4.1 Intrinsic Evaluation

To assess LLMs' discrimination performance, we reuse the generation results from the oracle-simulation experiments in [5]. Here, evaluation scripts were used to re-label the pre-generated SQL queries in simulated re-ranking experiments, where an oracle compares the top five execution results of the predicted and gold SQL queries using table cell overlap. We evaluate the discrimination ability using a range of metrics to capture both binary classification and ranking performance.

3.4.2 End-to-End Evaluation

Unlike intrinsic evaluation, which assesses discrimination ability on a fixed set of queries, our end-toend evaluation measures the effectiveness of LLMs in dynamically generating and ranking candidates within our planning framework. Specifically, we assess the full pipeline performance of our LLM planning framework with LLM-based generators and discriminators under the re-ranking planning strategy and determine their ability to generate and select correct queries in real-time decision-making scenarios. For consistency, we use the same 400 problem instances as in the intrinsic evaluation, enabling one-to-one correspondence across query batches.

3.5 Performance Metrics

3.5.1 Intrinsic Evaluation

To measure the discrimination abilities of LLMs, we employ four intrinsic metrics that evaluate both pairwise judgment and ranking quality [5]. Pairwise Discrimination Accuracy (**PwAcc**) measures the percentage of times the model assigns a higher score to a correct query over an incorrect one, given a pair of queries [1, 30]. Classification Macro F1 (**F1**) treats "correct" and "incorrect" as separate classes and computes the macro-averaged F1 score across them. Hit@1 (H@1) evaluates how often the top-ranked query from a candidate set is correct. Finally, mean Reciprocal Rank (**MRR**) computes the average reciprocal rank of the highest-scoring candidate in each batch.

3.5.2 End-to-End Evaluation

We evaluate the end-to-end performance of the models in the LLM-based planning framework using both execution and match-based metrics. This includes execution accuracy for text-to-SQL generation, exact match accuracy to the reference gold query, and partial match performance measured by accuracy, recall, and F1 score. We utilize the semantic evaluation test suite [50] for the metric calculation.

3.5.3 Reasoning Quality

To measure the richness and non-redundancy of the model's reasoning output, we assess its reasoning quality using a set of diversity and repetition-based metrics: **Repetition Ratio** quantifies the proportion of repeated sentences, **1—TTR** (One minus Type-Token Ratio) calculates lexical simplicity by capturing the proportion of repeated sentences relative to the total, with higher values indicating greater redundancy and simpler language, **Repeated n-grams** captures the frequency of recurring n-grams (e.g., 3-grams) by calculating the repetition ratio of phrases, **Entropy** reflects the distribution of sentence occurrences, with higher values signifying richer output. Finally, **Diversity** measures the overall variety in the generated content by utilizing sentence embeddings and cosine similarity to assess the average dissimilarity between sentence pairs. Higher diversity corresponds to lower average similarity.

4 Results

4.1 Reasoning Model as Discriminator

We investigate how our small reasoning model (Distill-R1) performs as an evaluator in our agentic LLM planning framework compared to other non-reasoning models.

4.1.1 Distill-R1 Outperforms Larger Non-Reasoning LLMs

Intrinsic Evaluation. Our intrinsic evaluation reveals several key takeaways. First, Distill-R1 achieves the highest classification F1 scores under both naive and executability-check settings, outperforming the non-reasoning models by approximately 58% and 87%, respectively (Table 1). Second, in terms of Hit@1 and MRR, Distill-R1 exceeds all models except CodeLlama-13B. In particular, compared to larger non-reasoning LLMs such as StableCode-3B and CodeLlama-7B, Distill-R1 demonstrates improvements of roughly 4.6% in Hit@1 and 2.1 2.6% in MRR under the naive setting, with gains of around 3% in Hit@1 and 1.4 1.9% in MRR when executability checks are applied (Table 1). Finally, for pairwise discrimination accuracy, Distill-R1 remains

highly competitive, with its performance surpassed only by StableCode-3B and CodeLlama-13B. In the naive setting, its accuracy is within 1% of StableCode-3B and approximately 12% lower than CodeLlama-13B. With executability checks, it is less than 0.5% lower than StableCode-3B and 1.8% lower than CodeLlama-13B. These findings show that, while CodeLlama-13B may lead in overall accuracy, the reasoning capabilities of Distill-R1 yield substantial improvements as a discriminator, surpassing larger non-reasoning LLMs.

Models	N	aive Disc	riminato	r	+ Executability Check				
	PwAcc	F1	H@1	MRR	PwAcc	F1	H@1	MRR	
TinyLlama-1.1B	49.39	37.09	54.75	61.87	77.75	37.09	62.00	66.31	
Deepseek Coder-1.3B	50.86	37.09	54.50	61.42	76.28	37.09	61.75	66.13	
StableCode-3B	51.59	37.09	54.00	61.54	78.73	37.09	61.25	66.15	
CodeLlama-7B	49.88	37.09	54.00	61.26	75.79	37.09	61.50	65.86	
Distill R1-1.5B	51.34	58.70	56.50	62.83	78.48	69.29	63.25	67.09	
CodeLlama-13B	58.44	37.09	58.50	64.07	79.95	37.09	64.75	67.77	

Table 1: Performance comparison of LLMs as discriminators during intrinsic evaluation. **Best performance** is in bold. The reasoning model is in green.

