Review, Refine, Repeat: Iterative Decoding of AI Agents with Dynamic Evaluation and Selection

Anonymous ACL submission

Abstract

While AI agents have started excelling at various tasks, they may still struggle with complex structured generation and strategic planning. Improvements via standard fine-tuning is often impractical, as solving agentic tasks rely on black-box API access without control over model parameters. Inference-time methods offer a viable alternative, but existing approaches require white/gray-box access, limiting their applicability to black-box settings. A natural black-box solution is Best-of-N (BoN) sampling, a simple yet effective inference-time technique that operates without access to model weights or logits. However, BoN is inherently static and lacks iterative feedback integration, reducing its effectiveness in complex tasks. To address this, we propose IAD, an *iterative decoding* approach that combines iterative refinement with dynamic candidate evaluation and selection guided by a verifier, to improve upon BoN. IAD is flexible, model-agnostic, and seamlessly integrates with API-based models, making it broadly applicable to agentic tasks. We evaluate IAD on Sketch2Code, Text2SQL, and Webshop, where it consistently outperforms baselines by over 15% across multiple metrics and setups.

1 Introduction

AI agents have demonstrated remarkable capabilities across a wide range of tasks, including natural language understanding, code generation, and mathematical reasoning (Fu et al., 2024; Liu et al., 2024; Jiang et al., 2024; Patel et al., 2024; Huang et al., 2024; Liu et al., 2023). However, state-of-the-art AI agents still face significant challenges when handling tasks that require multimodal inputs and structured output generation (Costarelli et al., 2024; Kinniment et al., 2024; Liu et al., 2023; Verma et al., 2024; Valmeekam et al., 2024). For example, consider agentic tasks such as **Sketch2Code** (Li et al., 2024b), **Text2SQL** (Li et al., 2024a), and the web-navigation task **Webshop** (Yao et al., 2023). Sketch2Code requires an AI agent to generate structured HTML from rough sketches, often resulting in incorrect generations. Text2SQL involves translating natural language queries into executable SQL statements, demanding precise reasoning while avoiding semantic and logical errors. Meanwhile, Webshop is a sequential decision-making task that evaluates an agent's ability to navigate an e-commerce platform effectively. Across these tasks, AI agents consistently underperform compared to human experts, underscoring the limitations of current agentic models in handling complex, structured outputs and sequential reasoning (Yao et al., 2023; Liu et al., 2024; Li et al., 2024b) where agents achieve merely 20-30% performance in task specific metrics.

A key challenge in improving agentic AI systems is their ability to handle structured reasoning and multimodal tasks effectively. One promising approach to addressing this challenge is fine-tuning and AI alignment (Ouyang et al., 2022a). While supervised fine-tuning reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022a,b; Bai et al., 2022) has significantly enhanced the capabilities of generative models, it is not directly applicable to AI agents due to their inherently black-box nature. Most agentic systems and frameworks (Liu et al., 2024; Fu et al., 2024) operate in environments where internal model updates are infeasible, making fine-tuning-based alignment techniques ineffective. An alternative approach is an inference-time alignment, such as controlled decoding (Mudgal et al., 2024; Chakraborty et al., 2024; Khanov et al., 2024), which can align generative models without fine-tuning. However, these methods still rely on access to token-level logits from the reference model to steer responses, an assumption that does not hold for black-box agentic systems. These limitations highlight the need for an inference-time alignment technique that is (i) com-

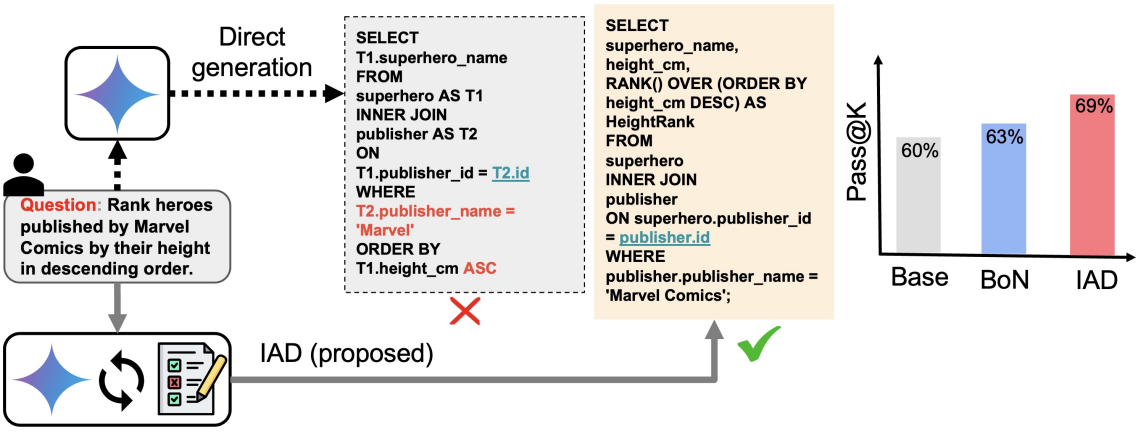


Figure 1: Qualitative (Left) and Quantitative (Right) illustration of the performance benefit of IAD (Ours) over Single-turn response generations using Gemini-1.5 (Base) Text2SQL task. IAD improves performance by correctly handling query logic and joins, improving the accuracy over baseline and BoN (Best-of-N).

patible with various underlying models (without requiring model-specific adaptations), (ii) easy to implement (requires no modification of the underlying models), and (iii) commercial API-friendly (Applicable to closed-source models that are only accessible via API. This leads us to an important question:

How can we improve the inference alignment of AI Agents with Black-box access for complex and structured tasks?

An immediate solution to the above question is to leverage Best-of-N (BoN) sampling and its variants (Beirami et al., 2024; Jinnai et al., 2024; Amini et al., 2024), as it is black-box compatible and logit-free, making it a natural choice for inference-time alignment. BoN selects the best response from multiple generated candidates using a verifier, improving response quality in structured tasks such as HTML generation and SQL query synthesis. However, despite its advantages, BoN operates in a single-shot manner, meaning it lacks iterative refinement or feedback integration. This limitation prevents it from progressively improving responses based on evaluation signals, reducing its effectiveness in complex, structured reasoning tasks. We summarize our contributions as follows. **(1)** We propose a critique-guided iterative decoding framework, where responses undergo iterative refinement through verifier-guided feedback, ensuring enhanced performance in black-box structured generation tasks.

(2) Comprehensive evaluations on three agentic tasks. Experimental results demonstrate that our approach achieves improvements in complex structured output generation tasks, over existing baselines while maintaining computational efficiency.

