TableHallu: A Benchmark for Uncovering Hallucinations in Query-Driven Table Generation

Anonymous ACL submission

Abstract

While Large Language Models (LLMs) excel at processing unstructured text, their reliability falters in structured data generation, leading to a critical issue we term *table hallucination*. Existing benchmarks, reliant on monolithic accuracy scores, fail to diagnose the specific ways models err. To address this, we introduce a systematic framework for understanding and evaluating this problem. Our contributions are threefold. **First**, we provide a formal definition and a comprehensive taxonomy of table hallucinations. **Second**, based on this taxonomy, we construct **TableHallu**, the first diagnostic benchmark for this task. TableHallu is built using a novel, scalable pipeline that programmatically injects distractors to create challenging test cases. This automated process, which eliminates the need for costly manual annotation, is proven to be over **95% accurate under human verification**. **Third**, we conduct a comprehensive evaluation of state-of-the-art LLMs on TableHallu. The results reveal alarming and previously obscured vulnerabilities: models universally struggle with ordering constraints, frequently invent non-existent entities or attributes, and fail at elementary arithmetic during table generation. Our work provides the first systematic analysis of table hallucinations and a robust benchmark to steer future research from pursuing simple accuracy towards achieving verifiable, multi-faceted reliability. *Code and data will be available.*

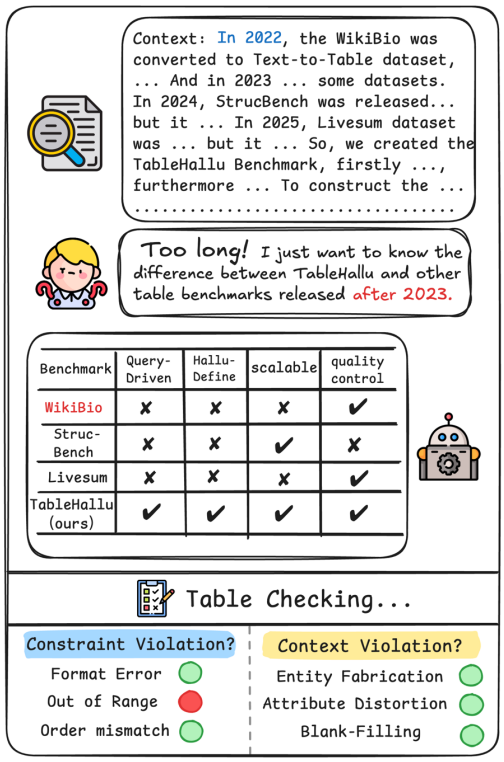

Figure 1: An example of **TableHallu**. The user query requires a quick comparison of our proposed TableHallu benchmark with other table benchmarks created after 2023. The LLM generates a table for efficient information acquisition. However, the table includes the WikiBio, which the context states was converted into a table dataset in 2022, thus violating the time constraint in the user's query.

1 Introduction

The ability of Large Language Models (LLMs) to transform unstructured text into structured tables is a cornerstone of their real-world utility. However, this powerful capability is plagued by a critical and under-examined failure mode: hallucination (Ji et al., 2023). In high-stakes domains such as finance, where an LLM might generate a financial summary with fabricated figures (Khan and Umer, 2024), or in healthcare, where it could produce a patient data table with distorted medical facts (Thirunavukarasu et al., 2023), such hallucinations are not merely inaccuracies—they are critical risks that undermine the reliability of these systems. This paper confronts this challenge directly within the paradigm of **Query-Driven Table Generation**.

We define Query-Driven Table Generation as the task of synthesizing a table that strictly adheres

to the constraints specified in a user's query while accurately reflecting the information present in a source context. This paradigm imposes a dual obligation on the model: linguistic faithfulness to the source and structural adherence to the query. Drawing inspiration from the dichotomy of *faithfulness* versus *instruction following* in traditional hallucination research (Zhang et al., 2025; Huang et al., 2023), and integrating the structural definitions of *Entities* and *Attributes* from database theory (Silberschatz et al., 2002; Yu et al., 2018), we propose a grounded taxonomy of table hallucinations. Specifically, we categorize these failures into two primary forms: (1) **Constraint Violation**, where the model fails to adhere to the explicit schema or logic required by the query (e.g., omitting a required column or ignoring a date range filter), and (2) **Context Violation**, where the model fabricates or distorts the entity values relative to the source text (e.g., hallucinating non-existent data points), as illustrated in Figure 1.

