# The Role of Feedback in Inference-Time AI Agent Alignment

Anonymous ACL submission

## Abstract

Inference-time alignment through scaling test-time compute is a promising approach for improving the performance of AI agents. Such approaches typically involve three key components: *sampling*, *evaluation*, and *feedback*. While the role of sampling and evaluation are well studied in the literature, the role of feedback on inference-time alignment is relatively under-explored. We address this gap by introducing Iterative Agent Decoding (IAD), a general sequential framework that enables the integration of different forms of feedback to improve the performance. We analyze how feedback impacts agent performance across four dimensions: (1) accuracy vs compute - budget controlled scaling (2) impact of adaptive feedback beyond sampling diversity (3) impact of feedback modalities, (4) sensitivity to feedback quality. Our evaluations on Sketch2Code, Text2SQL, Intercode and Webshop demonstrate that feedback plays a crucial role in inference-time alignment, yielding performance gains of up to 10% over strong baselines. Our findings provide a unified understanding of the role of feedback mechanisms in inference-time alignment.

## 1 Introduction

AI agents demonstrate remarkable potential across a wide range of applications, but still face challenges in tasks that require multimodal understanding, strategic planning, and reasoning. For example, in complex agentic tasks such as **Sketch2Code** (Li et al., 2024b), **Text2SQL** (Li et al., 2024a), **Intercode** (Yang et al., 2023) and **Webshop** (Yao et al., 2023), state-of-the-art AI agents can only achieve 20–30% accuracy. Post-training via SFT and RL (Ouyang et al., 2022a,b; Bai et al., 2022) has been an effective tool to enhance the capabilities of generative models; however, it is not directly applicable to AI agents due to their inherently black box nature because of their operation in environments

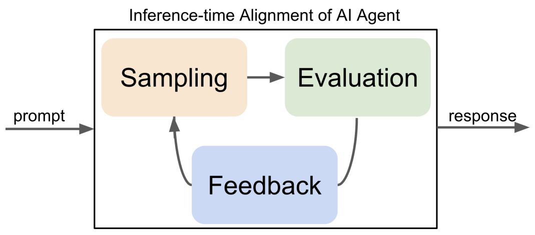


Figure 1: General Pipeline for inference alignment approaches for agents

with no access to the internals (Liu et al., 2024; Fu et al., 2024). Hence, we focus on inference-time approaches which are (i) API-friendly (applicable to closed-source models accessible via commercial APIs) and (ii) compatible with various underlying models.

**Inference-time AI agent alignment:** Broadly, such approaches consist of iteration over three key components: *sampling*, *evaluation*, and *feedback* as illustrated in Figure 1. A typical inference-time iteration involves sampling one or more outcomes from the generative model, evaluating them using a judge/reward, and then generating feedback to improve subsequent outcomes.

The role of *sampling* and *evaluation* is well-studied in literature with efficient and scalable algorithms based on best-of-N (BoN) approaches (Nakano et al., 2021; Beirami et al., 2024). BoN-based approaches operate by sampling multiple responses and selecting the best according to a verifier, effectively leveraging the sampling and evaluation steps. However, a key limitation of BoN approaches is their inability to incorporate feedback mechanisms for iterative refinement.

Hence, recently several sequential approaches have emerged that aim to integrate feedback mechanisms primarily using self-LLMs as judges (Madaan et al., 2023), or leveraging heuristically driven refinement and verification techniques. However, existing works often lack a focused analysis of how feedback should be designed, integrated,

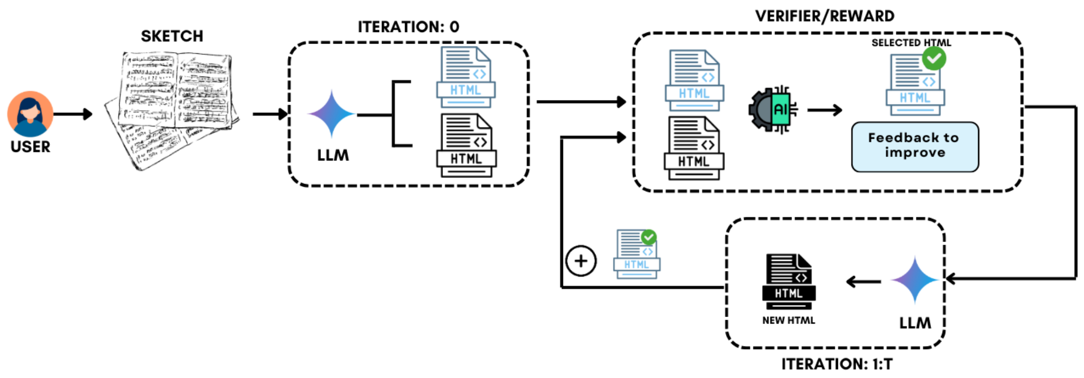


Figure 2: Demonstrates the high-level schematic of - Iterative Decoding of AI Agents - Sequential framework to study the role of feedback for Inference alignment.

or optimized. As a result, the role of feedback in inference-time alignment, especially in black box agentic environments, remains under-examined. We aim to address this gap by focusing on feedback and systematically understanding its impact on inference-time alignment for agents.

**Understanding the role of feedback:** To understand the role and impact of feedback on inference-time alignment, we require a unified framework that flexibly integrates diverse forms of feedback, guided by a verifier, without relying heavily on prompt engineering, which can be costly and fragile. To this end, we introduce Iterative Agent Decoding (IAD), a sequential framework designed to incorporate various forms of feedback to understand the role of feedback in inference-time alignment of agents. Through IAD we uncover several intriguing results, detailed below.

**1. Accuracy vs compute - budget controlled scaling:** We demonstrate that feedback plays a critical role in budget-constrained settings, achieving up to a 10% accuracy gain over strong feedback-free baselines. However, this margin narrows with increased budget—either through higher $N$ or more capable models.

**2. Impact of adaptive feedback beyond sampling gain:** We perform controlled experiments that isolate the effect of sampling, and demonstrate that the gains in IAD arise from adaptive, feedback-guided refinement rather than stochastic sampling alone.

**3. Design and role of feedback form:** We investigate the impact of different forms of feedback on inference alignment. When feedback is textual, it is relatively easy to integrate as is. However, designing effective feedback from scalar rewards and preferences remains underexplored. In IAD, we focus on various methods to extract useful signal from scalar rewards and demonstrate performance gains crucially depend on the design of proper feedback. However, how to map different feedback forms into textual feedback remains an interesting and open research direction.

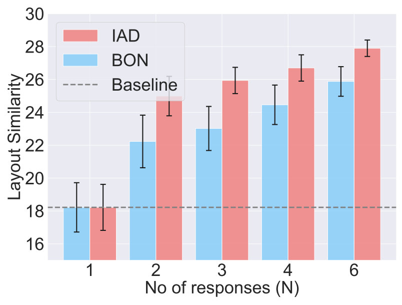
**4. Sensitivity to feedback quality:** We study the sensitivity and robustness of inference alignment to feedback quality via controlled sparsity and noise in feedback signals. We see that IAD remains effective under moderate degradation, but its performance declines with increasing noise, especially when reward is sparse.

## 2 Inference-time Agent Alignment

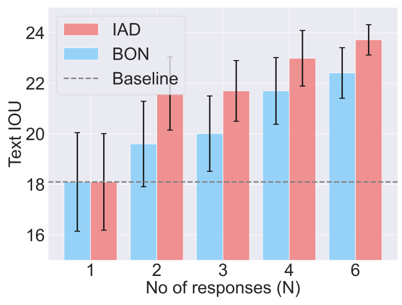
In the black box agentic settings, we only have access to a reference policy $\pi_0(\cdot|x)$, whose internals are inaccesible. The core challenge is to optimize inference-time decisions such that the generated response from $\pi_0(\cdot|x)$ is better aligned with the outputs of optimal policy $\pi^*(\cdot|x)$. We assume access to a verifier function $R(x, y)$, which evaluates the quality of a response $y$ (further details in Appendix). With this formalization, we identify three core components of inference-time alignment for black-box agents: **1. Sampling** : Generate one or more candidate response from the reference policy $y_1, y_2 \cdots y_N \sim \pi_0(\cdot|x)$, where the sampling diversity can be controlled with parameters such as temperature, nucleus sampling etc. **2. Evaluation** : Evaluate each candidate using the verifier $R(x, y_j)$ which serves as a proxy for how well $y_j$ aligns with $\pi^*(\cdot|x)$ and **3. Feedback** : Generate feedback from the verification scores—either scalar or judge-based—to iteratively refine the prompt or guide subsequent generations, enabling adaptive alignment over inference rounds.

### 2.1 Iterative Agent Decoding

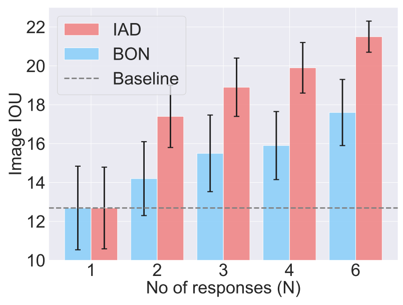
To study the role of feedback, we introduce Iterative Agent Decoding (IAD), a general sequential



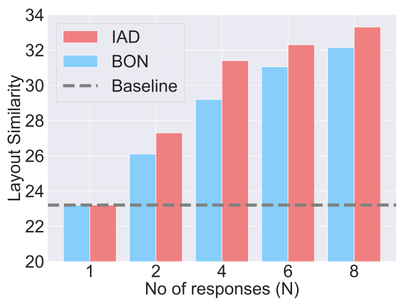
(a)



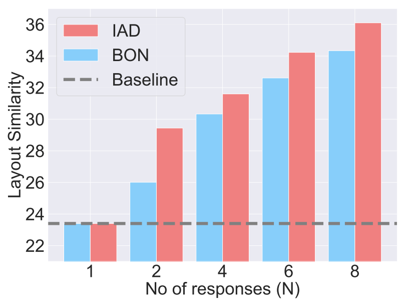
(b)



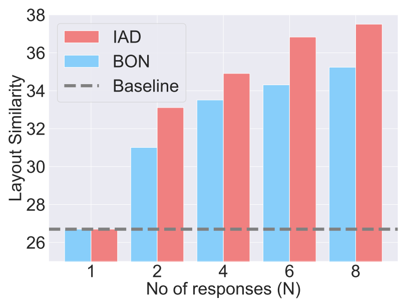
(c)



(d)



(e)



(f)

Figure 3: **Sketch2code**: This figure provides a comparison of IAD (ours) against Best-of-N sampling (SoTA) and single-turn generation with Gemini-1.5-Pro w.r.t metrics - (a) Layout Similarity (b) TextIoU (c) ImageIoU across varying the number of generations (N). Figure demonstrates IAD outperforms BoN consistently across N, but as N increases the gap reduces. Fig(d, e, f) provides a comparison with improved model capabilities from **Gemini-2.0-Flash, Gemini-2.5-Flash, Gemini-2.5-Pro** respectively.

framework designed to incorporate various feedback forms for understanding the role of feedback in inference-time alignment of agents.

**Step 1: Sampling.** At each step $t$, we sample a candidate response with the reference policy $\pi_0(\cdot|x)$ and iteratively refine its outputs.

$$y_{t+1} \sim \pi_0(\cdot|x, \hat{y}_t, s_t, p_t), \tag{1}$$

where $\hat{y}_t$ is the best response from previous iterations, and $p_t$ encodes prompt-based guiding instructions for ex: *Surpass the best response, avoiding previous mistakes.*

**Step 2: Verifier-guided selection.** Among the generated response and the previous best response, we select the one maximizing the reward function:

$$\hat{y}_{t+1} = \arg\max_{y \in (y_t, \hat{y}_t)} R(x, y) \tag{2}$$

IAD has the flexibility to incorporate critique-based feedback, such as LLM-judge, which identifies specific areas needing improvement. This extends to

$$y_{t+1} \sim \pi_0(\cdot|x, \hat{y}_t, p_t, fb_t) \tag{3}$$

where $fb_t$ highlights specific components (e.g., incorrect tags in HTML, invalid SQL joins), guiding refinement at a more granular level in an iterative fashion.