Remarks : While multi-turn iterative approaches exist, most focus on train-time methods (Qu et al., 2024; Zhou et al., 2024; Shani et al., 2024), making them unsuitable for black-box inference. Inference-time methods (Madaan et al., 2023; Chao et al., 2024; Mehrabi et al., 2024; Muennighoff et al., 2025) have explored iterative refinement in specific contexts like jailbreak, math, and QA, but primarily for simpler tasks rather than complex agentic settings requiring structured generations like HTML and SQL. Specifically (Chao et al., 2024; Mehrabi et al., 2024) iteratively refine prompts for adversarial attacks but lack direct comparisons with BoN, making performance gains less interpretable. Meanwhile, (Madaan et al., 2023) and its variants emphasize on LLM self-evaluation rather than focussing on the design choices for iterative refinement based approach for structured output generation.

2 AI Agent Alignment

In this section, we formalize the problem of inference-time alignment for AI agents operating in black-box settings, particularly in complex structured-generation tasks such as HTML and SQL synthesis in multi-modal environments like Sketch2Code, Text2SQL, and Webshop. Given a task distribution $\mathcal{T} \in \Delta$, we define the optimal but inaccessible agent policy $\pi^*(\cdot|x)$, which generates the ideal structured response $y^* \sim \pi^*(\cdot|x)$ for a given input prompt x. However, in the black-box setting, we only have access to a reference policy

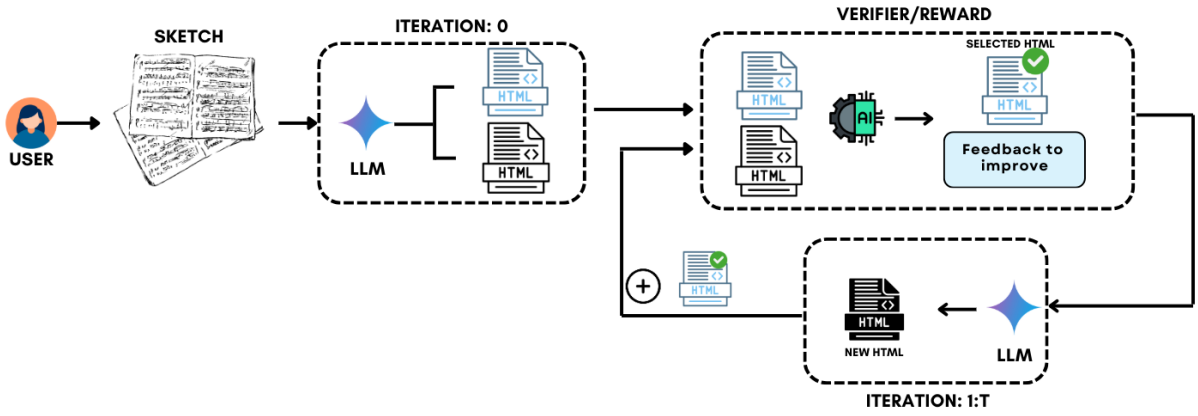


Figure 2: Demonstrates the high-level schematic of our proposed approach - Iterative Decoding of AI Agents

$\pi_0(\cdot|x)$, which lacks direct optimization capabilities due to constraints on model updates.

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 $\pi^*(\cdot|x)$. To formally quantify this alignment gap, we define the inference-time misalignment as an f-divergence between the optimal and reference policies as $d_f(\pi^*(\cdot|x), \pi_0(\cdot|x))$, where $d_f(\cdot,\cdot)$ measures how far the black-box agent deviates from the ideal structured output distribution. We explore methods to reduce this divergence using iterative inference-time strategies, which we detail in the following sections. We assume access to a verifier function $R(x, y)$, which evaluates the quality of a response y given the prompt x (further details in Appendix).

Task specific details: In Sketch2code, the reference policy $\pi_0(\cdot|x)$ generates structured HTML y given the prompt $x = [s, c]$ with wireframe sketch s, noisy text prompt c. The verifier $R(x, y)$ evaluates alignment with the target layout, textual coherence, and semantic correctness, ensuring that the generated HTML accurately reflects the sketch's structure. In Text2SQL, $\pi_0(\cdot|x)$ maps a natural language query c to an SQL query y conditioned on the database schema x. The verifier $R(x, y)$ assesses syntactic validity and execution correctness, identifying errors in joins, logical consistency, and query structure. Finally, in WebShop, the agent's actions are conditioned on the full interaction history up to the current time step, i.e $x_t = [x, y_{<t}]$, where x is the initial user query and $y_{<t}$ represents all previously generated product page interactions. Based on this history, the reference policy $\pi_0(\cdot|x_t)$ generates the next action a_t which involves searching or clicking and the corresponding product description.

2.1 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 sub-optimal 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 feed-

back, 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).

3 Proposed Methodology

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)**	**26.67**	**22.6**	**19.93**
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)**	**29.92**	**25.99**	**22.47**
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)*∗	**31.98**	**28.39**	**23.01**

Table 1: **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. Table demonstrate that IAD (Ours) consistently outperform SoTA baseline by >10% margin.

.

To address the limitations of the existing approaches, as discussed in the previous section, we propose an iterative procedure for inference time alignment of AI agents in this section. Before discussing the strategy, we first discuss our key insight which helped us to propose our approach.

Our key insight: Our objective is to generate a high-quality response $y^* \sim \pi^*(\cdot|x)$ for a given input x , while having only sampling access to the reference policy $\pi_0(\cdot|x)$. This practical limitation inherently constrains us via the f - divergence defined in equation. However, we observe that iteratively updating the input prompt; we can progressively shift the conditional sampling distribution toward the optimal response. Formally, if we denote each iteration as $t = 1$ to T, we can define an iterative mapping over multiple turns, $f(y_1, y_2, \ldots, y_T)$, such that the gap $d_f(\pi^*(\cdot|x), \pi_0(\cdot|x, f(y_1, y_2, \ldots, y_T)))$ is minimized. This iterative refinement process effectively guides the reference policy toward producing responses that increasingly align with the optimal distribution.

Proposed approach: We propose an iterative refinement approach for black-box AI agents, where response generation is guided by a verifier to progressively improve outputs.