End-To-End evaluation. In the end-to-end setting, Distill-R1 demonstrates superior overall execution accuracy in the naive discriminator mode, outperforming CodeLlama-13B by 3.7% and achieving the highest accuracy across all problem categories except the extra hard set (Table 2). When executability checks are applied, Distill-R1 maintains the top overall accuracy (Figure 3), and notably, its performance on extra hard problems improves 18.2% relative to CodeLlama-13B (Table 3). These results indicate that incorporating executability checks can significantly benefit reasoning models on more challenging queries. Meanwhile, the exact match and partial match metrics present mixed outcomes, with Distill-R1 consistently trailing CodeLlama-13B by 10~12%.

Models		Execu	tion Ac	ccuracy		Exact Match	Partial Match		
1,10,000	Easy	Mid	Hard	Extra	All	Acc	Acc	Recall	F1
TinyLlama-1.1B	80	66	46	21	53.2	30.00	66.83	43.69	50.96
Deepseek Coder-1.3B	81	65	46	18	52.5	29.20	72.30	42.71	50.73
Stable Code-3B	80	70	49	16	53.7	28.00	66.27	40.81	48.94
CodeLlama-7B	80	64	44	19	51.7	28.20	65.67	41.19	49.23
CodeLlama-13B	84	70	43	19	54.0	33.00	70.67	45.39	52.98
Distill R1-1.5B	85	71	50	18	56.0	29.50	65.51	43.06	50.57

Table 2: Execution accuracy and gold SQL matching performance of LLMs as discriminators in end-to-end evaluation (Naive Discriminator). **Best performance** is in bold. The reasoning model (Distill-R1) is in green.

4.1.2 Increasing Test-Time Compute Yields Diminishing Returns

Given the strong performance of Distill-R1 as a discriminator in both intrinsic and end-to-end evaluations, we further investigate the impact of test-time compute budget on its discrimination capabilities. Specifically, we explore whether allocating additional computation for reasoning enhances its performance. Experiments are conducted using Distill-R1 as the discriminator under varying token budgets.

Our results show that increasing test-time compute (via the maximum new token limit) significantly enhances discrimination performance initially, but levels off after reaching a certain threshold. At very low token limits (e.g., 256 tokens), the reasoning process is heavily constrained, which causes drastically low pairwise discrimination accuracy (< 2%) and a high failure rate (94 + %; Figure 4; Appendix: Table 8). As the token limit increases, the pairwise accuracy improves gradually and peaks around 1024 tokens. Beyond this point, additional tokens provide only marginal gains, with accuracy increasing by less than 0.4% from 1024 to 2048 tokens. Similar trends are observed in other

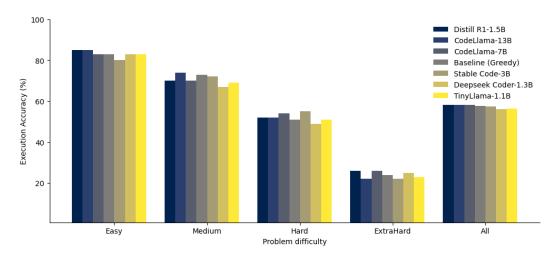


Figure 3: Performance comparison of LLMs as discriminators in terms of execution accuracy during end-to-end evaluation (with Executability Check).

Models		Execu	tion Ac	ccuracy		Exact match	Pa	Partial match		
	Easy	Mid	Hard	Extra	All	Acc	Acc	Recall	F1	
TinyLlama-1.1B	83	69	51	23	56.5	31.20	65.95	46.56	53.00	
Deepseek Coder-1.3B	83	67	49	25	56	29.70	68.72	44.84	52.16	
Stable Code-3B	80	72	55	22	57.3	28.70	63.94	42.37	49.71	
CodeLlama-7B	83	70	54	26	58.3	30.20	65.80	44.43	51.69	
CodeLlama-13B	85	74	52	22	58.3	34.50	67.50	47.08	53.68	
Distill R1-1.5B	85	70	52	26	58.3	30.20	63.73	45.45	52.09	

Table 3: Execution accuracy and gold SQL matching performance of LLMs as discriminators in end-to-end evaluation (+ Executability Check). **Best performance** is in bold. The reasoning model (Distill-R1) is in green.

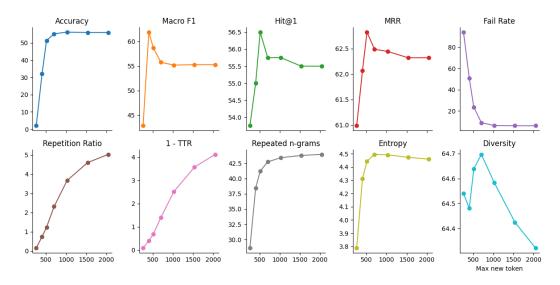


Figure 4: Discrimination performance (*top row*) and reasoning quality (*bottom row*) of the reasoning model (Distill-R1) as a function of test-time compute budget (token limit).

metrics, such as F1, Hit@1, and MRR, while the failure rate decreases sharply with more tokens before stabilizing.