**Step 3: Acceptance Criterion.** A new response $y_{t+1}$ is accepted if it provides an improvement over the previous best, $R(x, y_t) - R(x, \hat{y}_t) > 0$ (where $\delta = 0$ is the special case)

**Step 4: Iterative Refinement.** The accepted response $\hat{y}_t$ updates the context for subsequent generations, progressively re-weighting the proposal distribution toward higher-quality outputs. By conditioning on $\hat{y}_t$, the sampling process is guided towards responses with higher rewards, reducing the gap between the reference distribution $\pi_0(\cdot|x, \hat{y}_t)$ and the optimal distribution $\pi^*(\cdot|x)$, as shown in experimental results.

For instance, in an SQL code generation task, the model might initially produce a non-functional or erroneous SQL statement. However, in the next iteration, it generates a different SQL code, and through verifier comparisons, we determine which version is better—e.g., if the later version passes more test cases, we infer that the model should move in that direction. Using an LLM as a judge makes this refinement process more targeted, as it can provide explicit feedback on errors like *"This condition is incorrect", "Fix the syntax here", or "This table join is unnecessary"* and suggest improvements at each iteration.

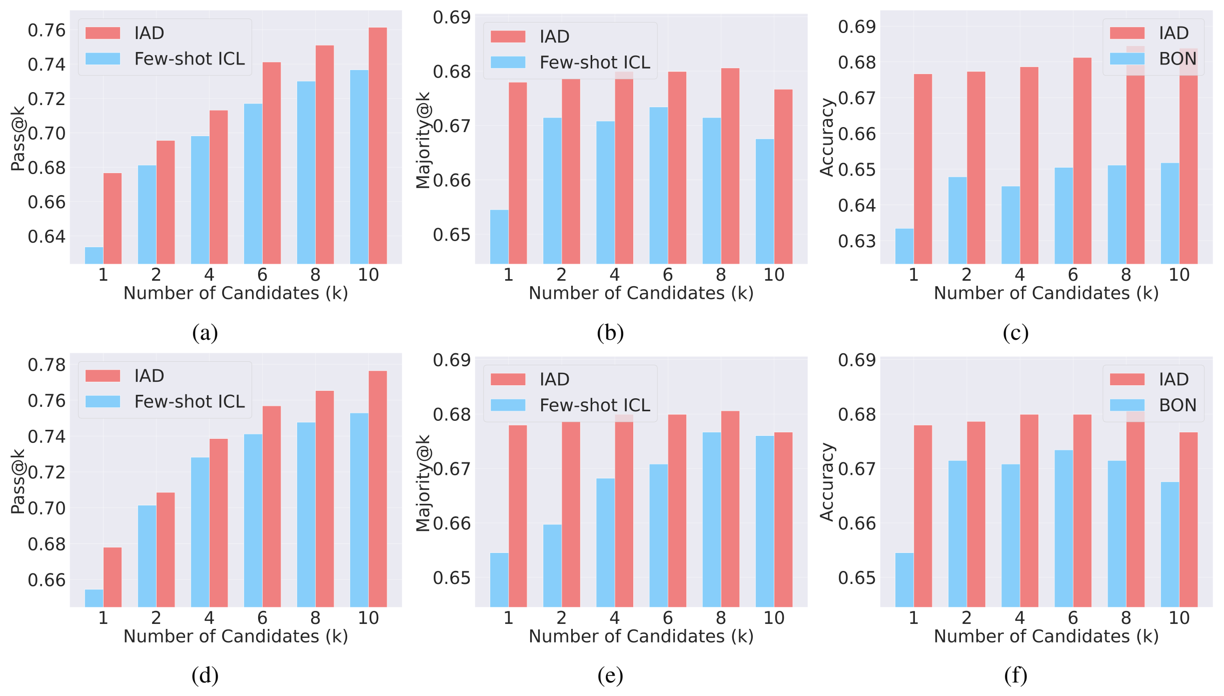


Figure 4: **Text2SQL Experiments**: (a) Pass@K performance comparison between queries generated using our approach and Few-shot ICL with Gemini-1.5-flash. (b) Majority@K performance comparison between queries generated using our approach and Few-shot ICL with Gemini-1.5-flash. (c) Accuracy comparison between the best-of-N method and our approach for Gemini-1.5-flash. (d) Pass@K performance comparison between queries generated using our approach and Few-shot ICL with Gemini-1.5-pro. (e) Majority@K performance comparison between queries generated using our approach and Few-shot ICL with Gemini-1.5-pro. (f) Accuracy comparison between the best-of-N method and our approach for Gemini-1.5-pro. IAD consistently outperforms all the baselines across the settings.

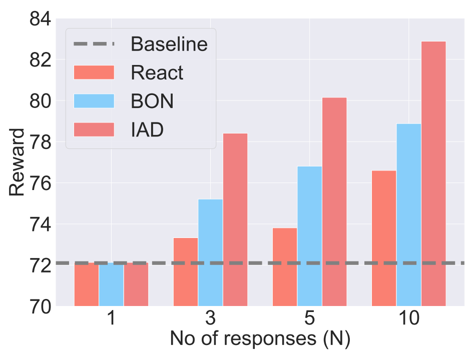


Figure 5: **Intercode** : Comparison of IAD with baselines on the accuracy–compute trade-off shows that IAD, with appropriate feedback, enables more effective test-time scaling

## 3 Experiment Setup

In this section, we provide detailed discussion on the experimental analysis with respect to the agentic tasks below. For all our experiments, we have utilized NVIDIA A100-SXM4-40GB GPU with 40GB of VRAM, running on CUDA 12.4 and driver version 550.90.07.

### 3.1 Environmental Details and Setup

To understand the role of feedback and evaluate IAD across the 4 keypoints, we perform empirical analysis on 4 challenging agentic environment benchmarks incldluing Text2SQL, Sketch2code, Intercode and Webshop.

**Text2SQL** tasks map natural language questions to executable SQL queries over structured databases. These tasks require deep reasoning to interpret user intent and generate syntactically and semantically correct queries. We use the BIRD benchmark (Li et al., 2024a), which includes 12,751 question-SQL pairs spanning 95 databases and 37 professional domains. Performance is evaluated using execution accuracy (EX), where a prediction is correct if it yields the same results as the ground truth query when executed.

**Sketch2code** (Li et al., 2024b) tests the multimodal abilities of agents by converting wireframe sketches into functional HTML prototypes. The task requires aligning visual cues with structured code, handling layout ambiguities, and generating precise UI structures. Evaluation is based on

| Model | Layout. | Txt IoU | Img IoU |
|---|---|---|---|
| **Single-Turn Approaches** | | | |
| Llava-1.6-8b* | 8.01 | 9.26 | 1.95 |
| Claude-3-Sonnet* | 14.22 | 15.85 | 6.62 |
| Gemini-1.5-Flash | 17.85 | 17.50 | 10.77 |
| Gemini-1.5-Pro | 18.25 | 18.20 | 12.69 |
| GPT-4o* | 19.20 | 17.12 | 16.19 |
| Gemini-1.5-Flash (CoT) | 19.84 | 19.13 | 10.02 |
| **Multi-Turn Approaches (`Gemini-1.5-Flash`)** | | | |
| Sk2code (N=2)** | 19.41 | 20.45 | 11.81 |
| Self-Refine (N=2) | 19.51 | 19.35 | 10.71 |
| BoN (N=2) | 21.45 | 20.1 | 13.5 |
| IAD (N=2) | **24.78** | **23.01** | **15.29** |
| **IAD-fb (N=2, K=2)** | **24.86** | 23.4 | 14.69 |
| Self-Refine (N=4) | 19.97 | 19.11 | 11.74 |
| Sk2code (N=4)** | 20.41 | 21.46 | 12.67 |
| BoN (N=4) | 24.02 | 22.59 | 15.91 |
| IAD (N=4) | **25.97** | **24.13** | **16.98** |
| **IAD-fb (N=4, K=4)** | **26.61** | 24.62 | 17.36 |
| Self-Refine (N=6) | 19.89 | 18.91 | 11.61 |
| Sk2code (N=6) | 21.43 | 21.53 | 13.78 |
| BoN (N=6) | 25.75 | 22.91 | 17.67 |
| IAD (N=6) | **26.75** | **24.91** | **19.12** |
| **IAD-fb (N=6, K=6)** | **27.95** | 24.99 | 19.01 |

Table 1: **Sketch2Code**: Performance comparison between single-turn and multi-response generation methods with Layout score as the evaluation metric. IAD consistently outperforms SoTA baseline by an absolute margin of 3–4% with lesser budget, showing the importance of feedback.

.

three metrics: Layout Similarity (IoU over UI components), Text IoU (text alignment accuracy) and Image IoU (CLIP-based visual similarity). These metrics strongly correlate with human judgment.

**Intercode** (Yang et al., 2023) evaluates agents in structured programming tasks across four domains: Bash, Python, SQL, and CTF. Each task involves a series of agent decisions, including command execution and error handling, requiring both syntactic precision and logical planning. Performance is measured using reward signals derived from task execution correctness. We focus on the Bash domain for our experiments, where agents must generate correct shell commands based on natural language instructions.

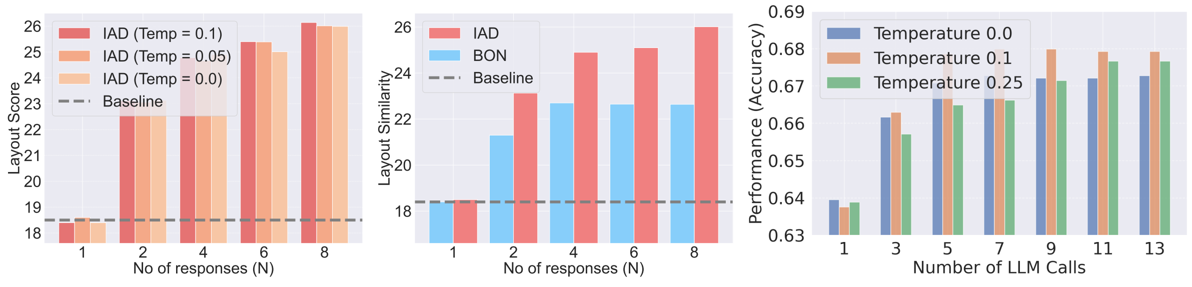
**Webshop** (Yao et al., 2023) simulates a real-world online shopping environment, requiring agents to perform long-horizon, language-guided decision-making. Given product queries, agents interact with webpages (search, click, select) to find matching items. Key evaluation metrics include: Success Rate (SR): whether the selected item satisfies all constraints. Progress Rate (PR): how closely the agent's actions align with task goals.