Step 1: 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:

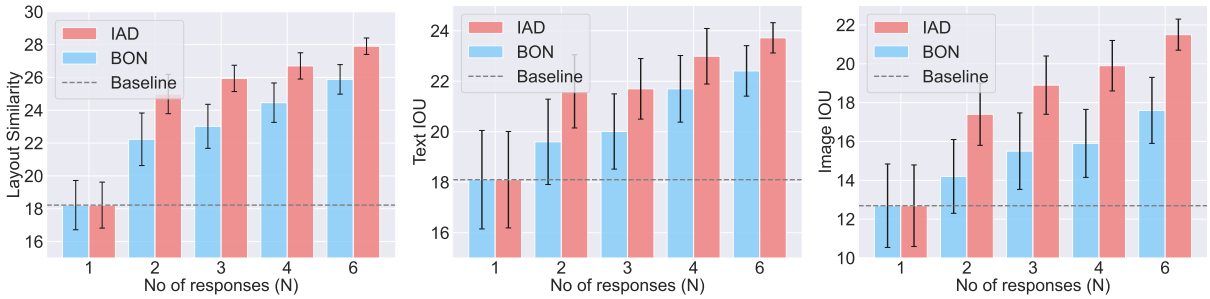
$$\hat{y}_{t+1} = \arg \max_{y \in (y_t, \hat{y}_t)} R(x, y) \tag{2}$$

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

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

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.

3. Acceptance Criterion : A new response y_{t+1} is accepted if it provides an improvement over the



(a) (b) (c)

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. Note that (b) and (c) are not directly optimized for this plot and test the generalization performance. This demonstrates the consistency in performance across models and metrics

previous best, $R(x, y_t) - R(x, \hat{y}_t) > \delta$ (where $\delta = 0$ is the special case)

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.

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.

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. This iterative refinement has been consistently observed in our experiments, where initial syntax errors in generated code progressively disappear through our approach.

4 Experimental Analysis

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.

4.1 Sketch2code Results and Analysis

Evaluation metrics and Baselines. We evaluate our approach against SoTA baselines using (i) Layout Similarity, (ii) Visual IoU, and (iii) Text IoU with reference HTML (Li et al., 2024b). These metrics strongly correlate ($\approx 90\%$) with human satisfaction (Li et al., 2024b) (details in Appendix). We use layout similarity as a verifier alongside LLM-as-judge (Li et al., 2024b) to refine the generations for both BoN (Beirami et al., 2024) and IAD. We compare our proposed approach IAD against single-turn SoTA models (GPT-4o, Claude-

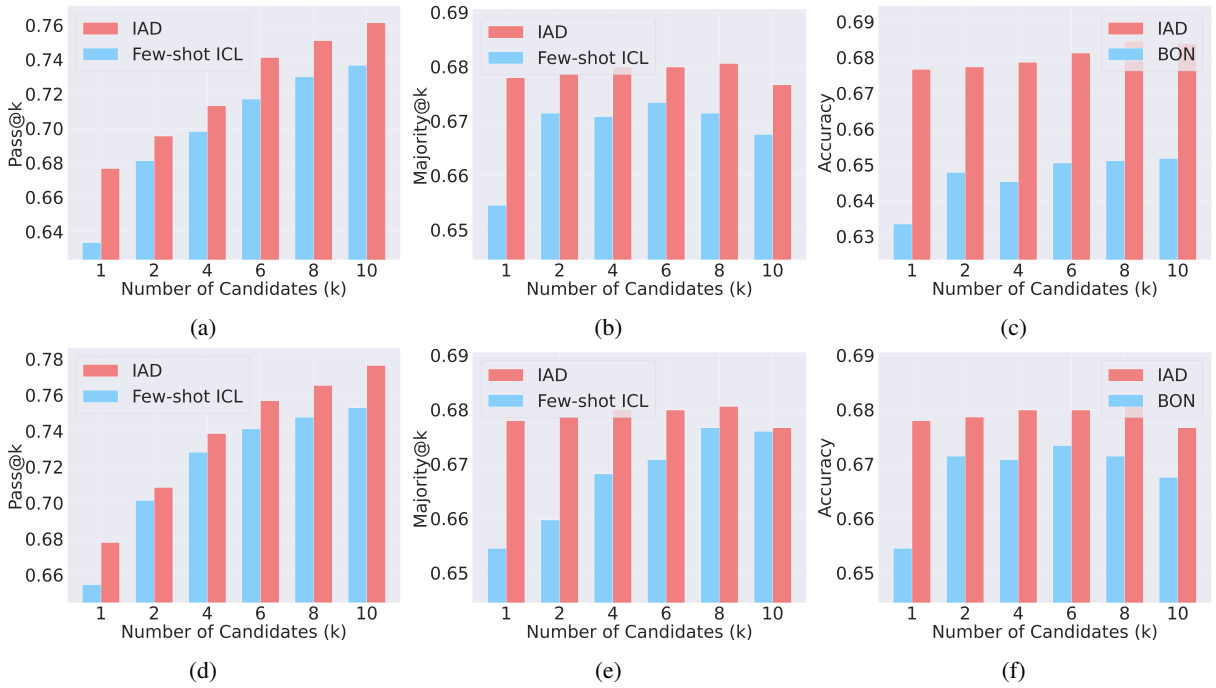


Figure 4: **Text2SQL Experiments**: (a) Pass@K performance comparison between queries generated using our approach and Few-shot ICL with Gemini-1.5-flash. (b) Majority@K performance comparison between queries generated using our approach and Few-shot ICL with Gemini-1.5-flash. (c) Accuracy comparison 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.

3, InternVL2, Gemini-1.5-Flash, CoT) and multi-response approaches (BoN, Sk2code). We compare the performance on the same evaluation setup and dataset as used in (Li et al., 2024b).

Initial results. Figure 9 and Table 1 qualitatively and quantitatively show that single-turn approaches, including SoTA models, fail to maintain the layout structure, block positions, text placement, and sizing, leading to poor alignment with the reference layout. These models score low across all three metrics. Multi-response methods significantly improve performance, even with N=2 samples. BoN, using the weaker Gemini-1.5-Flash, outperforms single-turn models. With N=4, BoN surpasses SoTA single-turn models by 15%, accurately capturing block positioning, title blocks, and overall layout. Performance monotonically improves with more responses, with layout scores rising from 20.41 to 25.7 as N increases to six.

Benefits of IAD (ours). BoN struggles with fine-grained layout details and often repeats positional errors across all N generations (Figure 1). In contrast, IAD iteratively refines responses, leading to a $\approx 15\%$ improvement over BoN and single-turn SoTA (Claude) in just two iterations (N=2), even with the simpler Gemini-1.5-Flash model (Table 1). At each iteration, IAD incorporates the best and worst HTML outputs as context, along with refinement instructions. This enables progressive learning of layout components and image semantics, significantly enhancing output quality. As iterations increase, IAD achieves a high layout score of 26.75, outperforming all baselines with the same number of generations. We also evaluate Image and Text IoU scores to check for reward over-optimization. However, as shown in Table 1 and Figure 3, text and image similarities also improve, confirming consistency in IAD's performance gains over BoN. Notably, with more generations, the performance gap between IAD and BoN narrows.