Despite its importance, this specific intersection of structural and content fidelity remains largely unaddressed by existing benchmarks. Datasets like **Struc-Bench** (Tang et al., 2023) or **Livesum** (Deng et al., 2024) primarily focus on information extraction with fixed or implicit queries (e.g., "summarize the context"), lacking the complexity of diverse, user-specified constraints. More critically, they fail to simulate the "needle-in-a-haystack" challenge common in real-world scenarios, where models must filter relevant signals from a sea of plausible but irrelevant "distractor" information. Without such noise, it is impossible to rigorously test whether a model is strictly adhering to query constraints or merely hallucinating based on peripheral context. Consequently, there is an urgent need for a dedicated benchmark to diagnose, quantify, and ultimately mitigate these nuanced failures.

To bridge this gap, we introduce **TableHallu**, the first diagnostic benchmark focused on table hallucinations in query-driven generation. Through it, we conduct a comprehensive evaluation of state-of-the-art LLMs and propose a foundational baseline. Our core contributions are as follows:

- **We introduce a systematic definition and taxonomy of table hallucinations grounded in both linguistic faithfulness and database structure.** Based on this taxonomy, we construct **TableHallu**, a benchmark specifically designed to probe the vulnerabilities of LLMs in query-driven structured output generation.
- **We propose a novel, scalable pipeline for automated data generation that significantly reduces reliance on expensive manual annotation.** This pipeline programmatically injects distractors to create challenging cases from scratch, producing high-fidelity data that achieves **over 95% accuracy under human verification**.
- **We conduct a comprehensive evaluation of state-of-the-art LLMs on TableHallu, providing the first systematic analysis of the factors contributing to table hallucinations.** Our study investigates the influence of the model's thinking mode, context length, and table formats, while also comparing the efficacy of various strategies to establish a foundational baseline for future research.

2 Related Work

2.1 Hallucination in Large Language Models

Hallucinations in LLMs are broadly categorized into *factuality* errors (contradicting world knowledge) and *faithfulness* errors (diverging from source context) (Ji et al., 2023; Rawte et al., 2023; Malin et al., 2024; Huang et al., 2025). While factuality is widely benchmarked (Lin et al., 2021; Min et al., 2023), our work targets faithfulness, which is paramount for conditional tasks including summarization (Maynez et al., 2020), dialogue (Dziri et al., 2022), and structured data generation (Huang et al., 2023). Faithfulness failures further split into *instruction adherence* (ignoring directives) and *context adherence* (ignoring source text) (Huang et al., 2025). Although general benchmarks like HaluEval (Li et al., 2023) exist, they lack the granularity to diagnose the complex, structural error patterns inherent in query-driven table generation, necessitating a specialized taxonomy.

2.2 Text-to-Table Generation

Text-to-Table (T2T) generation (Wu et al., 2021) has evolved from reversing data-to-text datasets (Wiseman et al., 2017; Lebret et al., 2016) to handling complex formats like HTML and JSON (Tang et al., 2023; Yang et al., 2025b). Recent works have further pushed boundaries in multi-source integration and coherence (Deng et al., 2024; Upadhyay et al., 2025; Anvekar et al., 2025). However, existing benchmarks suffer from two critical limitations

regarding hallucination evaluation. **First**, they predominantly rely on static extraction paradigms, neglecting the **dynamic, query-driven constraints** essential for personalized applications (Liu et al., 2024b). **Second**, they utilize monolithic accuracy metrics without **adversarial stress testing** (e.g., distractor injection), making it difficult to distinguish robust reasoning from rote copying. **TableHallu** addresses these gaps by introducing a fine-grained taxonomy and a scalable pipeline for generating adversarial, query-driven test cases.