We study the effect of feedback in IAD along five key dimensions: (1) Accuracy vs compute - Budget controlled scaling (2) Impact of adaptive feedback beyond sampling diversity (3) Impact of feedback modalities, (4) sensitivity to feedback quality under varying levels of noise and sparsity

## 4 Experiment Results and Analysis

### 4.1 Accuracy vs Compute for Budget-Controlled Scaling

We analyze how IAD with appropriate and accurate feedback, improves performance under varying computational budgets and the demonstrate scaling behaviour across Sketch2Code, Text2SQL, Intercode and Webshop environments and benchmarks. We observe that IAD with appropriate feedback with limited achieving up 10% accuracy gain over strong feedback-free baselines like BON. However, as the computational budget increases either in terms of sample or model capability, the performance gap with and without feedback reduces and this finding is consistent across all the tasks. *Sketch2Code:* Layout Similarity, Text IoU, and Image IoU show consistent improvement with increasing iterations. IAD with just N=2 outperforms BoN and single-turn baselines by 3–4% absolute gain with weaker models like Gemini-1.5-Pro as seen in Figure 3, Table 1. However, as the model capability increases from Gemini-1.5-Pro, Gemini-2.0, Gemini-2.5-Flash and Gemini-2.5-Pro, the gap with and without feedback reduces which shows as model capability improves, parallel sampling can do well. *Text2SQL:* We evaluate IAD against few-shot chain-of-thought prompting and Best-of-N baselines under identical model settings (Gemini-1.5-Flash, Gemini-1.5-Pro). IAD, through iterative feedback and refinement, achieves higher execution accuracy using a comparable or smaller number of LLM calls Figure 4. With just three rounds of feedback, it effectively corrects syntactic and semantic errors in generated SQL queries, demonstrating how feedback enables more efficient use of the compute budget. To assess generality, we also compare IAD with diverse approaches—MCS-SQL, DIN-SQL, DAIL-SQL, and MAC-SQL—that do not rely on fine-tuning the BIRD train set, Table 3. The consistent gains across these comparisons illustrate that integrating feedback,is a key driver of improved performance under test-time budget constraints. Similar trend is observed in Intercode Figure 5 which shown sequential refinement with ap-



(a) (b) (c)

Figure 6: **Sketch2code**: (a) Compares the performance of IAD at low temperature across varying number of generations to disentangle the effect of stochasticity and the improvement from iterative feedback based on IAD.(b) Comparison with BON at temperature = 0.1, which shows BON improvement ceases after 2 iterations Both (a, b) highlights the improvement due to adaptive refinement with verifier(c) **Text2SQL** :Compares the accuracy of IAD on BIRD development set at low temperature across varying number of generations to highlight the improvement from iterative feedback based on IAD.

propriate feedback outperforms baselines with low budget. *Webshop:* IAD outperforms BoN-SC and strong baselines like GPT-4o and Gemini. For example, SR improves from 29.3% (Gemini-1.5-Pro) and 41.09% (BoN-SC + GPT-4o) to **44.68%** (IAD + GPT-4o), a 3–4% absolute gain. In weaker models like Gemini-1.5-Flash, iterative refinement significantly boosts performance where BoN plateaus.

| **Models** | **(PR)** | **(SR)** |
|---|---|---|
| Lemur-70b | 71 | 11 |
| Mistral-7b | 68.2 | 13.9 |
| Vicuna-13b-16k | 73 | 21 |
| Gemini-1.5-Flash | 71.3 | 26.5 |
| Gemini-1.5-Pro | 71.9 | 29.3 |
| BoN-SC + Gemini-1.5-Pro | 72.12 | 30.31 |
| IAD + Gemini-1.5-Pro | 71 | 38.3 |
| GPT-4 | 75.8 | 38.5 |
| GPT-4o | 73.1 | 40.3 |
| BoN-SC + GPT-4o | 74.21 | 41.09 |
| **IAD + GPT-4o** | 74.6 | **44.68** |

Table 2: **Webshop**- Progress Rate (PR) and Success Rate (SR) for Models in the Webshop Environment(Yao et al., 2023). Perform a comparison of IAD against Baselines for the Webshop with the evaluation similar to followed in Agentboard (Ma et al., 2024)

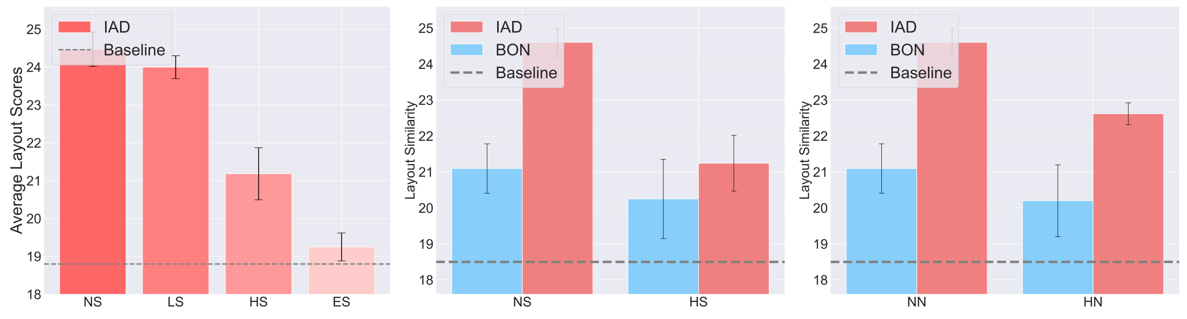
## 4.2 Impact of Adaptive Feedback Beyond Sampling Diversity

Prior works lack a clear experimental setup to disentangle the true source of improvement in their approaches. A key concern thus exists is whether the gains arise from sampling diversity, or from the actual effectiveness of the feedback. However, most prior methods do not include comparisons against stochastic sampling baselines like BON, leaving it unclear whether their feedback mechanisms meaningfully contribute to performance improvements. To answer this, we conduct a controlled experiment to isolate the effect of adaptive feedback in IAD from sampling diversity. Specifically, we reduce the generation temperature (to 0.1, 0.05, and 0.0), thereby minimizing randomness in the generation process. We then evaluate the performance of both IAD and BON over multiple iterations under these low-stochasticity settings. *Sketch2Code:* BoN saturates at layout score ~21.9 even with N=6, while IAD surpasses 26 with 6 iterations. Figure 6 confirms that adaptive refinement—not randomness—drives gains. *Text2SQL:* Accuracy increases steadily with IAD even at low temperature settings (Figure 6). This validates feedback's role in semantic correction.

**Key Insight**: The results demonstrate that the gains from IAD are not merely due to diversity in generation/sampling, but rather from the verifier-guided adaptive feedback-driven refinement. This also highlights a critical insight that when the diversity in the policy is low, feedback based sequential approaches can perform significantly better than sampling based BON approaches.

## 4.3 Design and role of feedback form

We investigate the impact of different forms of feedback on inference alignment. When feedback is textual, it is relatively easy to integrate as is. However, designing effective feedback from scalar rewards and preferences remains underexplored. In IAD, we focus on various methods to extract useful signal from scalar rewards and demonstrate performance gains crucially depend on the design of proper feedback. We analyze both the scenarios -1. Textual feedback from LLM as a judge



(a) (b) (c)

Figure 7: **Sketch2code**: (a) Represents the performance of IAD with N = 2 w.r.t Layout score with varied sparsity NS (No Sparsity), LS (Low Sparsity) HS (High Sparsity) and ES (Extreme) (b) Performance comparison of IAD with BON with N = 2 at varied sparsity levels which shows the gains over BON reduces with high sparsity (c) Performance comparison of IAD with BON with N = 2 at varied noise levels of reward score which shows gains reduces over BON. However, IAD still improves over baselines under Sparse and Noisy feedback.

2. Scalar rewards. Majority of the prior sequential approaches including (Madaan et al., 2023) have been designed with a focus on approach 1 i.e LLM as a judge and taking feedback from the same. However, we demonstrate that directly using off-the-shelf LLMs as judges fails to deliver consistent or monotonic improvements in complex scenarios (Figure ). Our experiments reveal that performance often plateaus, fluctuates, or even degrades across iterations. To highlight this, we include both quantitative and qualitative examples (Appendix) where self-LLM feedback is repetitive and uninformative—failing to identify or correct meaningful issues. This supports our central claim: for effective inference-time optimization, near-optimal verifiers are essential, especially in complex agentic scenarios like Sketch2code, text2sql.

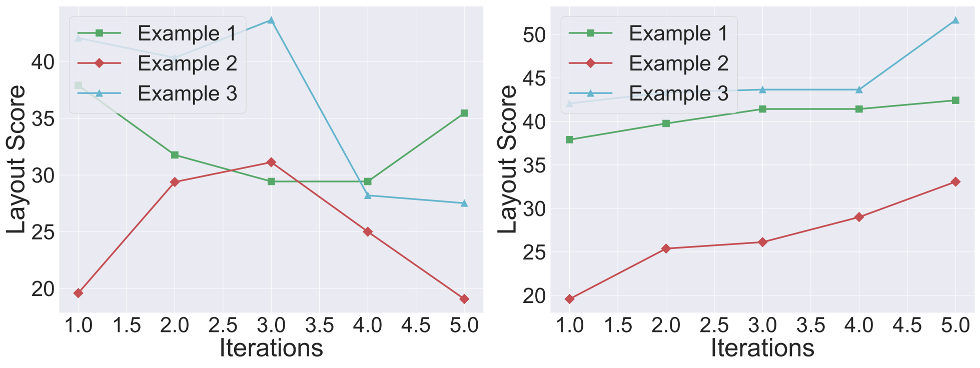
**How to design feedback from a scalar reward?** A crucial challenge lies in designing meaningful and informative feedback from scalar reward or preference, since the eventual feedback needs to be in textual form. Thus IAD is designed to extract as much signal as possible from the available feedback. Rather than relying solely on absolute reward scores, IAD transforms score-based or comparative feedback into structured guidance by identifying the best and worst responses at each iteration. These are explicitly fed back into the model through prompt conditioning, providing a clear directional signal for improvement. This process not only reinforces the distinction between good and bad outputs but also enables the use of dense feedback—even from weak or indirect supervision sources—turning minimal signals into effective updates. Intuitively, this approach is analogous to zeroth-order optimization where two sampled values from the objective function are sufficient to guide the optimization process toward the maximum. Similarly, feedback on the best and worst responses helps steer the model. We compare IAD with the above feedback for Sketch2code and show that it performs almost comparable with LLM judge with the reference

### 4.4 Sensitivity to feedback quality

We study the sensitivity and robustness of inference alignment to feedback quality via controlled sparsity and noise in feedback signals.

**Sparse Rewards.** In many agentic tasks, dense reward signals may not be available. We study this setting in the context of Sketch2code where we systematically sparsify our verifier/reward model by varying the level of feedback sparsity and comparing the performance of IAD against the BON baseline under Low, High, and Extreme Sparsity conditions. Sparsification was achieved by providing feedback only when the verifier score exceeded a threshold t; the higher the threshold, the sparser the reward signal. We define three such sparsity levels and evaluate the performance of IAD versus BON with N=2 responses per prompt.

We observe that as the level of sparsity increases, the performance gap between IAD and BON narrows. Both inference-time approaches—BON and IAD—show a decline in performance, approaching the baseline under extreme sparsity conditions. The primary hypothesis behind the drop in IAD performance is that IAD relies on adaptive feedback: the LLM is conditioned on the best and worst responses from previous iterations, akin to a zeroth-order optimization method. Feedback on both ends (positive and negative) helps steer the model to-



(a) (b)

Figure 8: **Sketch2code**: (a) Represents the performance of Self-Refine (Madaan et al., 2023) based approaches across iterations using LLM as a judge. (b) Represents the performance of IAD with scalar score which shows monotonic improvements, highlighting the importance of optimality of feedback and verification

ward reward-maximizing generations through updates to the system prompt. However, under extreme sparsity, most responses—regardless of quality—receive zero reward. This results in random selection and noisy update directions, limiting IAD's ability to effectively adapt and improve.

**Noisy Verification.** To further investigate the effect of noisy verification and how it impacts the performance of IAD compared to BON, we introduce varying levels of noise into the reward signal. Specifically, we add Gaussian noise with different variances to the reward scores, simulating imperfect or noisy verifier conditions. We then run both IAD and BON with N=2 responses per prompt and evaluate their performance under these noisy settings. This setup allows us to assess the robustness of IAD to reward noise and compare its stability and effectiveness relative to BON when the verification signal is noisy.

We observe trends similar to those in the sparse reward setting. Notably, IAD remains reasonably robust to mild noise in the reward signal. This is because IAD relies on adaptive feedback, where the LLM is conditioned on the best and worst responses from previous iterations—effectively leveraging pairwise comparisons rather than absolute scores. As long as the noise is limited, the relative preference between responses is preserved. For instance, if one layout score is 0.5 and another is 0.3, mild noise might shift them to 0.55 and 0.24, respectively—maintaining the same ordering. Therefore, IAD continues to improve under reasonable noise levels. However, as noise increases significantly, it can flip these preferences, leading to unstable updates and a decline in performance.

| Method | Exe Acc |
|---|---|
| DIN-SQL + GPT-4 | 50.72 |
| DAIL-SQL + GPT-4 | 54.76 |
| MAC-SQL + GPT-4 | 57.56 |
| MCS-SQL + GPT-4 | 63.36 |
| E-SQL + GPT-4o | 65.58 |
| **IAD (Ours) + GPT-4o** | 65.97 |
| **IAD (Ours) + Gemini-1.5-pro** | **68.05** |

Table 3: **Text2SQL** - Execution accuracy comparison of previous works with our proposed approach

## 5 Conclusion

In this work, we explore the underexplored role of feedback in inference-time alignment for black box AI agents. To understand the effect of feedback in inference alignment, we introduce Iterative Agent Decoding (IAD), a general sequential framework. Our study analyzes feedback through four lenses: (1) accuracy vs. compute trade-offs, (2) gains beyond sampling diversity, (3) feedback modality integration, and (4) sensitivity to feedback quality. Empirically, we find that feedback is especially valuable under constrained budgets—achieving up to 10% gains over feedback-free baselines. We also highlight the challenge of integrating diverse feedback modalities into sequential designs, not critically explored in literature. While textual feedback integrates naturally, representing scalar or preference signals remains an open challenge. We also observe that IAD's benefits diminish under highly sparse or noisy feedback, underscoring the importance of feedback fidelity for effective alignment.