Ablation. We also consider the sensitivity of the token length of the context plays a critical role in this case, where providing the entire HTML can affect the entropy of the distribution, and thus, over-conditioning can hinder structured generation by reducing diversity and exploration. Thus, we pro-

vide only the top 300-400 tokens of the best (and worst) HTMLs. However, we remark that if there were a judge to highlight which portion of the code needs to be updated, that would 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 provides feedback on improvement and sometimes snippets of HTML as well (however, we restrict that to 200 tokens 5-6% of the original HTML). This leads to a significant improvement of 36% for the layout score with just two iterations and a final score of 31.98 with 6 iterations, demonstrating the importance 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 analyses, our findings remain consistent, where IAD outperforms baselines.

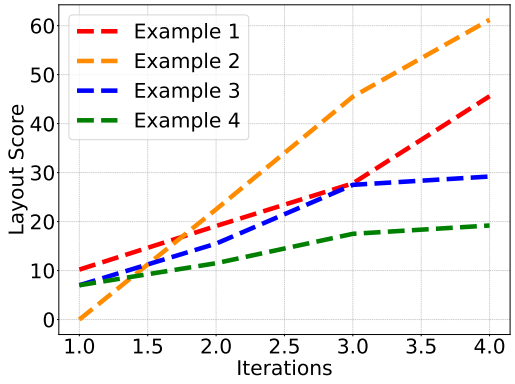


Figure 5: **Sketch2code** : Illustrates the improvement of our proposed approach over 4-turns w.r.t Layout similarity score for 4 examples. It shows that for most of the examples, there is a clear improvement in the scores over the iterations

4.1.1 Text-to-SQL Results and Analysis

In this section, we detail the experiments conducted on the BIRD text-to-SQL benchmark (Li et al., 2024a). Our evaluation involved two models, Gemini-1.5-pro and Gemini-1.5-flash, to test our proposed approach and compare it against baseline methods.

IAD vs test-time-compute methods: We compared our method to two baselines: few-shot chain-of-thought prompting and the best-of-N approach, both implemented using the same models. These comparisons highlight the effectiveness of our multi-turn strategy in narrowing the gap between the reference distribution and the optimal distribution. Our multi-turn approach addresses common issues in single-turn generation, which can produce SQL queries with syntax and semantic errors. By incorporating multiple rounds of response selection and feedback, our method effectively corrects these errors. Moreover in the multi-turn setting, the reward model benefits from observing query execution results, allowing it to more readily identify syntax and semantic issues. For the few-shot prompting baseline, we employed two metrics—Pass@K and Majority@K—to evaluate the quality of the generated candidate queries. As shown in Figure 4, our approach consistently produces candidate sets with higher correctness. In contrast, the best-of-N method aggregates results; therefore, we used the original execution accuracy for its evaluation. To compare the two, we computed accuracy across different values of N. For our method, we selected the best query among the N candidates using self-consistency (Wang et al., 2022)—executing the queries, grouping them by their execution results into buckets, and then choosing the query from the largest bucket as the most consistent answer, as illustrated in Figure 4.

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

IAD vs Text-to-SQL baselines We also compare our approach with several baseline methods in the text-to-SQL domain that do not rely on fine-tuning on the BIRD train set, ensuring a fair comparison. These baselines originate from different families of methods, for example, MCS-SQL (Lee et al., 2024) is a variant of the best-of-N method, DIN-SQL (Pourreza and Rafiei, 2024) and DAIL-SQL (Gao et al., 2023) employ human-engineered few-shot prompts to achieve higher performance, and MAC-SQL (Wang et al., 2023) is a multi-agent approach designed specifically for text-to-

SQL tasks. The results of these comparisons are provided in Table 2, where we achieved the highest execution accuracy by 3% margin compared to the second best method.

4.2 Webshop Results and Analysis

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
GPT-3.5-Turbo-16k	73	27
Text-Davinci-003	72	29
Gemini-1.5-Pro	73.5	29.3
BoN-SC + Gemini-1.5-Pro	74.12	30.31
DeepSeek-67b	72	31
GPT-3.5-Turbo	76	35
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 3: **Webshop**- Progress Rate (PR) and Success Rate (SR) for Models in the Webshop Environment(Yao et al., 2023). Perform a comparison of IAD (with two models GPT-4o and Gemini-1.5-Pro) against SoTA baselines for the Webshop Leaderboard with the evaluation similar to followed in Agentboard (Ma et al., 2024)

For all the experiments, we use greedy decoding with temperature=0.1, top p = 1.0 for our results. For BoN-SC, we try to vary the temperature to generate more diverse responses. We compare IAD* against SoTA baselines, including Lemur, Mistral, Vicuna, Gemini, and GPT models, in the Webshop environment, where agents interact with webpages through search and click actions. Performance is measured via progress rate (PR) and success rate (SR), with SR being the key metric for exact attribute satisfaction. We follow the same evaluation methodology as AgentBoard (Ma et al., 2024). IAD* consistently outperforms baselines, achieving over 25% improvement in SR across most models, with Gemini and GPT-4 variants performing best. To further enhance performance, we introduce feedback-based iterative refinement, where agents receive prompts like *"Reflect on your previous action and improve upon it"*, *"Improve the previous actions focusing on key-attributes"*, *"Follow the format to be action.name-'search' and action.params -'red tshirt....'"* for allowing LLMs to iteratively refine their decisions. A verifier-based LLM judge

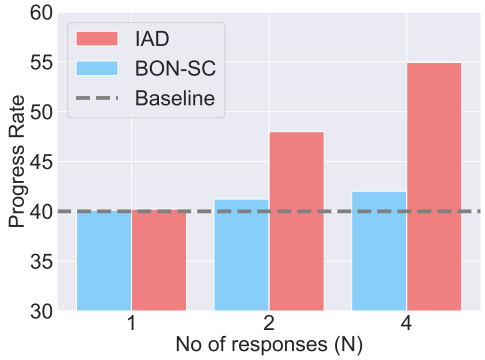


Figure 6: **Webshop** : Progress-rate comparison of IAD over Best of N with Self-consistency (BON-SC) and baseline on random selected 20% of the total evaluation set using weaker Gemini-1.5-Flash-01 (weaker model). Figure highlights the importance of iterative feedback while generation. Evaluation was done with a weaker model and on a subset of eval examples.

determines when to iterate or finalize actions. We also compare IAD* with BoN-SC (Best-of-N with Self-Consistency, N=4) but find that BoN provides limited improvement due to low diversity in generations, even with temperature = 0.6. Beyond this, higher temperatures introduce irrelevant attributes, reducing effectiveness. In contrast, IAD* improves search queries by refining unwanted attributes, enhancing focus on key parameters. Additionally, to show the effectiveness, we perform with a weaker version of Gemini-1.5-Flash-01 and perform the experiment, where we observe that it needs more iterations to actually follow the action format, without this iteration it fails to provide response in a meaningful format. On the other-hand, we approach the same with BoN-SC which couldn't improve much due to the issue in diverse response generation as highlighted before.