In terms of reasoning quality, our results suggest diminishing returns with increased test-time compute. As the maximum token limit increases, 1—TTR, repetition ratio, and repeated n-grams all rise (Figure 4; Appendix: Table 8), indicating that the model is producing more lexically redundant and repetitive outputs, rather than generating novel content. This suggests that beyond a certain point, additional compute budget leads to more verbose but not necessarily more insightful reasoning. Furthermore, entropy and diversity metrics exhibit peaks between 512 and 1024 tokens and then plateau or slightly decline (Appendix: Table 8). These indicate that the model's reasoning variety and information richness improve only up to a moderate compute budget, after which the quality and variety of reasoning stop improving, with longer outputs becoming more repetitive rather than more insightful. Taken together, these results show that while a minimal token threshold is necessary to enable coherent reasoning, simply increasing token limits beyond this threshold leads to diminishing improvements in both discrimination performance and reasoning quality.

4.1.3 Additional Context Does Not Guarantee Performance Gains

Next, we explore whether adding more context to the prompt helps the reasoning process and improves the discrimination performance of Distill-R1. For this, we add the corresponding database tables to the prompt as context (Appendix: Table 13) and compare the discrimination performance with the case where we don't add schema tables as context (Appendix: Table 12). We set the maximum token limit to 1024 for both cases.

Prompt	Di	iscrimir	nator Pe	rforma	nce	Reasoning Quality					
Condition	PwAcc	Acc F1 H		MRR	Fail Rate	Rep. Ratio	^ 1-11K		Entropy	Diversity	
w/o Schema	56.23	55.20	55.75	62.45	6.10	3.67	2.52	43.40	4.49	64.58	
with Schema	53.79	53.90	56.25	62.56	19.24	7.07	4.94	48.73	4.68	64.14	
Diff. (%)	-4.34	-2.36	+0.90	+0.18	+215.41	+92.64	+96.03	+12.28	+4.23	-0.68	

Table 4: Discrimination and reasoning performance under different prompt conditions for Distill-R1 (Naive Discriminator setting).

Our intrinsic evaluation results indicate that providing schema information as context has little to no benefit. Compared to the baseline without schema, the overall discrimination accuracy and F1 score drop by approximately 4.3% and 2.4%, respectively, while improvements in Hit@1 (0.9%) and MRR (0.18%) are negligible (Table 4). Notably, the failure rate increases dramatically by over 215%, suggesting that the additional context may actually hinder the model's ability to evaluate queries. In terms of reasoning quality, repetition ratio and repeated n-grams increase drastically by 90 + % while 1-TTR shows a 12% rise with schema information added to the prompt. These indicate a higher degree of repetition and lexical redundancy in the reasoning. The changes in entropy and diversity are minimal. These findings suggest that augmenting the prompt with the database schema does not yield significant improvements in discrimination performance and may introduce more redundancy and instability in the reasoning process.

4.1.4 Minimal Effect of Logits on Ranking Performance

Finally, we assess the impact of using logits for ranking on the discrimination performance of the reasoning model. For this, we calculate the final discrimination score using the reasoning text (binary score based on the true/false decision) instead of the logit-based soft discrimination score (Figure 2).

In intrinsic evaluation, switching from a logit-based soft score to a binary score (without logits) results in a substantial drop in overall pairwise discrimination accuracy (32.8% lower in naive setting and 11.5% lower with executability checks; Table 5) while F1, Hit@1, and MRR metrics remained largely unchanged. In contrast, the end-to-end evaluation reveals only a marginal reduction in overall execution accuracy (<1.5%; Table 6). These findings indicate that the final discrimination scoring method (with or without logits) has little impact on overall task performance in practical LLM planning scenarios.

Condition	N	aive Disc	riminato	r	+ Executability Check					
	PwAcc	F1	H@1	MRR	PwAcc	F1	H@1	MRR		
with Logits	51.34	58.70	56.50	62.83	78.48	69.29	63.25	67.09		
w/o Logits	34.47	58.96	57.50	63.10	69.44	69.48	63.75	67.16		
Diff. (%)	-32.84	+0.44	+1.77	+0.43	-11.53	+0.27	+0.79	+0.10		

Table 5: Performance of intrinsic evaluation with and w/o logits for Distill-R1.

Condition		Naive	Discrin	ninator		+ Executability Check					
0011411011	Easy	Mid	Hard	Extra	All	Easy	Mid	Hard	Extra	All	
with Logits	85.00	71.00	50.00	18.00	56.00	85.00	70.00	52.00	26.00	58.30	
w/o Logits	86.00	67.00	50.00	18.00	55.20	86.00	70.00	51.00	25.00	58.00	
Diff (%)	+1.18	-5.63	0.00	0.00	-1.43	+1.18	0.00	-1.92	-3.85	-0.51	

Table 6: Execution accuracy in end-to-end evaluation with and w/o logits for Distill-R1.

4.2 Reasoning Model as Generator

Our results show that the reasoning model, Distill-R1, is not an effective generator. In particular, with Distill-R1 as the baseline, the smallest non-reasoning model (TinyLlama-1.1B) shows improvements of approximately 56.9% in overall execution accuracy, 87.5% in exact match accuracy, and 49.4% in partial match F1 score (Table 7 and Figure 5).