3 The *TableHallu* Benchmark: A Stress Test for Query-Driven Table Generation

3.1 Formal Problem Definition

As illustrated in Figure 1, the query-driven text-to-table task is defined as follows: given a context C (a long-form text) and a query Q, a model is required to generate a table T. The query Q specifies a set of constraints $\{q_1, q_2, \ldots, q_n\}$, which can encompass various conditions such as temporal limitations (e.g., "after 2023"), categorical requirements (e.g., "text-to-table benchmarks"), and sorting orders (e.g., "sorted by year in ascending order"). More cases can be found in Appendix E

A generated table T is considered non-hallucinatory if and only if it satisfies two fundamental properties:

1. **Constraint Adherence**: The table must strictly comply with all constraints specified in the query. Formally, $\forall q_i \in Q, T \models q_i$.

2. **Context Grounding**: All information presented in the table must be verifiably grounded in the source context. Formally, $\text{Content}(T) \subseteq \text{Content}(C)$.

Failure to meet these properties results in Table Hallucination, which we categorize into a fine-grained taxonomy to enable precise diagnostics.

- **Constraint-Violation Hallucinations**: Violations of Constraint Adherence.
  - *Out-of-Range*: An entity in the table falls outside the scope defined by a query constraint (e.g., a 2022 benchmark appearing when the query asks for post-2023).
  - *Order Mismatch*: The table rows are not sorted according to the specified order.
  - *Format Error*: The output fails to conform to the requested format (e.g., Markdown syntax errors).
- **Context-Violation Hallucinations**: Violations of Context Grounding.
  - *Entity Fabrication*: An entity appears in the table but is absent from the context.
  - *Attribute Distortion*: An entity's attribute in the table contradicts the context.
  - *Blank-Filling*: A cell that should be empty (never mentioned in the context) is populated with fabricated information. While a subset of *Attribute Distortion*, we isolate it due to its high frequency and its origin in the model's propensity for fluent, structured completion.

3.2 The *TableHallu* Pipeline: Principled Construction of Adversarial Data

Our benchmark is constructed entirely through a novel, LLM-driven automated pipeline. This approach is designed to preemptively address the critical question of benchmark fidelity—"**How can an LLM, itself prone to hallucination, be used to create a reliable benchmark?**"—by developing the *TableHallu* Pipeline. Our solution is not a simple generate-then-verify paradigm, but a novel methodology founded on a two-fold insight designed to maximize adversarial complexity while ensuring ground-truth fidelity.

First, we employ a **decoupled generation process**. Instead of attempting to generate a complex context and its corresponding ground-truth table simultaneously—a process fraught with the risk of co-hallucination—we first construct a pristine, minimal "ground-truth kernel" (Table, Context, Query). We then exclusively perturb the context by injecting adversarial distractors, ensuring the original ground-truth table and query remain untouched and verifiable. This decoupling strategy radically reduces the possibility of generating flawed ground-truth data from the outset.

Second, this process is safeguarded by a **multi-stage verification protocol**. We verify the integrity of the initial "ground-truth kernel" *before* perturbation, and then critically re-verify the final adversarial context *after* perturbation. This dual-checkpoint system systematically eliminates errors at each creation stage. The success of this principled methodology is demonstrated by our final manual audit, which confirmed a **data accuracy of over 95%**, validating our pipeline as a scalable and reliable solution for generating high-quality adversarial benchmarks.