5 Conclusion

In this work, we proposed IAD : an iterative decoding approach for AI agent alignment with black-box access which highlights the effectiveness of iterative decoding (guided by a verifier) for these complex agentic tasks. IAD enhances prior baselines by dynamically integrating verification and critique mechanisms to iteratively refine and improve response quality. Experimental results across diverse domains show that our approach consistently outperforms several existing baselines including highly competitive BoN highlighting its ability to bridge the gap between reference and optimal policies at inference time.

6 Limitations

While IAD- our iterative decoding approach improves upon prior baselines by leveraging verifier feedback, it is inherently sequential, leading to increased computational overhead compared to easily parallizable BoN approaches. Addressing this trade-off between quality improvement and computational efficiency remains an important area for future research. Efficient optimization approaches, such as adaptive stopping, speculative decoding, and more efficient verifier-guided selection, could improve the efficiency in iterative decoding for agentic tasks. Additionally, verifier and 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.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. 2024. Jailbreaking black box large language models in twenty queries. *Preprint*, arXiv:2310.08419.

Anthony Costarelli, Mat Allen, Roman Hauksson, Grace Sodunke, Suhas Hariharan, Carlson Cheng, Wenjie Li, Joshua Clymer, and Arjun Yadav. 2024. Gamebench: Evaluating strategic reasoning abilities of llm agents. *Preprint*, arXiv:2406.06613.

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.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.

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*.

Xiang Huang, Sitao Cheng, Shanshan Huang, Jiayu Shen, Yong Xu, Chaoyun Zhang, and Yuzhong Qu. 2024. Queryagent: A reliable and efficient reasoning framework with environmental feedback-based self-correction. *Preprint*, arXiv:2403.11886.

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

Bowen Jiang, Yangxinyu Xie, Xiaomeng Wang, Yuan Yuan, Zhuoqun Hao, Xinyi Bai, Weijie J. Su, Camillo J. Taylor, and Tanwi Mallick. 2024. Towards rationality in language and multimodal agents: A survey. *Preprint*, arXiv:2406.00252.

Yuu Jinnai, Tetsuro Morimura, Kaito Ariu, and Kenshi Abe. 2024. Regularized best-of-n sampling to mitigate reward hacking for language model alignment. *Preprint*, arXiv:2404.01054.

Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. 2024. Args: Alignment as reward-guided search. *Preprint*, arXiv:2402.01694.

Megan Kinniment, Lucas Jun Koba Sato, Haoxing Du, Brian Goodrich, Max Hasin, Lawrence Chan, Luke Harold Miles, Tao R. Lin, Hjalmar Wijk, Joel Burget, Aaron Ho, Elizabeth Barnes, and Paul Christiano. 2024. Evaluating language-model agents on realistic autonomous tasks. *Preprint*, arXiv:2312.11671.

Dongjun Lee, Choongwon Park, Jaehyuk Kim, and Heesoo Park. 2024. Mcs-sql: Leveraging multiple prompts and multiple-choice selection for text-to-sql generation. *arXiv preprint arXiv:2405.07467*.

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.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2023. Agentbench: Evaluating llms as agents. *Preprint*, arXiv:2308.03688.

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.

Ninareh Mehrabi, Palash Goyal, Christophe Dupuy, Qian Hu, Shalini Ghosh, Richard Zemel, Kai-Wei Chang, Aram Galstyan, and Rahul Gupta. 2024. Flirt: Feedback loop in-context red teaming. *Preprint*, arXiv:2308.04265.

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.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *Preprint*, arXiv:2501.19393.

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.

Ajay Patel, Markus Hofmarcher, Claudiu Leoveanu-Condrei, Marius-Constantin Dinu, Chris Callison-Burch, and Sepp Hochreiter. 2024. Large language models can self-improve at web agent tasks. *Preprint*, arXiv:2405.20309.

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*.

Mohammadreza Pourreza and Davood Rafiei. 2024. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, 36.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. 2024. Recursive introspection: Teaching language model agents how to self-improve. *Preprint*, arXiv:2407.18219.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Lior Shani, Aviv Rosenberg, Asaf Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szpektor, Avinatan Hassidim, Yossi Matias, and Rémi Munos. 2024. Multi-turn reinforcement learning from preference human feedback. *Preprint*, arXiv:2405.14655.

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*.

Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. 2024. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench. *Preprint*, arXiv:2409.13373.

Mudit Verma, Siddhant Bhambri, and Subbarao Kambhampati. 2024. On the brittle foundations of react prompting for agentic large language models. *Preprint*, arXiv:2405.13966.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Qian-Wen Zhang, Zhao Yan, and Zhoujun Li. 2023. Mac-sql: Multi-agent collaboration for text-to-sql. *arXiv preprint arXiv:2312.11242*.

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.

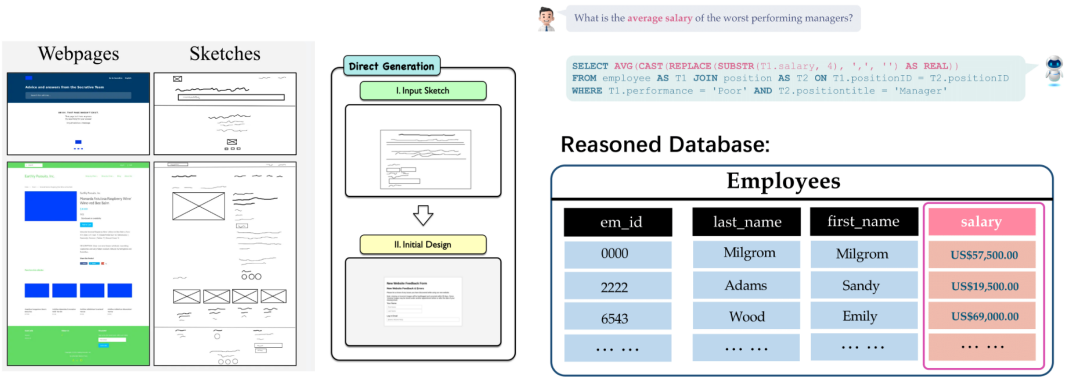
Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2023. Webshop: Towards scalable real-world web interaction with grounded language agents. *Preprint*, arXiv:2207.01206.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. 2024. Archer: Training language model agents via hierarchical multi-turn rl. *Preprint*, arXiv:2402.19446.