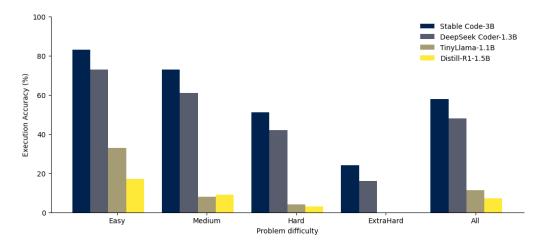


Figure 5: Performance comparison of LLMs as generators in terms of execution accuracy.

Models		Execu	tion Ac	curacy		Exact match	Pa	rtial ma	tch
11204015	Easy	Mid	Hard	Extra	All	Acc	Acc	Recall	F1
Distill-R1-1.5B	17.00	9.00	3.00	0.00	7.20	4.00	53.87	14.23	27.22
TinyLlama-1.1B	33.00	8.00	4.00	0.00	11.30	7.50	36.28	17.04	40.66
DeepSeek Coder-1.3B	73.00	61.00	42.00	16.00	48.00	19.80	66.11	29.29	37.81
Stable Code-3B	83.00	73.00	51.00	24.00	57.80	33.80	69.50	44.70	52.36

Table 7: Execution accuracy and gold SQL matching performance in end-to-end evaluation as a generator (Greedy). **Best performance** is in bold.

Moreover, performance among non-reasoning models increases with model size. A mid-sized model yields gains of roughly 566.7% in execution accuracy, 395% in exact match accuracy, and 38.9% in partial match F1 score relative to Distill-R1. The largest model further enhances these metrics with improvements of 702.8%, 745%, and 92.3% in execution accuracy, exact match accuracy, and partial match F1 score, respectively. These findings underscore two key points: first, that the reasoning model is outperformed by even smaller non-reasoning models, and second, that scaling non-reasoning architectures leads to progressively higher performance gains in SQL generation tasks.

5 Discussion

In this work, we study the role of a reasoning model within the generator-discriminator LLM planning framework[5]. In particular, we use the re-ranking planning method and benchmark a reasoning model (Distill-R1-1.5B) against several non-reasoning LLMs in the text-to-SQL task. Our investigation reveals several key insights into the roles of reasoning models in LLM planning frameworks, with significant implications for both the design and deployment of such models in an agentic framework.

5.1 Distill-R1 Outperforms Larger Non-Reasoning LLMs as Discriminator

First, our findings show that the reasoning model excels as a discriminator not only in distinguishing correct from incorrect queries but also by achieving the highest overall execution accuracy in the LLM planning framework, outperforming larger non-reasoning models in both cases. During our intrinsic evaluation, Distill-R1 shows strong capabilities in discriminating between correct and incorrect queries more effectively than its non-reasoning counterparts. In particular, compared to CodeLlama-7B, Distill-R1 demonstrates improvements of 2.9% in pairwise accuracy, 58% in F1, 4.6% in Hit@1, and 2.6% in MRR under the naive setting, which become 3.5%, 87%, 2.8%, and 1.9%respectively, when executability checks are applied (Table 1). In terms of overall execution accuracy during end-to-end evaluation of our LLM planning framework, Distill-R1 with only 1.5B parameters outperformed non-reasoning models that are more than 8 times larger (3.7\% improvement compared to CodeLlama-13B; Table 2). These findings suggest that reasoning models are particularly effective in evaluating candidate queries, even when a smaller model is used. Thus, by leveraging their inherent reasoning abilities, such models can discern subtle differences between candidates more effectively than non-reasoning models. Moreover, the performance difference between logit-based soft scoring and binary true/false discrimination is minimal (< 1.5%) for Distill-R1, showing that reasoning models are good discriminators in the LLM planning framework even when only binary outcomes are available. This can provide a significant advantage in commercial settings where access to logit values for reasoning models may be restricted or expensive. Together, these results suggest that reasoning models hold substantial promise for robust discrimination tasks, and further research could drive even greater performance improvements.

5.2 Inherent Limits of Reasoning Capabilities

Second, our findings demonstrate that for reasoning models, there is a limit to the benefits of increased test-time compute for evaluating candidates. A very low token limit severely restricts the reasoning process, leading to extremely low accuracy and a very high failure rate, indicating the necessity of a minimum compute threshold. As the token budget increases, performance improves and peaks around 1024 tokens, beyond which the gains become negligible. These show that allocating more test-time compute budget does not directly translate into more insightful reasoning. We also observe that increasing the token budget leads to more redundancy and repetition in reasoning outputs (Figure 4; Appendix: Table 8), indicating the occurrence of circular reasoning and loss of context. Furthermore, we find that incorporating database schema information into the prompt does not offer significant improvements for Distill-R1 (Table 4). In fact, it can introduce redundancy and even elevate failure rates. Our findings are consistent with earlier studies, which also showed a minimum computing budget, a diminishing return after a certain threshold, and an inverse relationship between CoT length and accuracy [47, 39]. Together, these results indicate that there is a limit to the logical capabilities of these reasoning models, and providing more context or allowing more compute budget for reasoning is not sufficient for improving their performance. Future studies should focus not only on scaling compute and maximizing context window but also on targeted fine-tuning of such reasoning models.