## Limitations

While IAD – our iterative decoding approach – improves upon prior baselines by better leveraging verifier feedback, it is inherently sequential, leading to increased user facing latency compared to easily parallizable BoN approaches. Addressing this tradeoff between quality improvement, computational cost, and user facing latency remains an important area for future research, which may require properly combining these techniques with adaptive stopping, controlled decoding (Mudgal et al., 2024), speculative decoding (Leviathan et al., 2023). Additionally, more efficient verifier-guided selection could improve the efficiency in iterative decoding for agentic tasks. As we learnt, the verifier (or judge) plays a crucial role in our approach. Thus a more concrete investigation and selection of a judge for these challenging tasks is a valid and crucial next step of our work. We highlight that this work is of academic nature and has no direct or immediate harmful impacts to society. However, since this work deals with improving AI agents, it should be done under safety protocols and guidelines. We want to highlight that this study is limited to English language text primarily due to the nature of open-source datasets used.

## References

Afra Amini, Tim Vieira, and Ryan Cotterell. 2024. Variational best-of-n alignment. *Preprint*, arXiv:2407.06057.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Ahmad Beirami, Alekh Agarwal, Jonathan Berant, Alexander D'Amour, Jacob Eisenstein, Chirag Nagpal, and Ananda Theertha Suresh. 2024. Theoretical guarantees on the best-of-n alignment policy. *Preprint*, arXiv:2401.01879.

Souradip Chakraborty, Soumya Suvra Ghosal, Ming Yin, Dinesh Manocha, Mengdi Wang, Amrit Singh Bedi, and Furong Huang. 2024. Transfer q star: Principled decoding for llm alignment. *Preprint*, arXiv:2405.20495.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. 2024. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. *Preprint*, arXiv:2403.08978.

Yingqi Gao, Yifu Liu, Xiaoxia Li, Xiaorong Shi, Yin Zhu, Yiming Wang, Shiqi Li, Wei Li, Yuntao Hong, Zhiling Luo, et al. 2024. Xiyan-sql: A multi-generator ensemble framework for text-to-sql. *arXiv preprint arXiv:2411.08599*.

Kevin G. Jamieson, Robert D. Nowak, and Benjamin Recht. 2012. Query complexity of derivative-free optimization. *Preprint*, arXiv:1209.2434.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024a. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Ryan Li, Yanzhe Zhang, and Diyi Yang. 2024b. Sketch2code: Evaluating vision-language models for interactive web design prototyping. *Preprint*, arXiv:2410.16232.

Yanming Liu, Xinyue Peng, Jiannan Cao, Shi Bo, Yuwei Zhang, Xuhong Zhang, Sheng Cheng, Xun Wang, Jianwei Yin, and Tianyu Du. 2024. Tool-planner: Task planning with clusters across multiple tools. *Preprint*, arXiv:2406.03807.

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

Karime Maamari, Fadhil Abubaker, Daniel Jaroslawicz, and Amine Mhedhbi. 2024. The death of schema linking? text-to-sql in the age of well-reasoned language models. *arXiv preprint arXiv:2408.07702*.

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. 2023. Self-refine: Iterative refinement with self-feedback. *Preprint*, arXiv:2303.17651.

Youssef Mroueh. 2024. Information theoretic guarantees for policy alignment in large language models. *Preprint*, arXiv:2406.05883.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, Jilin Chen, Alex Beutel, and Ahmad Beirami. 2024. Controlled decoding from language models. *Preprint*, arXiv:2310.17022.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022a. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022b. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O Arik. 2024. Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql. *arXiv preprint arXiv:2410.01943*.

Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. 2024. Chess: Contextual harnessing for efficient sql synthesis. *arXiv preprint arXiv:2405.16755*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. *Preprint*, arXiv:2309.03409.

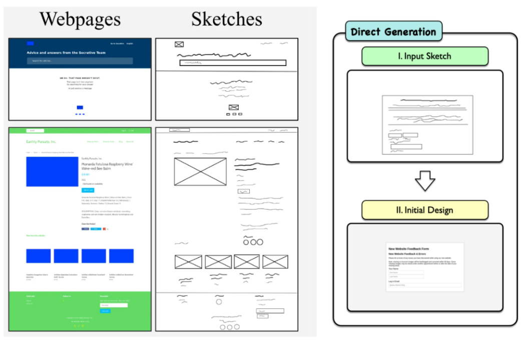
John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. 2023. Intercode: Standardizing and benchmarking interactive coding with execution feedback. *Preprint*, arXiv:2306.14898.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2023. Webshop: Towards scalable real-world web interaction with grounded language agents. *Preprint*, arXiv:2207.01206.

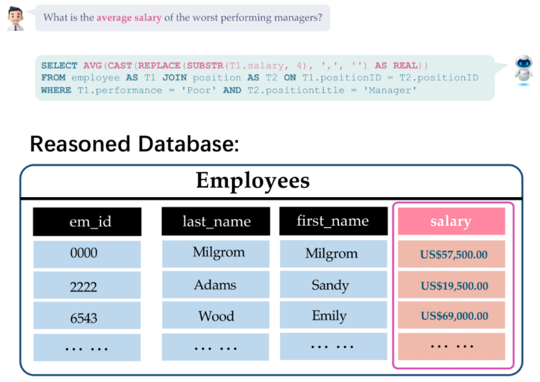
# A Appendix

## A.1 Detailed Environment Description

**1. Text-to-SQL** Text-to-SQL serves as a critical interface between natural language and structured query languages by enabling users to translate natural language queries into executable SQL commands. This functionality empowers individuals without SQL expertise to interact with complex databases, thereby facilitating data exploration, informed decision-making, automated analytics, and advanced feature extraction for machine learning. Generally, a Text-to-SQL system receives a natural language question and any pertinent metadata about the tables and columns, which serves as external knowledge to aid in database comprehension. Consequently, such systems are responsible not only for interpreting user intent and identifying relevant information from a potentially vast set of tables and columns but also for generating SQL queries that may include multiple conditions—a process that is inherently reasoning intensive. To evaluate our proposed framework, we employ the BIRD benchmark (Li et al., 2024a), a challenging and widely used dataset in the Text-to-SQL domain. BIRD comprises an extensive collection of 12,751 unique question-SQL pairs drawn from 95 large databases with a total size of 33.4 GB. The benchmark spans more than 37 professional domains, including blockchain, hockey, healthcare, and education, making it a comprehensive resource for assessing the robustness and generalizability of Text-to-SQL systems. The primary metric for model comparison in this domain is execution accuracy (EX), where the ground truth SQL query and the predicted SQL query are both executed over the target database, if they both generate same sets of results the accuracy for the predicted SQL query is considered as accurate.



(a) Sketch2Code Environment



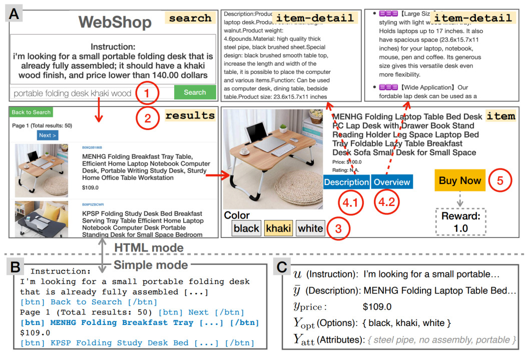
What is the average salary of the worst performing managers?
Reasoned Database:

```
SELECT AVG(CAST(REPLACE(SUBSTR(T1.salary, 4), ',', '') AS REAL))
FROM employee AS T1 JOIN position AS T2 ON T1.positionID = T2.positionID
WHERE T1.performance = 'Poor' AND T2.positiontitle = 'Manager'
```

Employees

| em_id | last_name | first_name | salary |
|---|---|---|---|
| 0000 | Milgrom | Milgrom | US$57,500.00 |
| 2222 | Adams | Sandy | US$19,500.00 |
| 6543 | Wood | Emily | US$69,000.00 |
| … … | … … | … … | … … |

(b) Text-to-SQL Environment



A
WebShop
search
Instruction:
i'm looking for a small portable folding desk that is already fully assembled; it should have a khaki wood finish, and price lower than 140.00 dollars
portable folding desk khaki wood
Search
1
2
results
Back to Search
Page 1 (Total results: 50)
Next >
MENHG Folding Breakfast Tray Table, Efficient Home Laptop Notebook Computer Desk, Portable Writing Study Desk, Sturdy Home Office Table Workstation
$109.0
KPSP Folding Study Desk Bed Breakfast Serving Tray Table Efficient Home Laptop Notebook Computer Desk Portable Standing Desk for Small Space Bedroom
item-detail
item-detail
item
MENHG Folding Laptop Table Bed Desk PC Lap Desk with Drawer Book Stand Reading Holder Leg Space Laptop Bed Tray Foldable Lazy Table Breakfast Desk Sofa Small Desk for Small Space
Price: $100.0
Rating: N.A.
Description
Overview
4.1
4.2
Buy Now
5
Reward: 1.0
Color
black
khaki
white
3
HTML mode
Simple mode
B
C

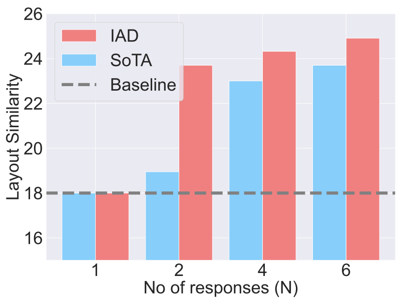
```
Instruction:
I'm looking for a small portable folding desk
that is already fully assembled [...]
[btn] Back to Search [/btn]
Page 1 (Total results: 50) [btn] Next [/btn]
[btn] MENHG Folding Breakfast Tray [...] [/btn]
$109.0
[btn] KPSP Folding Study Desk Bed [...] [/btn]
```

$u$ (Instruction): I'm looking for a small portable…
$\bar{y}$ (Description): MENHG Folding Laptop Table Bed…
$y_{\text{price}}$: $109.0
$Y_{\text{opt}}$ (Options): { black, khaki, white }
$Y_{\text{att}}$ (Attributes): { steel pipe, no assembly, portable }

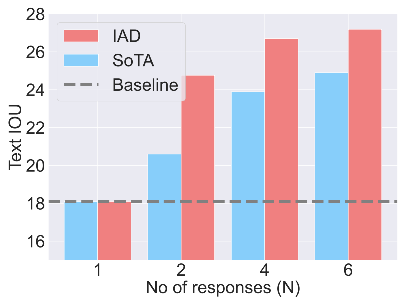
(c) Webshop Environment

Figure 9: These three figures given an overview of three diverse and challenging agentic tasks that we consider to evaluate the performance of agents with our proposed approach vs baselines -(a) Sketch2code(Li et al., 2024b) (b)Text2SQL (Li et al., 2024a) and (c) Webshop (Yao et al., 2023)

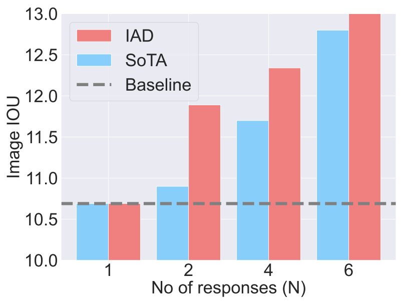
**2. Sketch2code**: Sketch2code (Li et al., 2024b) challenges and evaluates the multi-modal capabilities of agent where the objective is transform wireframe-style rough userk sketches into functional HTML prototypes with embedded CSS. Sketch2Code uniquely tests multi-modality, requiring structured code generation from imprecise visual input, often leading to misaligned text, incorrect spacing, and structural