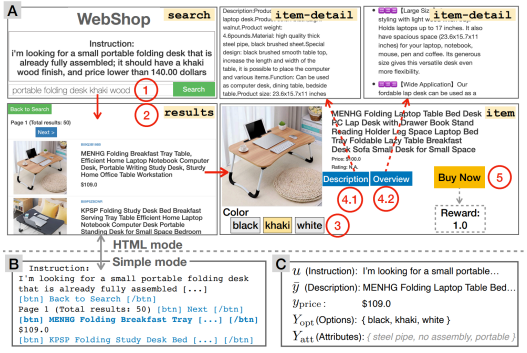
A Appendix

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 (b) Text-to-SQL Environment



(c) Webshop Environment

Figure 7: 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

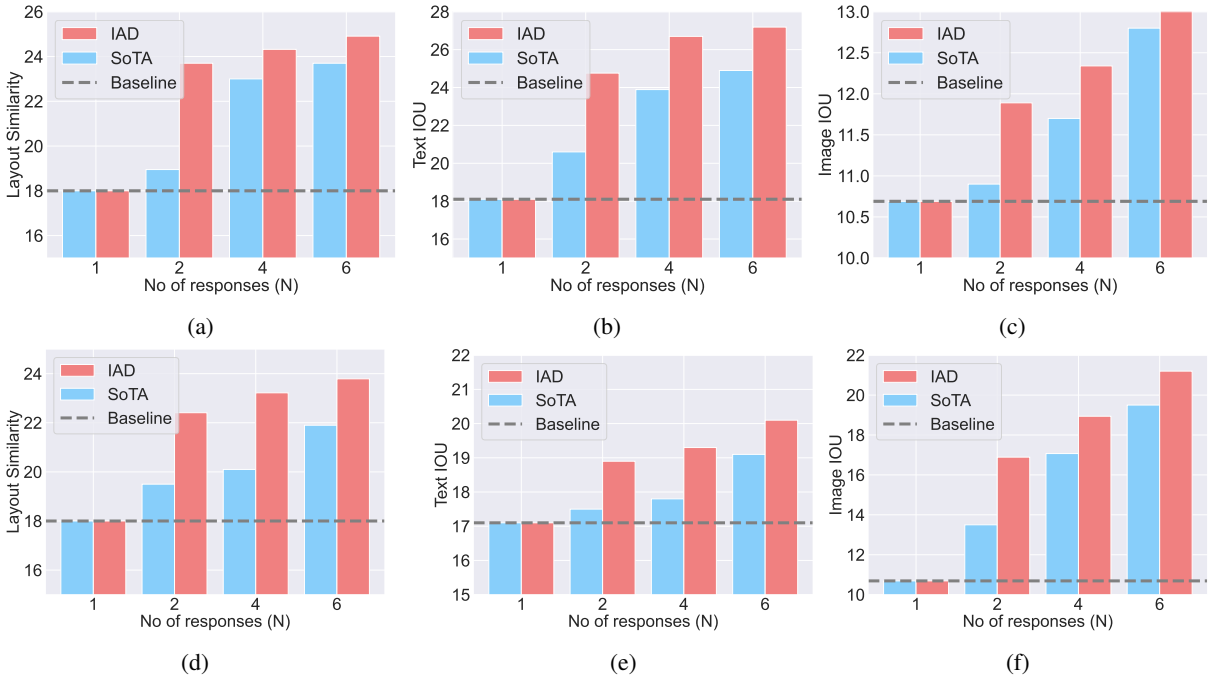


Figure 8: **Sketch2code**: This figure provides a comparison of IAD (ours) with Best-of-N sampling (SoTA) and single-turn generation with Gemini-1.5-Flash for the metrics - (a, d) Layout Similarity (b, e) TextIoU (c, f) ImageIoU across varying the number of generations (N). Top 3 rows, the optimization 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 $\hat{y}_T$

Initialize: Sample an initial response $y_0 \sim \pi_0(\cdot|x)$
Compute its reward $r_0 = R(x, y_0)$
Accept the initial response: $\hat{y}_0 \leftarrow y_0$ and $r^* \leftarrow r_0$
for $t = 1, 2, \ldots, T$ **do**
  Sample a new candidate response $y_t \sim \pi_0(\cdot|x, \hat{y}_{t-1})$
  Compute its reward $r_t = R(x, y_t)$
  if $r_t - r^* > \delta$ **then**
    Accept the candidate: $\hat{y}_t \leftarrow y_t$ and $r^* \leftarrow r_t$
  else
    Reject the candidate: $\hat{y}_t \leftarrow \hat{y}_{t-1}$
  end if
end for
return $\hat{y}_T$

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

A.2 Limitation of Single-turn Approach

In this section we characterize the performance gap Δ 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.

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

where $R(x, y)$ represents the reward function measuring the quality of the generated response, and $d_{\text{TV}}(\pi^*(\cdot|x), \pi_0(\cdot|x))$ 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 Δ 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 Motivation of our Proposed approach

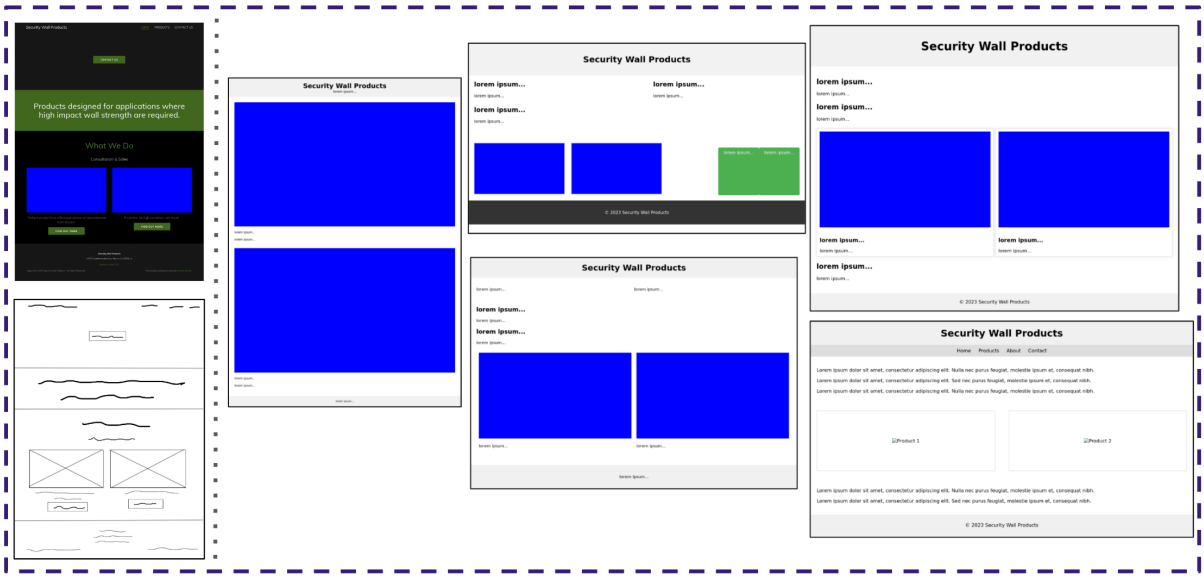
In this section, we provide an intuitive explanation of why our proposed approach works via connecting to the equivalence of the reward function and the log-probability of the optimal policy. In our proposed approach, at each turn t, the response y_t is sampled from the proposal distribution $\pi_0(\cdot|x, \hat{y}_{t-1})$ which refines the sampling process by updating the "context" of the generation based on the previously accepted response. Next, we estimate the mean of this updated distribution which can be given as :