5.3 Generation is Harder than Discrimination

Third, while Distill-R1 shows strong performance as a discriminator, it performs poorly as a generator. In fact, for generating high-quality SQL queries, it is outperformed by smaller non-reasoning models (Table 7 and Figure 5). This suggests that while reasoning models are effective at evaluating candidate queries, they may not be well-suited for generating them. This result is not surprising, as SQL generation from natural language involves complex reasoning that can branch into convoluted CoT. With our earlier results indicating a fundamental limitation in the reasoning capabilities, navigating through a complex CoT is challenging for reasoning models. Hence, while previous studies have shown that for non-reasoning LLMs, the discriminator role is typically more challenging and significant than the generator role [5, 36, 10, 31], our results indicate the opposite for reasoning models, which is similar to humans [9].

5.4 Future Works and Limitations

In this study, we demonstrate that reasoning models can be employed to generate soft scores for evaluating candidates, which is a non-trivial task given that CoT reasoning may produce arbitrarily long token outputs, complicating soft score extraction. By leveraging our approach, it is potentially possible to fine-tune such models (with soft score as the cost function) and adjust their reasoning approaches to improve performance even further. We plan to explore this in future studies.

Our study has several limitations. The experiments are conducted within the specific context of text-to-SQL generation and discrimination, which may not directly generalize to other domains or reasoning tasks. Additionally, while our analysis provides insights into the trade-offs between compute budget and performance, further research is necessary to fully understand the interplay between model architecture, token limits, and task complexity. Lastly, we use a relatively small reasoning model (Distill-R1, 1.5B parameters) and a reduced dataset due to our limited budget for computational resources. However, our goal in this work is to investigate and gain a deeper understanding of reasoning models within LLM planning frameworks. In this aspect, we believe our findings will provide significant insights even for larger reasoning models.

6 Conclusion

In this work, we explore the role of a reasoning model, Distilled DeepSeek-R1-1.5B, within an LLM planning framework for text-to-SQL task. We introduce a novel approach to extract soft scores from its CoT outputs for discrimination and systematically evaluate its capabilities as a discriminator well as a generator. We show that DeepSeek-R1 outperforms much larger non-reasoning models in query discrimination, highlighting the potential of such models as discriminators within LLM planning frameworks. Our study also reveals that reasoning models face inherent limitations. In particular, increasing test-time compute and additional context yield diminishing returns. Finally, while DeepSeek-R1 excels at evaluating candidate queries, it struggles as a generator, where even smaller non-reasoning models deliver higher-quality SQL outputs. Though our experiments are constrained in scope and scale, these insights can point to promising directions for developing smaller, more efficient agentic systems that strategically integrate reasoning models into planning frameworks.

Data and Code Availability

All code, datasets, and detailed instructions necessary to reproduce our results are publicly available at: https://github.com/MDFahimAnjum/llm-planning-with-reasoning.

Computing Hardware

All experiments are conducted on commercially available, consumer-grade laptops equipped with NVIDIA GeForce RTX 4080 (12GB) and RTX 3070 (8GB) GPUs.

References

[1] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath,

- Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022.
- [2] Vidhisha Balachandran, Jingya Chen, Lingjiao Chen, Shivam Garg, Neel Joshi, Yash Lara, John Langford, Besmira Nushi, Vibhav Vineet, Yue Wu, and Safoora Yousefi. Inference-time scaling for complex tasks: Where we stand and what lies ahead, 2025.
- [3] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models, 2025.
- [4] Qiguang Chen, Libo Qin, Jiaqi WANG, Jingxuan Zhou, and Wanxiang Che. Unlocking the capabilities of thought: A reasoning boundary framework to quantify and optimize chain-of-thought. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024
- [5] Ziru Chen, Michael White, Raymond Mooney, Ali Payani, Yu Su, and Huan Sun. When is tree search useful for llm planning? it depends on the discriminator, 2024.
- [6] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
- [7] Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training, 2024.
- [8] Google. Gemini 2.0 flash thinking experimental. https://deepmind.google/technologies/gemini/flash-thinking/, December 2024. Accessed: April 2025.
- [9] Yu Gu, Xiang Deng, and Yu Su. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4928–4949, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [10] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024.
- [11] Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. The impact of reasoning step length on large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1830–1842, Bangkok, Thailand, August 2024. Association for Computational Linguistics.
- [12] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022.
- [13] Pei Ke, Fei Huang, Fei Mi, Yasheng Wang, Qun Liu, Xiaoyan Zhu, and Minlie Huang. DecompEval: Evaluating generated texts as unsupervised decomposed question answering. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9676–9691, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [14] Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making language models better reasoners with step-aware verifier. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada, July 2023. Association for Computational Linguistics.