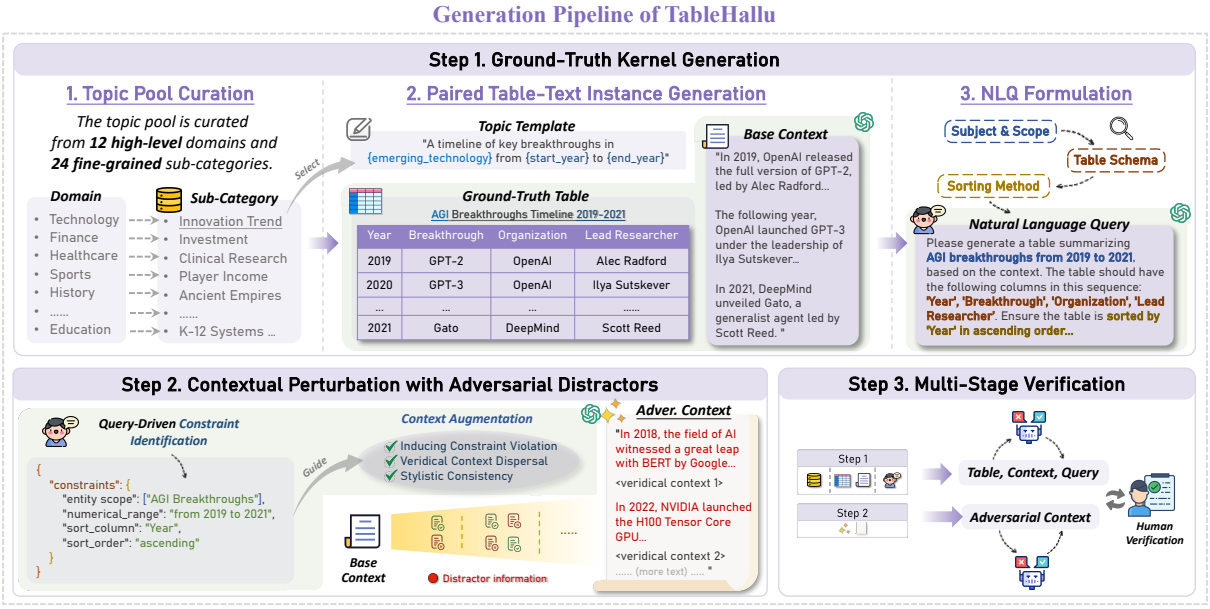

Figure 2: An illustration of the TableHallu Generation Pipeline. The pipeline systematically constructs adversarial test cases through a three-stage process designed to detect and diagnose hallucinations of LLMs in query-driven table generation tasks.

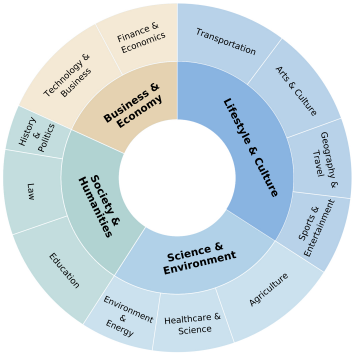

Figure 3: Hierarchical Data Distribution across 4 high-level Categories and 12 Domains.

Stage 1: Ground-Truth Kernel Generation. To ensure diversity, we curated a topic pool spanning 12 domains and 24 fine-grained sub-categories, each containing numerous topics. Our generation process for each sample begins by prompting an LLM with a randomly selected topic to generate a structured table. This table then serves as a factual seed for the LLM to produce a perfectly aligned base context. Finally, the LLM creates a complex, multi-constraint query derived from both the table and the context. This stage yields a pristine (Table, Context, Query) triplet, establishing the unimpeachable "ground-truth kernel" where all facts and constraints are internally consistent.

Stage 2: Adversarial Context Perturbation. This is the core of our adversarial design. The objective is to inject highly relevant distractors that are inconsistent with the query's constraints into the context, without corrupting the original ground truth. We first parse the query Q to extract its set of constraints $\{q_i\}$. These constraints define the precise boundaries for our "counter-constraint" distractors. For example, if a constraint is "in 2023," the pipeline instructs the LLM to weave plausible narratives involving similar events in "2022" or "2024" into the base context. This surgical injection creates a highly adversarial context that forces the model to perform fine-grained reasoning and information filtering, rather than simple pattern matching.

Stage 3: The Multi-Stage Verification Protocol. To guarantee the pipeline's integrity, we implement the rigorous, two-tiered automated verification process introduced in our methodology. First, an LLM-based verifier confirms the consistency and format correctness of the initial ground-truth kernel. This is the crucial first checkpoint. Second, after noise injection, another verifier performs the second-checkpoint validation, ensuring that the augmented context (a) retains all information from the original base context and (b) does not accidentally introduce new, valid information that satisfies the query. As a final, definitive validation of our entire automated pipeline, we employed 20 expert English-speaking annotators to manually audit the data. Their cross-verification confirmed the

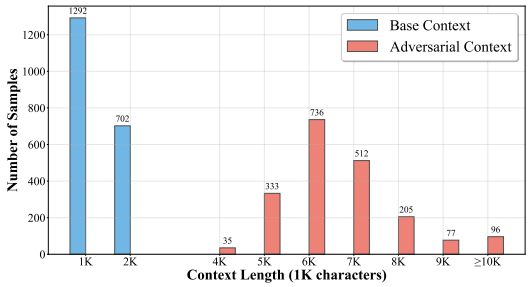

Figure 4: Distribution of Context Lengths for Base and Adversarial Samples.