$$\mu_t = \mathbb{E}_{y_t\sim\pi_0}\pi_0(y_t|x, \hat{y}_{t-1}) = g(x, \hat{y}_{t-1})$$

which shows that the updated policy mean is a function $g(x, \hat{y}_{t-1})$ of the best response till the current step. It is important to note that at any point, we are selecting the response y_t which has the reward $R(x, y)$. A keey insight from the alignment literature (Rafailov et al., 2023) lies in the fact , that there exists a direct equivalence between the reward function and the corresponding optimal policy under the reward as:

$$\log \pi^*(y|x) = -\log Z(x) + \frac{1}{\beta} R(x, y) \tag{4}$$

Thus the above equation highlights that maximizing the reward is equivalent to maximizing the log-probability of the optimal policy. Our acceptance criterion ensures a response y_t is retained only if it improves upon the previous best $\hat{y}_{t-1}$, thus enacts a step-wise refinement of the policy toward an optimal target policy $\pi^*(\cdot|x)$, where the verifier $R(x, y)$ plays the role of the unnormalized log probability of the optimal policy. Specifically, while comparing the responses y_1, y_2 using the reward function $R(x, y)$, we estimate $R(x, y_2) - R(x, y_1)$ which is equivalent to evaluating $\log \frac{\pi^*(y_2|x)}{\pi^*(y_1|x)}$. Thus, the acceptance/rejection mechanism ensures that the refinement process is guided toward responses with higher rewards, progressively reducing the gap between the proposal distribution and the target distribution. This is an intuitive explanation of our proposed approach and providing a rigorous connection and convergence analysis of our algorithm remains a valid scope of future research. Also this highlights the importance of an accurate verifier, since an erroneous verifier might bias the proposed approach and a detailed investigation of the same is a valid direction of future research.



(a)

Figure 9: **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 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

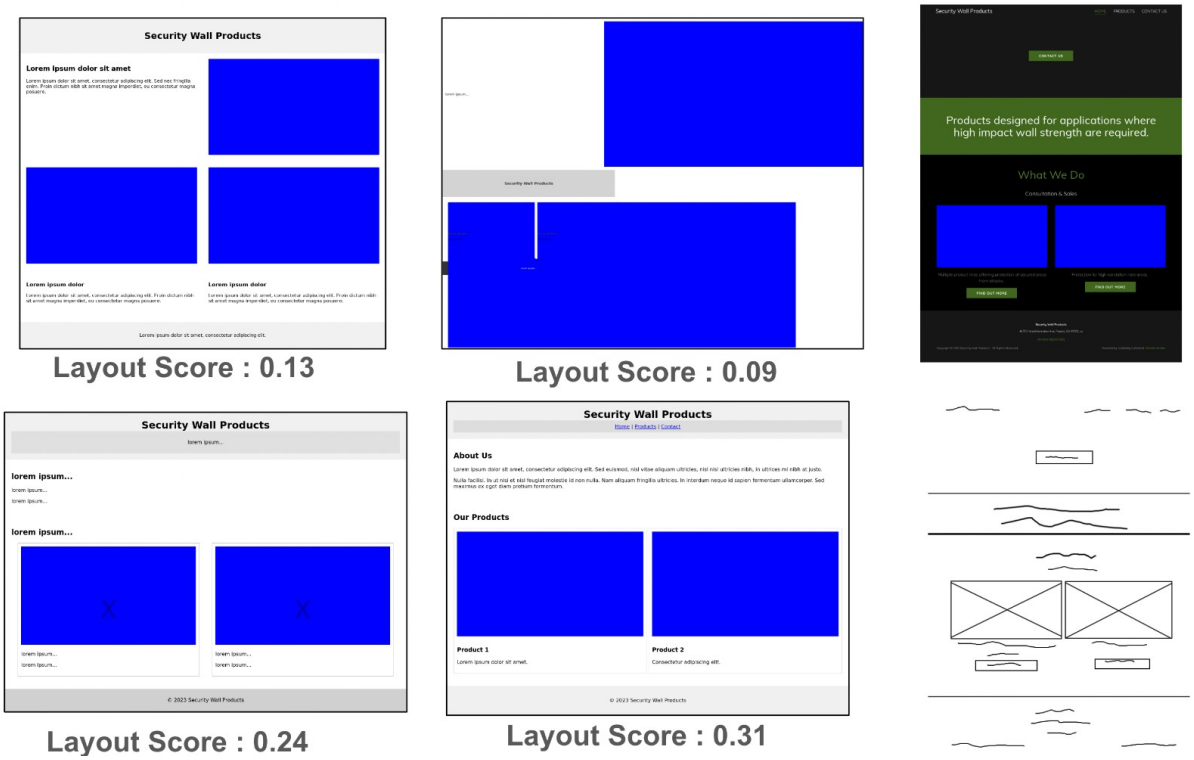


Figure 10: **Sketch2code**: Provides a qualitative verification of layout score as a metric and corresponding correlation to human judgement. It is evident that HTMLs with higher match with the reference layout (right-top) and user sketch(right-bottom) has higher layout score and vice-versa showing that its a valid metric.

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 2, 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.

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.

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.