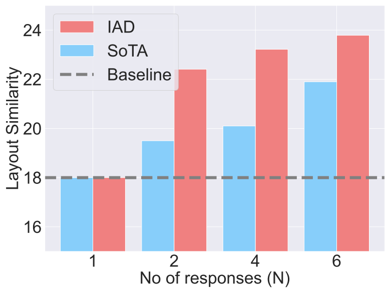
(a)



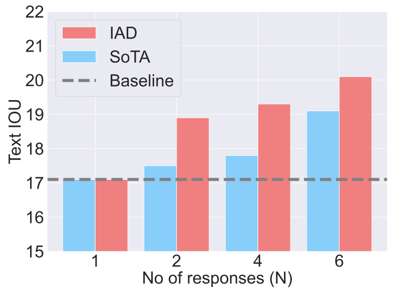
(b)



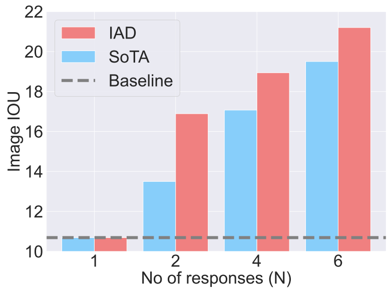
(c)



(d)



(e)



(f)

Figure 10: **Sketch2code**: This figure provides a comparison of IAD (ours) with Best-of-N sampling (SoTA) and single-turn generation with Gemini-1.5-Flash for the metrics - (a, d) Layout Similarity (b, e) TextIoU (c, f) ImageIoU across varying the number of generations (N). Top 3 rows, the optimization is done taking Text IOU as the verifier and the bottom 3 rows with Image IOU as the verifier. So, this also shows both the generalisability and performance improvement of IAD over baselines.

inconsistencies. This leads to challenges such as misaligned text, incorrect spacing, missing components, structural inconsistencies, making it an extremely challenging benchmark for multimodal LLMs. The complexity of this task arises from: ambiguity in hand-drawn sketches, where component boundaries, spacing, and positioning are not precisely defined. The evaluation of the generation is done primarily with three key metrics : *Layout Similarity, Text IOU, Image IOU*. Layout Similarity (IoU-based metrics): Intersection-over-Union (IoU) is computed for different UI components (e.g., buttons, images, text blocks) to measure how well their positions match the reference. Intersection-over-Union (IoU) is computed for different UI components (e.g., buttons, images, text blocks) to measure how well their positions match the reference implementation. Text-IOU similarly measures how accurately the generated text aligns with the reference design. Image IOU uses CLIP embeddings to compare the visual appearance of the generated webpage with the reference design and evaluates color similarity, element positioning, and component rendering. These metrics provide a reliable way to measure the quality of the generated response and strongly correlates with human judgement. Evaluations are done also with LLM as judge to compare the performance.

**3. Webshop** is a large-scale, web-based interactive environment designed to test an AI agent's capability to perform sequential decision-making in an online shopping scenario under sparse feedback (Yao et al., 2023). The environment is modeled as a partially observable Markov decision process, where the agent navigates a simulated e-commerce platform to fulfill a user's product request based on natural language instructions. At each step, the agent receives an observation in the form of a webpage—such as search results, product details, or checkout options—and must decide on an action, including searching for a product, clicking on an item, or selecting options. The evaluation is based on success rate (SR), which measures whether the agent successfully selects a product that matches all specified criteria (attributes, price, and options), and task score, which represents the overall alignment of the final selection with the given instruction. The WebShop environment presents significant challenges, including sparse rewards (since feedback is only provided at the end of an episode), the need for strategic backtracking and exploration, and handling noisy or ambiguous natural language instructions. This setup makes WebShop a

**Algorithm 1** Proposed Approach: Iterative Decoding black box Inference with AI Agents

```
Require: Proposal distribution $\pi_0(\cdot|x, \hat{y}_t)$, input prompt $x$, reward function $R(x, y)$, threshold $\delta > 0$,
    number of iterations $T$
Ensure: Final accepted response $\hat{y}_T$
 1: Initialize: Sample an initial response $y_0 \sim \pi_0(\cdot|x)$
 2: Compute its reward $r_0 = R(x, y_0)$
 3: Accept the initial response: $\hat{y}_0 \leftarrow y_0$ and $r^* \leftarrow r_0$
 4: for $t = 1, 2, \ldots, T$ do
 5:     Sample a new candidate response $y_t \sim \pi_0(\cdot|x, \hat{y}_{t-1})$
 6:     Compute its reward $r_t = R(x, y_t)$
 7:     if $r_t - r^* > \delta$ then
 8:         Accept the candidate: $\hat{y}_t \leftarrow y_t$ and $r^* \leftarrow r_t$
 9:     else
10:         Reject the candidate: $\hat{y}_t \leftarrow \hat{y}_{t-1}$
11:     end if
12: end for
13: return $\hat{y}_T$
```

rigorous benchmark for evaluating long-horizon reasoning, language understanding, and decision-making in real-world-like online navigation scenarios.

**Remark (Why it works?).** The generation of better responses occurs through stochastic sampling in each iteration from the base model, conditioned on the best response so far (and the worst candidate), followed by a pairwise comparison from the verifier. This feedback mechanism helps guide the generator to sample responses with higher expected rewards over the iterations. Practically, we achieve this by incorporating prompts like "Improve upon the best response while avoiding mistakes from the worst response." Additionally, explicit feedback from a judge (e.g., verifier critiques or an LLM acting as a judge) accelerates the improvement process by providing targeted guidance. Intuitively, this approach is analogous to zeroth-order optimization (Jamieson et al., 2012; Yang et al., 2024), where two sampled values from the objective function are sufficient to guide the optimization process toward the maximum. Similarly, feedback on the best and worst responses helps steer the model toward generating responses that maximize rewards, reinforced through system prompts.

## A.2 Limitation of Single-turn Approach

In this section we characterize the performance gap $\Delta$ as the difference between the reward the optimal or ground-truth agent is achieving vs the reward achieved by the reference achieved by the reference agent policy.

$$\begin{aligned}
\Delta &= \mathbb{E}_{y\sim\pi^*(\cdot|x)}[R(x, y)] - \mathbb{E}_{y\sim\pi_0(\cdot|x)}[R(x, y)] \\
&\leq \sup_{R\in\mathcal{R}} \mathbb{E}_{y\sim\pi^*(\cdot|x)}[R(x, y)] - \mathbb{E}_{y\sim\pi_0(\cdot|x)}[R(x, y)] \\
&\leq \|R\|_{\max} d_{\mathrm{TV}}(\pi^*(\cdot|x), \pi_0(\cdot|x)),
\end{aligned}$$

where $R(x, y)$ represents the reward function measuring the quality of the generated response, and $d_{\mathrm{TV}}(\pi^*(\cdot|x), \pi_0(\cdot|x))$ is the total variation (TV) distance between the optimal policy $\pi^*(\cdot|x)$ and the reference policy $\pi_0(\cdot|x)$ (Mroueh, 2024). This result demonstrates that the performance gap $\Delta$ is inherently limited by the quality of the reference agent policy $\pi_0(\cdot|x)$, as measured by its divergence from the optimal policy. Thus, if $\pi_0(\cdot|x)$ is close to $\pi^*(\cdot|x)$ (in terms of TV distance), the performance gap will be small, resulting in near-optimal responses and viceversa.

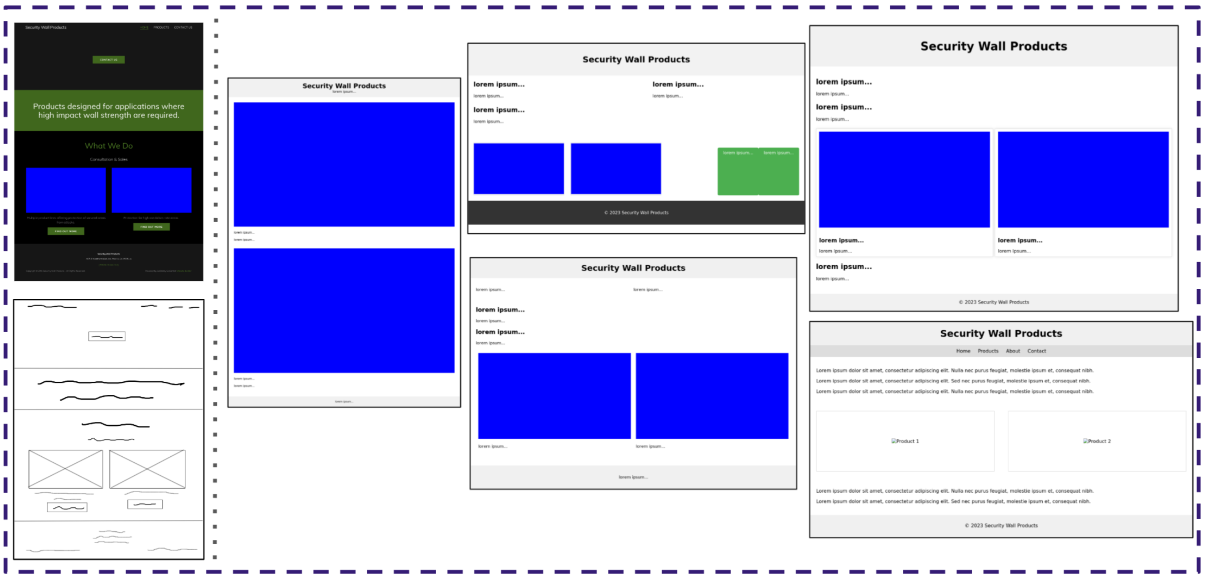
## A.3 Limitation of Prior approaches

In this section, we first provide a brief description of the baseline approaches and then discuss their pros and cons in this context.

**Single-turn Approaches**: In single-turn approaches, the response $y \sim \pi_0(\cdot|x)$ is directly generated from the reference agent policy. his method is straightforward, fast, and does not rely on a verifier, making it applicable even in verifier-agnostic settings. However, as evident from Figure 1, direct generation—even with SoTA models like Gemini-1.5-Pro, Gemini-1.5-Flash, GPT-4, and Claude—remains highly suboptimal for complex tasks like Sketch2Code and Text2SQL, also highlighting the difficulty of these tasks. Thus in single-turn generation, the performance is limited by the quality of the reference policy $\pi_0(\cdot|x))$ where larger f-divergence indicates greater misalignment.

**BoN sampling:** Best-of-N sampling improves upon single-turn generation by drawing $N$ i.i.d samples from the reference policy $\pi_0(\cdot|x)$ and selecting the highest-reward response based on the verifier $R(x, y)$. BoN is simple, parallelizable, and computationally efficient and doesn't rely on logits/model access thus applicable to black box agentic scenarios. It works even with scalar rewards and has been shown to achieve near-optimal tradeoffs between win rate and KL divergence (Beirami et al., 2024; Amini et al., 2024). Despite its advantages, BoN remains limited by the quality of the reference policy lacks the ability to iteratively refine responses based on verifier feedback. For example: BoN cannot incorporate targeted feedback, such as refining specific HTML structures in Sketch2Code or correcting systematic SQL errors in Text2SQL (further details in exp section)

**Controlled decoding:** Majority of prior decoding-based methods (Mudgal et al., 2024; Chakraborty et al., 2024) rely on access to logits for controlled generation , making them inapplicable in black box inference settings. While block-wise decoding (Mudgal et al., 2024) can be applied without logits as well however improper block selection disrupts syntax and semantics for structured generation (Appendix).


Security Wall Products
Products designed for applications where high impact wall strength are required.
What We Do
Security Wall Products
lorem ipsum...
© 2023 Security Wall Products
Home Products About Contact
Product 1
Product 2

(a)

Figure 11: **Sketch2code**: Qualitative evaluation of the generated HTMLs with BoN sampling (N=4) corresponding to the user-sketch (left-bottom) and reference html (left-top). The figures show that BoN performs much better in matching the reference HTML but still misses specific properties like rectangular structue, position of text, relative positioning of smaller blocks etc.