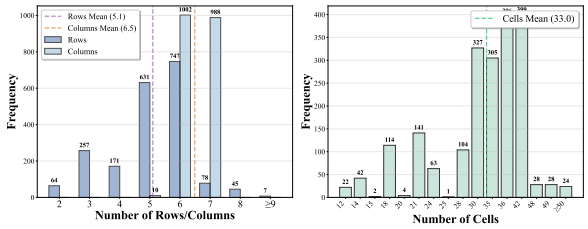

Figure 5: Distribution of table tows, columns and cells for TableHallu.

integrity of the triplets and the adversarial effectiveness of the noise, yielding the aforementioned data accuracy of over 95%. This result powerfully validates our methodology, demonstrating its ability to produce complex, high-quality evaluation data at scale, significantly mitigating the cost and time associated with traditional manual annotation. Further details on human evaluation can be found in Appendix B.

3.3 Benchmark Statistics and Characteristics

Our pipeline produced a dataset of 2,000 samples for evaluation and an additional 1,000 for model training. The data distribution is meticulously designed to ensure comprehensive and unbiased evaluation.

Topic Diversity (Figure 3.): Our dataset is meticulously constructed to encompass 12 distinct domains across 4 high-level categories, ensuring comprehensive coverage of diverse real-life scenarios. As depicted in Figure 3, the data is well-balanced across these domains, exhibiting a near-uniform distribution.

Context Complexity (Figure 4.): The presence of adversarial distractors creates a bimodal distribution in context length, as illustrated in Figure 4. Base contexts are concise, typically ranging from 1,000 to 2,000 characters. In contrast, adversarial contexts, after being augmented with distractors, become substantially longer, predominantly spanning 4,000 to 10,000 characters.

Table Structure (Figure 5): Tables typically contain 5-7 columns and 5-6 rows, averaging around 35 cells. This deliberately larger "surface area" compared to benchmarks like Struc-Bench increases the likelihood of exposing various hallucination types within a single sample.

3.4 Evaluation Framework: Detection and Diagnostics

To move beyond a monolithic "accuracy" score, we propose a diagnostic framework that first detects and classifies hallucinations, then quantifies them using fine-grained metrics.

Table Hallucination Detection. Our detection logic classifies hallucinations through a hierarchical series of targeted checks. We first identify *Entity Fabrication* and *Out-of-Range* errors by aligning primary keys against the context and query constraints. Crucially, to accommodate the generative variability of LLMs, this alignment is not limited to strict string matching; it incorporates type-aware tolerance mechanisms to normalize format discrepancies in specific domains (e.g., dates, monetary values) and employs semantic similarity as a final safeguard to capture linguistic paraphrasing. Subsequently, *Attribute Distortion* and *Blank-Filling* are detected by comparing the remaining cell values to the ground truth. Finally, we verify the row sequence to detect *Order Mismatch* and categorize structural anomalies, such as parsing failures or incorrect headers, as *Format Errors*.

Diagnostic Metrics Based on the detection results, we quantify the prevalence of each hallucination type by calculating its sample-level hallucination rate. This approach allows for a clear, high-level analysis of a model's primary failure modes. For a given hallucination type H, the Hallucination Rate(HR) is defined as:

$$\mathrm{HR}(H) = \frac{\mathrm{Count(Samples\ exhibiting\ H)}}{\mathrm{Count(Total\ Samples)}} \quad (1)$$

Furthermore, we also quantify the *TotalHallu(%)*, representing the percentage of generated responses that exhibit any form of hallucination. This framework transforms evaluation from a simple pass/fail judgment into a deep diagnostic tool, pinpointing the specific categories where a model struggles. More details can be found in Appendix C.