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 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%

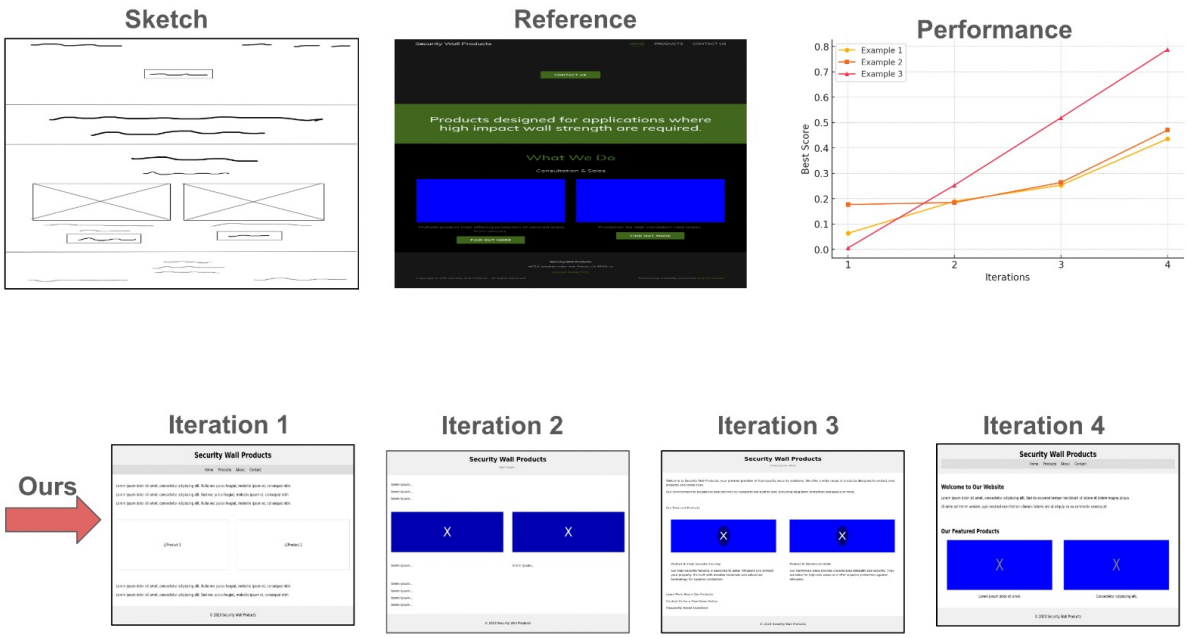


Figure 11: **Sketch2code** : Top row shows the user sketch, reference image and the performance of IAD over iterations. The figure highlights improvement of IAD over 4-turns w.r.t Layout similarity score (1/100) for 3 examples. It shows clear improvement over iterations. We also qualitatively analyse 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.

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 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.

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.

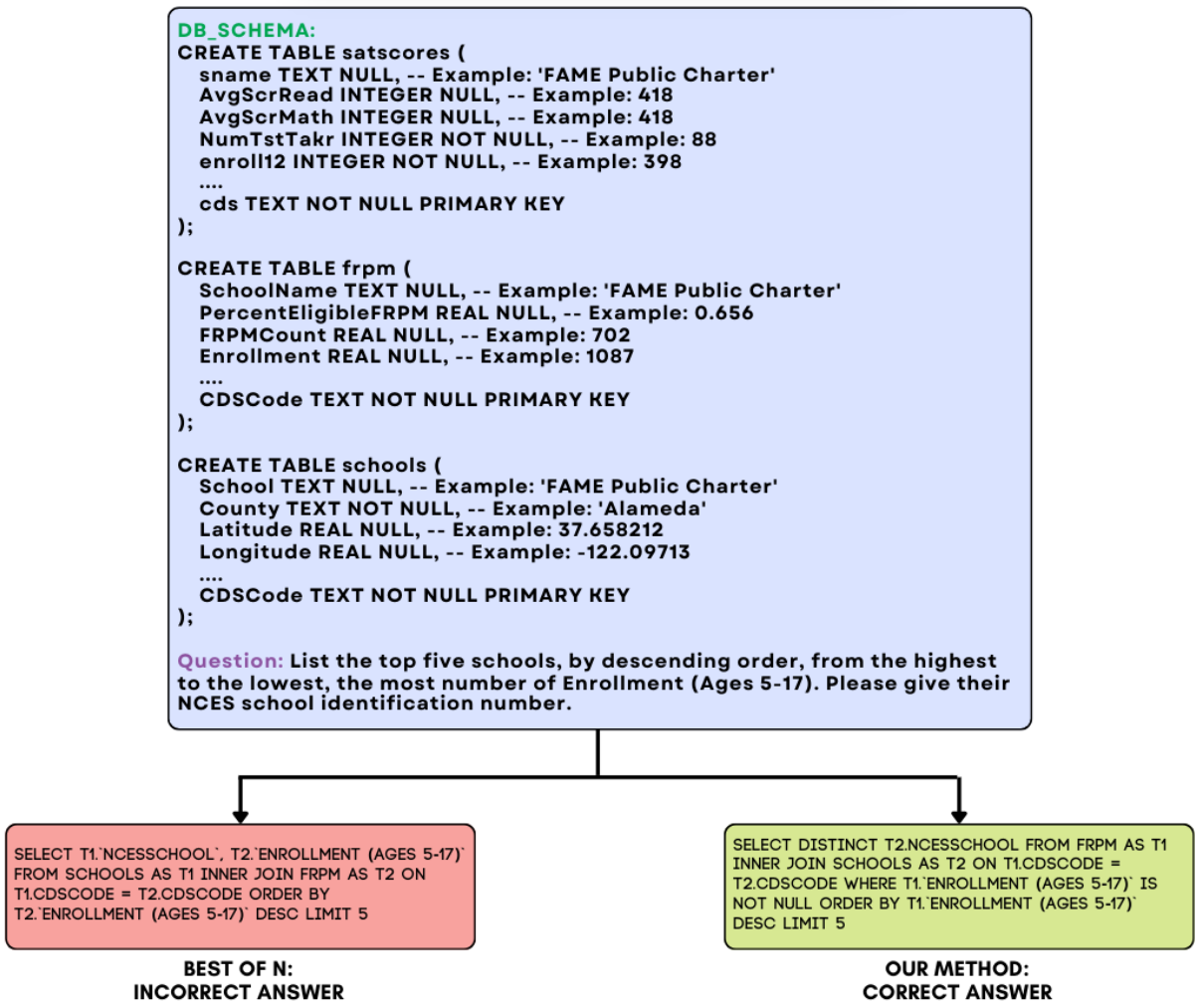


Figure 12: **Text2SQL**: An example of two responses is presented: the first response, generated using our proposed approach, is correct, while the second response, produced using the best-of-N method, is incorrect.

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)**

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 4: **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 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**

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**