### A.4 Detailed Experimental Analysis

### A.5 Text-to-SQL Detailed Results and Analysis

In this section, we detail the experiments conducted on the BIRD text-to-SQL benchmark (Li et al., 2024a). For these experiments, we employed the Gemini-1.5-pro and Gemini-1.5-flash models both to generate actions at each state and as judge models to predict the reward. At each state, the LLM is provided with the database schema and the user's query, based on which it generates a draft SQL query. This draft query is then evaluated by the judge model, which also produces feedback on how to improve

**Sketch2code : Oracle Judge Prompt for providing Feedback**

**Input: Act as you are a front-end designer working with a code agent to implement an HTML webpage . You are provided with two images : the first image is the reference webpage, and the second one is the current implementation from the code agent . Note that images have already been replaced with blue rectangles as the placeholder. The task is to carefully compare the agent 's implementation against the reference webpage , and provide feedback to help the agent make its implementation closer to the reference webpage . Your feedback should be specific to the differences in layouts and visual components on the two webpages. If required provide small code snippets to help the user-agent but provide very few lines. Don't focus on the style components too much and focus on layout similarity and visual match with the reference webpage.**

the draft. The LLM uses this feedback to generate a revised query, establishing a self-correction loop. Finally, the answer with the highest reward value is selected as the candidate output. This process can be repeated to generate multiple candidate SQL queries. We then apply self-consistency (Wang et al., 2022) by executing all candidate queries over the database, grouping them based on their execution results, and selecting a query from the largest result cluster as the final answer. In the following sections, we first compare our proposed method with the widely used few-shot prompting approach in terms of Pass@k performance and final accuracy after self-consistency (Majority@K) using execution accuracy as the metric in order to demonstrate that using our method we can generate a pool of candidates with a higher quality. Subsequently, we compare our approach with the best-of-N approach which is one of the strong baselines as a test-time compute approaches to demonstrate the effectiveness of the proposed framework. Finally, we compare our method with all previously proposed test-time methods on the BIRD development set benchmark, excluding works that rely heavily on fine-tuning LLMs (Pourreza et al., 2024; Talaei et al., 2024; Maamari et al., 2024; Gao et al., 2024) for a fair comparison.

**Comparing with Few-shot prompting** We compared our method with the widely used few-shot in-context learning approach for text-to-SQL tasks. We evaluated and reported the Pass@K and self-consistency performance for up to 10 candidates using both the Gemini-1.5-flash-002 and Gemini-1.5-pro-002 models, as illustrated in the Figure 4. As demonstrated by these figures, our approach consistently outperforms the few-shot in-context learning method by a significant margin on both pass@K and self-consistency scores.

**Comparing with Best-of-N** In this section, we compare our proposed framework with the well-established best-of-N method to highlight the importance of searching through possible answers based on their rewards. For this comparison, we generated up to 20 candidate queries for each sample in the BIRD development set and utilized the model itself to select the best answer from the candidates. As shown in the Figure 4, our proposed method outperformed the best-of-N approach for both the Gemini-1.5-pro and Gemini-1.5-flash models, demonstrating the significance of incorporating a feedback loop to enhance the quality of candidate responses.

**Comparison with Previous works** In this section, we compare our approach with Gemini-1.5-Pro and other previous methods that rely on test-time computation. As shown in the Table 3, our method outperforms all previous approaches, demonstrating the effectiveness of the proposed framework in leveraging test-time computations to enhance model performance on the BIRD benchmark development set.

## A.6 Sketch2code

For Sketch2code (Li et al., 2024b), we provide a detailed comparison of our approach against SoTA baselines on several evaluation criterion and metrics. We used the hyperparameter setting of temperature = 0.5, max tokens = 4096, top p = 1.0, frequency/repetition penalty = 0.0, and presence penalty = 0.0

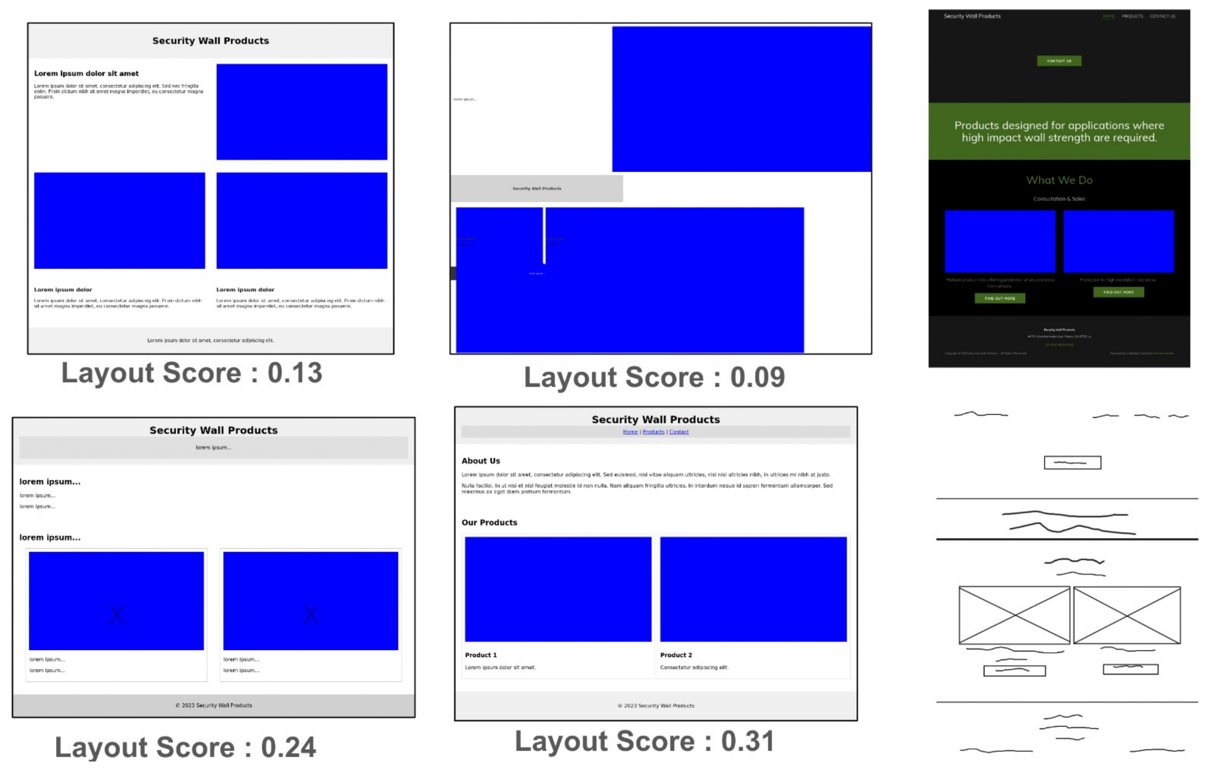


Figure 12: **Sketch2code**: Provides a qualitative verification of layout score as a metric and corresponding correlation to human judgement. It is evident that HTMLs with higher match with the reference layout (right-top) and user sketch(right-bottom) has higher layout score and vice-versa showing that its a valid metric.

for all our results. For the metrics, we consider metrics centring *1.Layout Similarity, 2. Visual IoU, 3. Text IoU* with reference HTML following (Li et al., 2024b). These metrics offer a comprehensive and reliable assessment of HTML generation quality, demonstrating a strong correlation ($\tilde{9}0\%$) with human satisfaction, as shown in (Li et al., 2024b) (further details in Appendix). Hence, we use Layout similarity as a verifier along with LLM-as-judge (Li et al., 2024b) to guide the generations for both BoN (Beirami et al., 2024) and IAD. We report comparison with baseline single-turn approaches including SotA models GPT-4o, Claude-3, InternVL2, Gemini-1.5-Flash, CoT and variants along with multi-response generation approaches including BoN, Sk2code and IAD (Ours). *Single turn* approaches even from SoTA models fail to match the layout structure, position of blocks, textual content, size of the blocks etc in the given user-sketch, causing a mismatch w.r.t to the reference layout as can be clearly seen in Figure and achieves a low score in-terms of all the three metrics in-comparison with multi-response generation approaches even with N=2. Best-of-N sampling (BoN) with a weaker model *Gemini-1.5-Flash* improves over single-turn approaches and , with $N = 4$ generations, it outperforms SoTA models with single-turn responses by a margin of 15-18%, by correctly identifying the block position, title block, overall layout structures etc. We see monotonic improvement in performance over the number of responses as the layout score improves from 20.41 to 25.7 with 6 responses. However, BoN struggles in incorporating fine-grained details about layout structure and makes some-times makes repetitive mistakes in the position of block in all the $N$ generation for the prompts (as shown in Fig). Our proposed approach IAD, mitigates this gap by iteratively improving the responses and as shown in Table 1, it achieves a major improvement of $\ 15\%$ from BoN as well as single-turn SoTA Claude with just 2 iterations (eq : N=2) even with simpler model *Gemini-1.5-Flash*. At each iteration, we pass the best and worst HTML as a context along with instruction, for generating the next iteration. We observe IAD is able to learn fine-grained layout components, image semantics over iterations with the context of the Best and Worst HTML. We see that with increased iterations, performance of IAD improves reaching to a very high layout score of 26.75, outperforming all baselines with same generations. We also report the Image and Text IoU scores while optimizing with the

**Sketch2code : Feedback from LLM as a Judge (Self Verification)**

**Feedback provided : Iter1 : The HTML structure and CSS styling do not reflect the provided wireframe. Iter2 :The layout needs to be revised to accurately represent the sketch's two-column image section and the distribution of text blocks. Iter 3: The layout of the text blocks and image containers does not accurately reflect the provided wireframe. The layout uses flexbox but doesn't accurately reflect the sketch's proportions and image placement. The large image should be centered and the smaller images should be positioned to the left of their respective text blocks. Iter4 : Implement a more precise grid-based layout using CSS grid or flexbox to achieve the correct positioning and sizing of all elements. Iter 5: The provided HTML closely resembles the wireframe but still needs significant layout adjustments. Use CSS Grid to precisely position and size all elements according to the wireframe's proportions.**

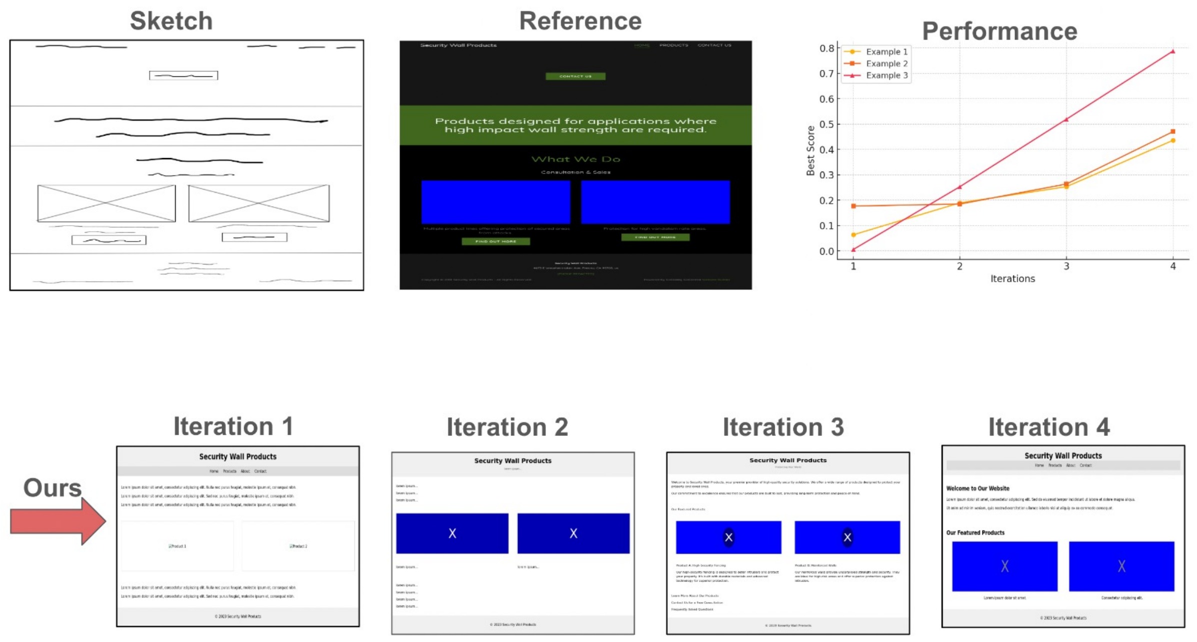


Figure 13: **Sketch2code** : Top row shows the user sketch, reference image and the performance of IAD over iterations. The figure highlights improvement of IAD over 4-turns w.r.t Layout similarity score (1/100) for 3 examples. It shows clear improvement over iterations. We also qualitatively analyse the snapshots of the HTMLs generated by the agent, which demonstrates that over iteration the qualitative performance improves and matches the input sketch/reference HTML.

layout-score, to check for reward-overoptimization of the metric.