- [15] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [16] Marcelo G. Mattar and Máté Lengyel. Planning in the brain. Neuron, 110(6):914–934, 2022.
- [17] A. Newell and H. Simon. The logic theory machine–a complex information processing system. *IRE Transactions on Information Theory*, 2(3):61–79, 1956.
- [18] Ansong Ni, Srini Iyer, Dragomir Radev, Veselin Stoyanov, Wen-Tau Yih, Sida Wang, and Xi Victoria Lin. LEVER: Learning to verify language-to-code generation with execution. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 26106–26128. PMLR, 23–29 Jul 2023.
- [19] OpenAI. Openai o1 system card, 2024.
- [20] Shubham Parashar, Blake Olson, Sambhav Khurana, Eric Li, Hongyi Ling, James Caverlee, and Shuiwang Ji. Inference-time computations for llm reasoning and planning: A benchmark and insights, 2025.
- [21] Mihir Parmar, Xin Liu, Palash Goyal, Yanfei Chen, Long Le, Swaroop Mishra, Hossein Mobahi, Jindong Gu, Zifeng Wang, Hootan Nakhost, Chitta Baral, Chen-Yu Lee, Tomas Pfister, and Hamid Palangi. Plangen: A multi-agent framework for generating planning and reasoning trajectories for complex problem solving, 2025.
- [22] Nikhil Pinnaparaju, Reshinth Adithyan, Duy Phung, Jonathan Tow, James Baicoianu, and Nathan Cooper. Stable code 3b.
- [23] Qwen Team. QwQ-32B: Embracing the Power of Reinforcement Learning. https://qwenlm.github.io/blog/qwq-32b/, March 2025. Accessed: April 2025.
- [24] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2024.
- [25] Stuart Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall, 3 edition, 2010.
- [26] Swarnadeep Saha, Xian Li, Marjan Ghazvininejad, Jason Weston, and Tianlu Wang. Learning to plan & reason for evaluation with thinking-llm-as-a-judge, 2025.
- [27] Amrith Setlur, Nived Rajaraman, Sergey Levine, and Aviral Kumar. Scaling test-time compute without verification or rl is suboptimal, 2025.
- [28] DeepSeek AI Team. Deepseek-coder-1.3b-instruct, 2024. Fine-tuned on 2 billion tokens of instruction data, designed for advanced code generation and infilling tasks with a context length of 16K tokens.
- [29] DeepSeek AI Team. Deepseek-r1-distill-qwen-1.5b: Compact reasoning model for mathematical and coding tasks, 2025.
- [30] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien

- Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.
- [31] Boshi Wang, Xiang Yue, and Huan Sun. Can ChatGPT defend its belief in truth? evaluating LLM reasoning via debate. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11865–11881, Singapore, December 2023. Association for Computational Linguistics.
- [32] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.
- [33] Yu Wang, Nan Yang, Liang Wang, and Furu Wei. Examining false positives under inference scaling for mathematical reasoning, 2025.
- [34] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.
- [35] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.
- [36] Peter West, Ximing Lu, Nouha Dziri, Faeze Brahman, Linjie Li, Jena D. Hwang, Liwei Jiang, Jillian Fisher, Abhilasha Ravichander, Khyathi Chandu, Benjamin Newman, Pang Wei Koh, Allyson Ettinger, and Yejin Choi. The generative AI paradox: "what it can create, it may not understand". In *The Twelfth International Conference on Learning Representations*, 2024.
- [37] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.
- [38] Junde Wu, Jiayuan Zhu, and Yuyuan Liu. Agentic reasoning: Reasoning llms with tools for the deep research, 2025.
- [39] Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Understanding chain-of-thought length in llms, 2025.
- [40] Zixuan Wu, Francesca Lucchetti, Aleksander Boruch-Gruszecki, Jingmiao Zhao, Carolyn Jane Anderson, Joydeep Biswas, Federico Cassano, Molly Q Feldman, and Arjun Guha. Phd knowledge not required: A reasoning challenge for large language models, 2025.
- [41] Chengxing Xie and Difan Zou. A human-like reasoning framework for multi-phases planning task with large language models, 2024.
- [42] John Yang, Akshara Prabhakar, Karthik R Narasimhan, and Shunyu Yao. Intercode: Standardizing and benchmarking interactive coding with execution feedback. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.
- [43] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [44] 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*, 2023.
- [45] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning, 2025.
- [46] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of*

- the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3911–3921, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.
- [47] Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. Revisiting the test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities?, 2025.
- [48] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model. *ArXiv*, abs/2401.02385, 2024.
- [49] Kaikai Zhao, Zhaoxiang Liu, Xuejiao Lei, Jiaojiao Zhao, Zhenhong Long, Zipeng Wang, Ning Wang, Meijuan An, Qingliang Meng, Peijun Yang, Minjie Hua, Chaoyang Ma, Wen Liu, Kai Wang, and Shiguo Lian. Quantifying the capability boundary of deepseek models: An application-driven performance analysis, 2025.
- [50] Ruiqi Zhong, Tao Yu, and Dan Klein. Semantic evaluation for text-to-sql with distilled test suite. In *The 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2020.
- [51] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models, 2023.

A Appendix

A.1 Additional Tables

Max New	Dis	crimina	tor Per	forman	ce	Reasoning Quality						
Token	PwAcc	F1	H@1	MRR	Fail Rate	Rep. Ratio	1-TTR	Rep. n-grams	Entropy	Diversity		
256	1.96	42.87	53.75	60.98	94.30	0.15	0.09	28.62	3.79	64.54		
400	32.03	61.89	55.00	62.07	50.77	0.74	0.40	38.42	4.31	64.48		
512	51.34	58.70	56.50	62.83	23.57	1.23	0.69	41.20	4.45	64.64		
700	55.26	55.75	55.75	62.49	8.86	2.31	1.40	42.72	4.50	64.70		
1024	56.23	55.20	55.75	62.45	6.10	3.67	2.52	43.40	4.49	64.58		
1536	55.99	55.28	55.50	62.32	5.92	4.60	3.57	43.78	4.47	64.42		
2048	55.99	55.28	55.50	62.32	5.92	5.03	4.12	43.94	4.46	64.32		

Table 8: Performance metrics for different token budgets in intrinsic evaluation with Naive Discrimination setting using Distill-R1.