However, as can be observed in Table 1 and Figure-3, that text and image similarities are also improving over the iterations and our findings regarding comparison with baseline BoN are consistent with the same. However, we observe that with increase number of generations the performance gets closer to BoN. We also consider sensitivity of the token-length of the context plays a critical role in this case, where providing the entire HTMLs can affect the entropy of the distribution, and thus over-conditioning can hinder structured generation by reducing diversity and exploration (as shown in Figure ). Thus, we provide only the top 100-200 tokens of the best (and worst) HTMLs. However, it is clear that if there would be a judge to highlight which portion of the code needs to be updated that will be more targeted. Hence, we incorporate LLM-judge (Gemini-1.5-Pro) which has the reference policy and it checks with the current response and provide feedback on improvement and sometimes snippets of HTML as well (however, we restrict that to 100 tokens  5-8% of the original HTML). This leads to a significant improvement of  36% for the layout score with just two iteration and final score of 31.98 with 6 iterations, demonstration the important of iterative approaches for agent performance. However, Sk2code (Li et al., 2024b) also performs feedback based design with LLM as a judge, however their approach doesn't yield major improvements for several models like Gemini-1.5, which we hypothesize can be due to the incorrect design of the method and also issues in the GPT-4 judge. Overall, in all our ablation our findings remain consistent where IAD outperforms baseline by a major margin.

## A.7 Importance of Verifier and Reward functions

In this section, we provide a motivation and importance of the verifier in ensuring monotonic improvement with our proposed approach. We define $\pi^*(\cdot|x)$ as the target policy generating $y^* \sim \pi^*(\cdot|x)$. At each step $t$, we sample $y_{t+1} \sim \pi_0(\cdot|x, \hat{y}_t)$, where $\hat{y}_t$ is the best response so far, and update $\hat{y}_{t+1} = \arg\max y \in (y_t, \hat{y}_t) R(x, y)$, accepting $yt + 1$ if $R(x, y_t) - R(x, \hat{y}_t) > 0$. If $R(x, y)$ incorporates the information of $\pi^*(y|x)$ (upto a normalization i.e $R(x, y) = \frac{f(\pi^*(y|x))}{Z}$, $f$ being a monotonic function), we show that our iterative refinement never deteriorates performance. In other words, we assume that for any two responses $y_1, y_2$, if the reward function satisfies $R(x, y_1) > R(x, y_2)$ then it implies that the optimal policy assigns a higher probability to $y_1$ than $y_2$, i.e $\pi^*(y_1|x) > \pi^*(y_2|x)$. A natural way to measure closeness to the optimal response is by estimating the distance under the true probability distribution (i.e target policy) $\pi^*(\cdot|x)$, defined as

$$d(\hat{y}_{t+1}, y^*) = \pi^*(y^*|x) - \pi^*(\hat{y}_{t+1}|x) \tag{4}$$

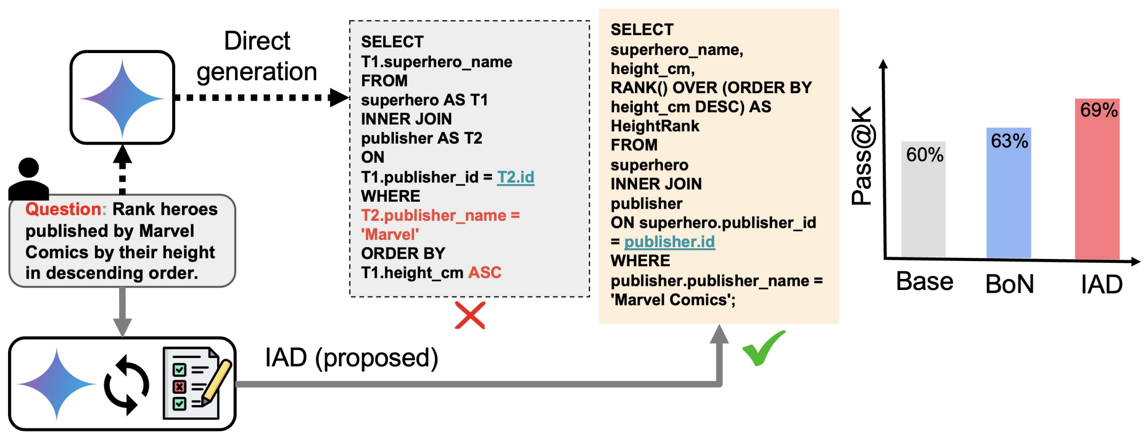
where the difference captures that how good the quality of the response is under optimal policy. If the response $\hat{y}_{t+1}$ is highly optimal, then $d(\hat{y}_{t+1}, y^*)$ will be low and viceversa, when $\hat{y}_{t+1} = y^*$, the gap will be zero.

$$\begin{aligned} d(\hat{y}_t, y^*) &= \pi^*(y^*|x) - \pi^*(\hat{y}_t|x) \\ &= \pi^*(y^*|x) - \pi^*(\hat{y}_t|x) - (\pi^*(\hat{y}_{t+1}|x) - \pi^*(\hat{y}_t|x)) \\ &\leq \pi^*(y^*|x) - \pi^*(\hat{y}_t|x) = d(\hat{y}_t, y^*) \end{aligned} \tag{5}$$

where, we first add and subtract the term $\pi^*(\hat{y}_t|x)$. Then by definition of our acceptance rule, we ensure that $\pi^*(\hat{y}_{t+1}|x) - \pi^*(\hat{y}_t|x) \geq 0$, where equality occurs when $\hat{y}_{t+1} = \hat{y}_t$. Thus we have $d(\hat{y}_t) <= d(\hat{y}_{t-1})$ i.e we ensure that the responses over the iteration are either improving or remain the same over iteration and won't deteriorate over iterations. However, it is important to note that this is based on the assumption that the reward function is aligned with the optimal distribution, meaning that selecting responses based on maximizing $R(x, y)$ leads to responses that are increasingly closer to the ground-truth distribution $\pi^*(\cdot|x)$.

**Verifier and Reward function**: We provide qualitative evaluation of considering layout similarity as a verifier due to its Interpretability and also correlation with human judgements also shown in (Li et al., 2024b). Additionally, we want to highlight that Sketch2code represents an extremely complex and challenging task for using self-LLM as a judge (Madaan et al., 2023) (without significant prompting) to compare between two generated HTMLs (by the agent) with its similarity to the input sketch and prompt. The input sketch has entirely different distribution than the image snapshot of the generated HTML which makes it harder for LLM as a judge to perform which is one of the reason we hypothesize that Self-refine (Madaan et al., 2023) type approaches doesn't provide improvements as shown in Table 1. On the other-hand, although LLM judge (oracle) provides more meaningful feedback when it has access to the reference HTML, however needs to be prompted efficiently to generate meaningful responses.

We accept the fact that our judge (oracle) for the feedback was allowed to provide more context than the one used in (Li et al., 2024b). However, the performance improvement in (Li et al., 2024b) feedback approch is very less and we hypothesize major reasons can be not performing IAD type approach, where we take previous best response (HTML) in the context along with specific instructions. Even for LLM-judge (oracle), we leverage feedback along with the previous best and worst HTMLs, which helps in providing more meaningful context to the agent in generating the correct HTML.

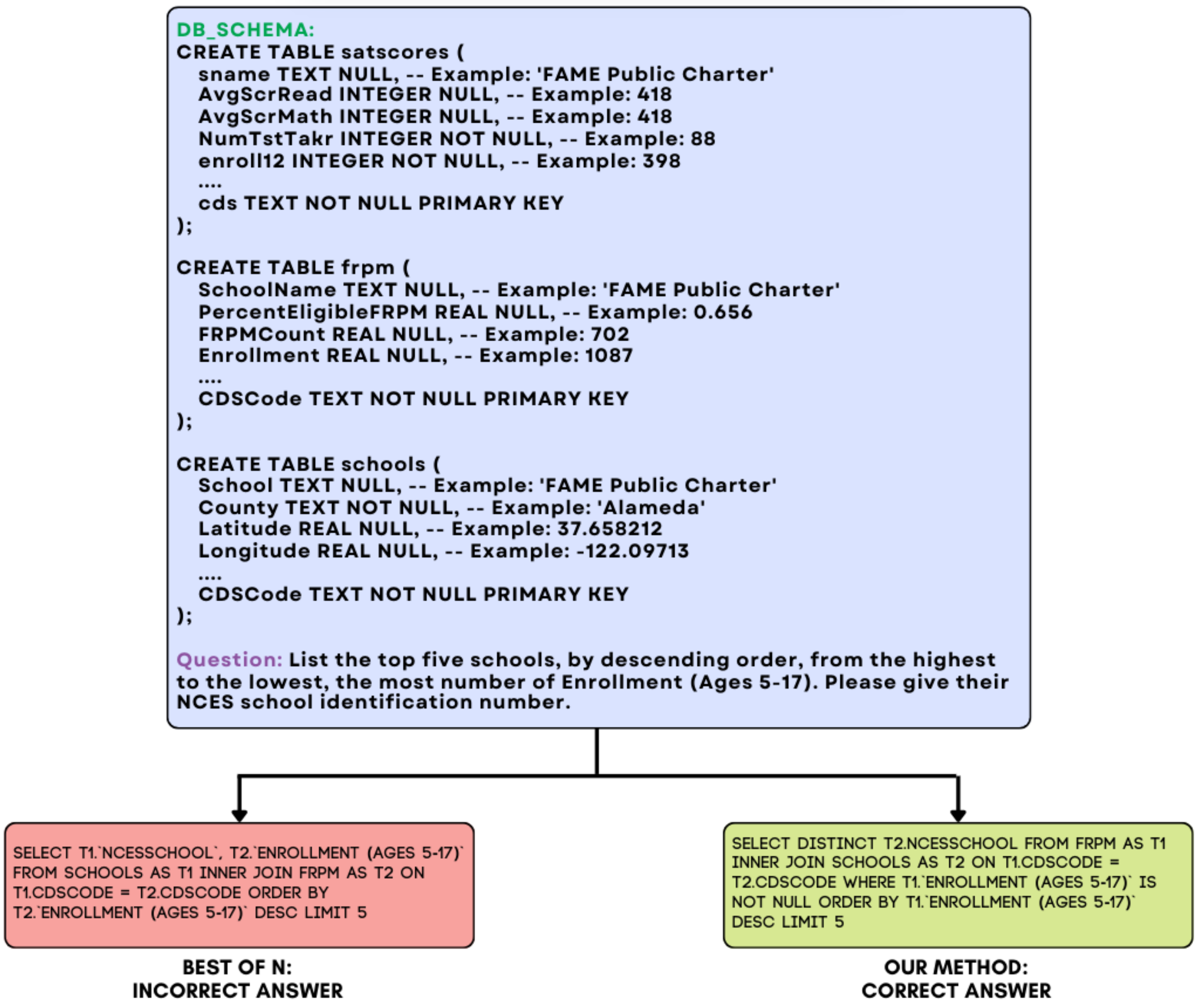


Question: Rank heroes published by Marvel Comics by their height in descending order.

```
SELECT
T1.superhero_name
FROM
superhero AS T1
INNER JOIN
publisher AS T2
ON
T1.publisher_id = T2.id
WHERE
T2.publisher_name =
'Marvel'
ORDER BY
T1.height_cm ASC
```

```
SELECT
superhero_name,
height_cm,
RANK() OVER (ORDER BY
height_cm DESC) AS
HeightRank
FROM
superhero
INNER JOIN
publisher
ON superhero.publisher_id
= publisher.id
WHERE
publisher.publisher_name =
'Marvel Comics';
```