Models		Execu	tion Ac	curacy		Exact match	Pa	rtial ma	tch
	Easy	Mid	Hard	Extra	All	Acc	Acc	Recall	F1
with Logits	85.00	70.00	52.00	26.00	58.30	30.20	63.73	45.45	52.09
w/o Logits	86.00	70.00	51.00	25.00	58.00	31.80	64.83	45.34	52.25
Diff (%)	+1.18	0.00	-1.92	-3.85	-0.51	+5.30	+1.73	-0.24	+0.31

Table 9: Execution accuracy and gold SQL matching performance in end-to-end evaluation with and w/o logits (+ Executability Check).

Models		Execu	tion Ac	curacy		Exact match	Pa	rtial ma	tch
	Easy	Mid	Hard	Extra	All	Acc	Acc	Recall	F1
with Logits	85.00	71.00	50.00	18.00	56.00	29.50	65.51	43.06	50.57
w/o Logits	86.00	67.00	50.00	18.00	55.20	30.20	67.16	43.04	50.96
Diff (%)	+1.18	-5.63	0.00	0.00	-1.43	+2.37	+2.52	-0.05	+0.77

Table 10: Execution accuracy and gold SQL matching performance in end-to-end evaluation with and w/o logits (Naive Discriminator).

A.2 Example Prompts and Responses

```
[INST] Answer the following Yes/No question: Is the SQL correct given the utterance?
- Utterance: What are all distinct countries where singers above age 20 are from?
- SQL:
SELECT DISTINCT country FROM singer WHERE age > 20;
- Answer: [/INST]
```

Table 11: Example prompt for SQL code evaluation with non-reasoning models.

```
Determine whether the SQL query is correct or not for the given question. Provide final answer in
{\tt JSON} format with the key 'correct'. For this task do the following:
- First, break down the query step by step
- Then, determine if the query provides the answer to the given question.
You only need to determine in terms of correctness, if query is correct or not. Do not generate
alternative query.
### Question:
What are all distinct countries where singers above age 20 are from?
### SQL query:
SELECT DISTINCT country FROM singer WHERE age > 20;
Okay, so I need to figure out if this SQL query is correct for the given question. The question is asking for all distinct countries where singers are above age 20. The SQL query provided is:
SELECT DISTINCT country FROM singer WHERE age > 20;
Let me break this down step by step.
First, the question is about finding countries where singers are above age 20. So, the key here is
to find the country (country) where the age (age) is greater than 20.
Looking at the SQL query, it starts with SELECT DISTINCT country FROM singer WHERE age > 20. So,
it's selecting the country from the singer table where the age is greater than 20.
Wait, but the question is about all distinct countries where singers are above age 20. So, the query
is correctly selecting the country where the age is greater than 20. It doesn't mention anything
about the singers themselves, just the country.
So, the query is correct because it's selecting the country where the age is greater than 20, which
answers the question about all distinct countries where singers are above age 20.
I think that's it. The SQL query is correct.
</think>
"json
"correct" : true
```

Table 12: Example SQL code evaluation with DeepSeek-R1 without using the schema as context. The black text represents the input prompt, while the generated SQL response is highlighted in blue.

```
Determine whether the SQL query is correct or not for the given question. Provide final answer in
JSON format with the key 'correct'.
For this task do the following:
- First, break down the query step by step. Use the Schema for this.
- Then, determine if the query provides the answer to the given question.
You only need to determine in terms of correctness, if query is correct or not. Do not generate
alternative query.
### Schema:
- Table stadium: stadium_id, location, name, capacity, highest, lowest, average
- Table singer: singer_id, name, country, song_name, song_release_year, age, is_male
- Table concert: concert_id, concert_name, theme, stadium_id, year
- Table singer_in_concert: concert_id, singer_id
### Question:
What are all distinct countries where singers above age 20 are from?
### SQL query:
SELECT DISTINCT country FROM singer WHERE age > 20;
Okay, so I need to figure out if the SQL query I wrote is correct for the given question. Let me
break it down step by step.
First, the question is asking for all distinct countries where singers are above age 20. So, I need
to find singers who are older than 20 and then see which countries they come from.
Looking at the provided SQL query:
SELECT DISTINCT country
FROM singer
WHERE age > 20;
Let me analyze each part of this query.
1. **SELECT DISTINCT country**: This part is straightforward. It's asking for the distinct values
of the 'country' column. So, it's selecting all unique countries where the condition is met.
2. **FROM singer**: This tells the SQL query where to look for the data. It's selecting from the
'singer' table. So, it's filtering based on the 'singer' table.
3. **WHERE age > 20**: This is the condition that filters the data. It's selecting rows where the
'age' column is greater than 20.
Putting it all together, the query is selecting distinct countries from the 'singer' table where the
singer's age is greater than 20.
Now, let me think about whether this query correctly answers the question.
The question is asking for all distinct countries where singers are above age 20. The query does
exactly that:
- It selects 'country' from 'singer' where 'age' > 20.
So, the query is correctly structured to answer the question.
I don't see any issues with this query. It's straightforward and meets all the requirements of the
question.
</think>
"json
"correct": true
```

Table 13: Example SQL code evaluation with DeepSeek-R1 using the schema as context. The black text represents the input prompt, while the generated SQL response is highlighted in blue.

```
Answer the question by generating a valid sqlite SQL query. Final answer must start with SELECT.
- Table addresses: address_id, line_1, line_2, line_3, city, zip_postcode, state_province_county,
country, other_address_details
- Table courses: course_id, course_name, course_description, other_details
- Table departments: department_id, department_name, department_description, other_details
- Table degree_programs: degree_program_id, department_id, degree_summary_name,
degree_summary_description, other_details
- Table sections: section_id, course_id, section_name, section_description, other_details
- Table semesters: semester_id, semester_name, semester_description, other_details
- Table students: student_id, current_address_id, permanent_address_id, first_name, middle_name,
last_name, cell_mobile_number, email_address, ssn, date_first_registered, date_left,
other student details
- Table student enrolment: student enrolment id. degree program id. semester id. student id.
other details
- Table student_enrolment_courses: student_course_id, course_id, student_enrolment_id
- Table transcripts: transcript_id, transcript_date, other_details
- Table transcript_contents: student_course_id, transcript_id
### Question: What other details can you tell me about students in reverse alphabetical order?
<t.hink>
Okay, so I need to figure out what other details I can tell about students in reverse alphabetical
order. Let me start by understanding the problem.
First, the question is about students ordered alphabetically in reverse, which means from Z to A.
I need to find other details about these students. The database schema includes several tables:
students, student_enrolment, student_enrolment_courses, etc.
I think the key here is to look at the 'students' table because it contains all the necessary
information about each student. The 'student_enrolment' and 'student_enrolment_courses' tables might
provide additional details, but they seem to be related to how students are enrolled and the courses
they take, which might not be directly relevant to their names.
So, focusing on the 'students' table, I can get the student_id, first_name, last_name, etc. But
the question is about reverse alphabetical order, so I need to sort the students by their names in
reverse and then extract the relevant details.
I should write a SQL query that selects the necessary fields from the 'students' table, orders them in reverse alphabetical order, and then returns the selected fields. The order by clause should be
based on the 'first_name' column, but I need to make sure it's in reverse.
Wait, the 'student_enrolment' table might have more information about each student, like their course
details or other details, but I'm not sure if that's necessary. The 'student_enrolment_courses'
table seems to be about which courses a student is enrolled in, which might not be directly relevant
So, the main data I can get is from the 'students' table. I can sort them by first_name in reverse
and then select the first_name, last_name, etc., to get the details in reverse alphabetical order.
I should also consider if there are any other fields that might be useful, like email_address, ssn,
date_first_registered, etc., but the question is about other details, so I think focusing on the
names is sufficient.
Putting it all together, the SQL query should select the first_name and last_name from the 'students'
table, order them by first_name in reverse, and then return these fields. This should give me the
details in the required order.
I think that's the approach. Now, I'll write the SQL query accordingly.
To determine the details of students in reverse alphabetical order, we can use the following SQL
query:
SELECT first_name, last_name
FROM students
ORDER BY first_name DESC;
This query selects the 'first_name' and 'last_name' from the 'students' table and orders them
in descending alphabetical order. The result will provide the details of students in reverse
alphabetical order.
```

Table 14: Example SQL code generation with DeepSeek-R1. The black text represents the input prompt, while the generated SQL response is highlighted in blue.

```
[INST] Given database schema and a question in natural language, generate the corresponding SQL
- Database student_transcripts_tracking:
- Table addresses: address_id, line_1, line_2, line_3, city, zip_postcode, state_province_county,
country, other_address_details
- Table courses: course_id, course_name, course_description, other_details
- Table departments: department_id, department_name, department_description, other_details
 Table degree_programs: degree_program_id, department_id, degree_summary_name,
degree_summary_description, other_details
- Table sections: section_id, course_id, section_name, section_description, other_details
- Table semesters: semester_id, semester_name, semester_description, other_details
- Table students: student_id, current_address_id, permanent_address_id, first_name, middle_name,
{\tt last\_name,\ cell\_mobile\_number,\ email\_address,\ ssn,\ \bar{date\_first\_registered},\ date\_left,}
other_student_details
- Table student_enrolment: student_enrolment_id, degree_program_id, semester_id, student_id,
other_details
- Table student enrolment courses: student course id. course id. student enrolment id
- Table transcripts: transcript_id, transcript_date, other_details
- Table transcript_contents: student_course_id, transcript_id
- Question: What other details can you tell me about students in reverse alphabetical order?
[/INST] SELECT s.*, a.line_1, a.line_2, a.line_3, a.city, a.zip_postcode, a.state_province_county,
a.country, a.other_address_details, d.department_name, dp.degree_summary_name, sp.semester_name
FROM students s
INNER JOIN addresses a ON s.current_address_id = a.address_id
INNER JOIN departments d ON s.department_id = d.department_id
INNER JOIN degree_programs dp ON s.degree_program_id = dp.degree_program_id INNER JOIN semesters sp ON s.semester_id = sp.semester_id
ORDER BY s.last_name DESC;
[/INST]
- Answer: The students in the student_transcripts_tracking database can be sorted in reverse
alphabetical order by the last name.
```

Table 15: Example SQL code generation with non-reasoning model (DeepSeek Coder-1.3B). The black text represents the input prompt, while the generated SQL response is highlighted in blue.