Figure 14: Qualitative (Left) and Quantitative (Right) illustration of the performance benefit of IAD (Ours) over Single-turn response generations using Gemini-1.5 (Base) Text2SQL task. IAD improves performance by correctly handling query logic and joins, improving the accuracy over baseline and BoN (Best-of-N). move it to appendix



```
DB_SCHEMA:
CREATE TABLE satscores (
   sname TEXT NULL, -- Example: 'FAME Public Charter'
   AvgScrRead INTEGER NULL, -- Example: 418
   AvgScrMath INTEGER NULL, -- Example: 418
   NumTstTakr INTEGER NOT NULL, -- Example: 88
   enroll12 INTEGER NOT NULL, -- Example: 398
   ....
   cds TEXT NOT NULL PRIMARY KEY
);

CREATE TABLE frpm (
   SchoolName TEXT NULL, -- Example: 'FAME Public Charter'
   PercentEligibleFRPM REAL NULL, -- Example: 0.656
   FRPMCount REAL NULL, -- Example: 702
   Enrollment REAL NULL, -- Example: 1087
   ....
   CDSCode TEXT NOT NULL PRIMARY KEY
);

CREATE TABLE schools (
   School TEXT NULL, -- Example: 'FAME Public Charter'
   County TEXT NOT NULL, -- Example: 'Alameda'
   Latitude REAL NULL, -- Example: 37.658212
   Longitude REAL NULL, -- Example: -122.09713
   ....
   CDSCode TEXT NOT NULL PRIMARY KEY
);
```

Question: List the top five schools, by descending order, from the highest to the lowest, the most number of Enrollment (Ages 5-17). Please give their NCES school identification number.

```
SELECT T1.`NCESSCHOOL`, T2.`ENROLLMENT (AGES 5-17)`
FROM SCHOOLS AS T1 INNER JOIN FRPM AS T2 ON
T1.CDSCODE = T2.CDSCODE ORDER BY
T2.`ENROLLMENT (AGES 5-17)` DESC LIMIT 5
```

BEST OF N:
INCORRECT ANSWER

```
SELECT DISTINCT T2.NCESSCHOOL FROM FRPM AS T1
INNER JOIN SCHOOLS AS T2 ON T1.CDSCODE =
T2.CDSCODE WHERE T1.`ENROLLMENT (AGES 5-17)` IS
NOT NULL ORDER BY T1.`ENROLLMENT (AGES 5-17)`
DESC LIMIT 5
```

OUR METHOD:
CORRECT ANSWER

Figure 15: **Text2SQL**: An example of two responses is presented: the first response, generated using our proposed approach, is correct, while the second response, produced using the best-of-N method, is incorrect.

**Webshop - Task Execution Flow - IAD (Success)**

Search: `"blue color toothbrushes"` $\longrightarrow$ Product List Found $\longrightarrow$ Selected: `Hoomall Kids U-Shaped Toothbrush (Blue, $10.95)` $\longrightarrow$ Clicked on Product $\longrightarrow$ Purchased $\longrightarrow$ **Task Completed (Reward: 1.0)**

| Model | Layout. | Txt IoU | Img IoU |
|---|---|---|---|
| **Single-Turn Approaches** | | | |
| InternVL2-8b* | 4.01 | 4.89 | 1.41 |
| Llava-1.6-8b* | 8.01 | 9.26 | 1.95 |
| Claude-3-Sonnet* | 14.22 | 15.85 | 6.62 |
| GPT-4o-Mini* | 16.29 | 20.84 | 0.72 |
| Claude-3-Opus* | 17.11 | 18.09 | 8.32 |
| Claude-3-Haiku* | 17.52 | 20.60 | 2.72 |
| Gemini-1.5-Flash | 17.85 | 17.50 | 10.77 |
| Gemini-1.5-Pro | 18.25 | 18.20 | 12.69 |
| GPT-4o* | 19.20 | 17.12 | 16.19 |
| Gemini-1.5-Flash (CoT) | 19.84 | 19.13 | 10.02 |
| Claude-3.5-Sonnet* | 22.26 | 25.33 | 9.21 |
| **Multi-Turn Approaches (`Gemini-1.5-Flash`)** | | | |
| Sk2code (N=2)** | 19.41 | 20.45 | 11.81 |
| Self-Refine (N=2) | 19.51 | 19.35 | 10.71 |
| BoN (N=2) | 21.45 | 20.1 | 13.5 |
| IAD (N=2) | **24.78** | **23.01** | **15.29** |
| **IAD-fb (N=2, K=2)** | **24.86** | 23.4 | 14.69 |
| Self-Refine (N=4) | 19.97 | 19.11 | 11.74 |
| Sk2code (N=4)** | 20.41 | 21.46 | 12.67 |
| BoN (N=4) | 24.02 | 22.59 | 15.91 |
| IAD (N=4) | **25.97** | **24.13** | **16.98** |
| **IAD-fb (N=4, K=4)** | **26.61** | 24.62 | 17.36 |
| Self-Refine (N=6) | 19.89 | 18.91 | 11.61 |
| Sk2code (N=6) | 21.43 | 21.53 | 13.78 |
| BoN (N=6) | 25.75 | 22.91 | 17.67 |
| IAD (N=6) | **26.75** | **24.91** | **19.12** |
| **IAD-fb (N=6, K=6)** | **27.95** | 24.99 | 19.01 |

Table 4: **Sketch2Code**: Performance comparison between single-turn and multi-response generation approaches. For each of the multi-response generation method Layout score acts as the reference metric (temperature =0.6). Table demonstrate that IAD (Ours) consistently outperform SoTA baseline by >3-4% margin (absolute). N denotes the number of LLM calls for generating the HTML (>2000 tokens) and K represents the calls to LLM judge for getting feedback (<200 tokens).

.

**Webshop - Task: Buy a Folding Storage Box Ottoman- IAD (Success)**

**Size:** 60x40x40cm **Material:** Faux Leather **Price:** Under $170

- **Search** → "folding storage box ottoman faux leather 60x40x40cm"
- **Product List** → Found 50 results
  - Ottoman Footstool (40x40x40cm) - $149.97
  - Other options did not match size or price
- **Click** → Select "Ottoman Footstool"
- **Size Selection** → Click "60x40x40cm"
- **Buy Now** → Proceed to checkout
- **Task Completed**

| Query | Search Attempts | Results Found | Final Outcome |
|---|---|---|---|
| Men's Black Loafers (Size 10.5, Rubber Soles, <60) | Multiple searches, clicked "Next" repeatedly, found unrelated shoes (sneakers, sandals, pumps) | None matched the requirement | Task Failed - No suitable options found (Reward: 0.0) |
| Blue Diamond Almonds (Gluten-Free, Pecan, 12 Pack) | Repeated searches, encountered "No Search button" error multiple times, retrieved irrelevant snack items | Nut Thins Crackers, Keto Bars, M&M's Chocolate | Task Failed - No relevant product found (Reward: 0.0) |
| Folding Storage Box Ottoman (Faux Leather, 60x40x40cm, <170) | Initial product matched but had incorrect size, next searches returned irrelevant furniture items | Found an ottoman, but wrong size & overpriced | Task Failed - No exact match found (Reward: 0.0) |
| Official Cleveland University Drawstring Shorts (Small, Charcoal, Machine Washable, <60) | Search led to incorrect results (Marvel T-Shirts, Women's Yoga Shorts), agent attempted refinement but couldn't find exact product | No official Cleveland University shorts found | Task Failed - No suitable options found (Reward: 0.0) |
| Organic Hair Growth Serum Roller Set (For All Hair Types, <60) | Search retrieved some serums but none matched exact request (wrong quantity or expensive) | Found a set, but incorrect product version | Task Failed - No exact match found (Reward: 0.0) |

Table 5: **Webshop** : Highlights several Failure Cases of the Baseline Agent (Gemini-1.5-Pro) in Retrieving Relevant products given the task. This represents the challenge of current model in performing strategic exploration in Webshop.

**Webshop - Task: Buy a Vegan, Gluten-Free Protein Shake - IAD (Success)**

**Requirements:** 100% Vegan, Gluten-Free, Soy-Free **Price:** Under $40

- **Search** → "gluten free vegan plant based protein shake"
- **Product List** → Found 50 results
  - OWYN Protein Shake (Cold Brew Coffee, 12oz) - $11.07
  - Other products exceeded price or dietary restrictions
- **Click** → Select "OWYN Protein Shake"
- **Buy Now** → Proceed to checkout
- **Task Completed**

## A.8 Text2SQL

In this section we provide some example responses of Best of N approach and our proposed method on the BIRD development set samples.

**Text2SQL : Generated queries for best of N approach (Wrong answer)**

**DB schema:**
CREATE TABLE satscores (
enroll12 INTEGER not null,
dname TEXT null,
...
);
CREATE TABLE frpm(
IRC INTEGER null,
'Free Meal Count (K-12)' REAL null,
...
);
CREATE TABLE schools(
GSserved TEXT null,
Street TEXT null,
...
);
**Question:** Which active district has the highest average score in Reading?
**Predicted Query:** SELECT T1.District FROM schools AS T1 INNER JOIN satscores AS T2 ON T1.CDSCode = T2.cds WHERE T1.StatusType = 'Active' GROUP BY T1.District ORDER BY AVG(T2.AvgScrRead) DESC LIMIT 1

**Text2SQL : Generated queries for our method (Correct answer)**

**DB schema:**
CREATE TABLE satscores (
enroll12 INTEGER not null,
dname TEXT null,
...
);
CREATE TABLE frpm(
IRC INTEGER null,
'Free Meal Count (K-12)' REAL null,
...
);
CREATE TABLE schools(
GSserved TEXT null,
Street TEXT null,
...
);
**Question:** Which active district has the highest average score in Reading?
**Predicted Query:** SELECT T1.dname FROM satscores AS T1 INNER JOIN schools AS T2 ON T1.cds = T2.CDSCode WHERE T2.StatusType = 'Active' AND T1.AvgScrRead IS NOT NULL GROUP BY T1.dname ORDER BY MAX(T1.AvgScrRead) DESC LIMIT 1

**Text2SQL : Generated queries for best of N approach (Wrong answer)**

**DB schema:**
CREATE TABLE satscores (
enroll12 INTEGER not null,
dname TEXT null,
...
);
CREATE TABLE frpm(
IRC INTEGER null,
'Free Meal Count (K-12)' REAL null,
...
);
CREATE TABLE schools(
GSserved TEXT null,
Street TEXT null,
...
);
**Question:** Among the schools with the average score in Math over 560 in the SAT test, how many schools are directly charter-funded?
**Predicted Query:** SELECT COUNT(T1.CDSCode) FROM schools AS T1 INNER JOIN satscores AS T2 ON T1.CDSCode = T2.CDSCode WHERE T2.AvgScrMath > 560 AND T1.FundingType = 'Directly funded'

**Text2SQL : Generated queries for our method (Correct answer)**

**DB schema:**
CREATE TABLE satscores (
enroll12 INTEGER not null,
dname TEXT null,
...
);
CREATE TABLE frpm(
IRC INTEGER null,
'Free Meal Count (K-12)' REAL null,
...
);
CREATE TABLE schools(
GSserved TEXT null,
Street TEXT null,
...
);
**Question:**Among the schools with the average score in Math over 560 in the SAT test, how many schools are directly charter-funded?
**Predicted Query:** SELECT COUNT(DISTINCT T1.CDSCode) FROM schools AS T1 INNER JOIN satscores AS T2 ON T1.CDSCode = T2.cds INNER JOIN frpm AS T3 ON T3.CDSCode = T2.cds WHERE T2.AvgScrMath > 560 AND T3.'Charter Funding Type' = 'Directly funded' AND T3.'Charter Funding Type' IS NOT NULL