Type	Model Name	Constraint-Violation			Context-Violation			TotalHallu(%) ↓
		Format Error(%) ↓	Range Hallu(%) ↓	Order Mismatch(%) ↓	Attribute Hallu(%) ↓	Blank Filling(%) ↓	Entity Fab.(%) ↓	
Closed-Source	GPT-4.1	0.4	11.85	10	36.05	29.5	2.55	47
	GPT-4o	0.05	9.25	10.25	43.75	36.7	0.9	52.45
	o3-mini	1.6	5.8	2.75	**23.95**	**15.25**	1.65	**31.35**
Open-Source	DeepSeek-R1	0.15	5.55	4.4	28.25	20.15	**0.7**	33.7
	DeepSeek-V3	**0**	14.25	11.55	39.95	32.45	1.4	51.15
	Gemma-3-27B	16.55	24.55	23.4	30.3	20.8	2.55	65.95
	Llama-3.1-8B-Instruct	7.6	36.65	31.6	43.15	30.2	4.45	73.45
	Llama-3.3-70B-Instruct	0.1	26.45	18.65	29.95	18.85	2.45	53.25
	Phi-4	31.05	17.8	22.3	30.25	21.95	3.1	74.4
	Qwen2.5-72B-Instruct	0.05	24.6	23.9	63	56.25	2.3	76.1
	Qwen2.5-7B-Instruct	0.85	39	45.1	75.7	66.35	7.25	90.4
	Qwen3-235B-A22B	2.6	**5.15**	**2.65**	44.2	38.35	**0.7**	51
	Qwen3-32B	0.1	8.9	6.35	47.8	41.15	1.3	54.65
	Avg.	4.7	17.68	16.38	41.25	32.92	2.41	58.07

Table 1: A comprehensive evaluation of Query-Driven Table Hallucinations in various LLMs. Lower percentages (indicated by ↓) indicate better performance. For each column, the best result is in **bold** and the second-best is underlined.

4 Experiments

To thoroughly assess model capabilities on the TableHallu benchmark, we performed extensive experiments on a diverse set of 15 leading Large Language Models (LLMs). This set includes 12 open-source models and 3 closed-source models, with model sizes ranging from 7B to 671B. Specifically, the models are: Qwen2.5 (7B, 72B) (Qwen et al., 2025), Qwen3 (8B, 14B, 32B) (Yang et al., 2025a), the Mixture-of-Experts (MoE) (Mu and Lin, 2025) model Qwen3-235B-A22B, Phi-4 (Abdin et al., 2024), Llama-3.1-8B-Instruct, Llama-3.3-70B-Instruct (Dubey et al., 2024), Gemma-3-27B (Team et al., 2025), DeepSeek-V3 (Liu et al., 2024a), DeepSeek-R1 (Guo et al., 2025), o3-mini (OpenAI, 2025b), GPT-4o(Hurst et al., 2024), and GPT-4.1 (OpenAI, 2025a). Additional details on the experimental setup and implementation can be found in the supplementary material.

4.1 Main Results

Table 1 presents the main results of our evaluation on the TableHallu benchmark. The findings show that table hallucination is a widespread and severe issue, affecting all evaluated LLMs. The *TotalHallu* rate is high across the board, demonstrating the task's difficulty.

A deeper analysis reveals specific, shared weaknesses. Models universally struggle with ordering constraints (*Order Mismatch*), a task where even top performers like o3-mini and Qwen3-235B-A22B show error rates of 2.75% and 2.65%, respectively. For many others, like Llama-3.3-70B-Instruct, this error is far more pronounced (18.65%). Similarly, *Context-Violation* errors are common, with *Attribute Hallucination* and *Blank Filling* being the primary contributors. These two error types reveal a strong tendency for models to invent data, as seen in GPT-4o (43.75% and 36.7%) and Qwen2.5-72B-Instruct (63% and 56.25%).

A clear trend emerges among the top-performing models: the presence of an explicit reasoning or "thinking" step. The leading models in our benchmark—o3-mini, DeepSeek-R1, and Qwen3—all incorporate mechanisms that allow them to process and plan before generating a final answer. This suggests that the ability to internally decompose the query, cross-reference constraints, and structure the output in a preliminary step is critical for navigating the adversarial challenges of this task.

Performance among open-source models varies widely. DeepSeek-R1 stands out as the top open-source performer with a *TotalHallu* of 33.7%, nearly matching the best proprietary model. In contrast, models like Llama-3.1-8B-Instruct (73.45%) and Qwen2.5-7B-Instruct (90.4%) struggle significantly. Some models also fail at the basic task of generating a well-formed table, Gemma-3-27B and Phi-4, for example, have high *Format Error* rates of 16.55% and 31.05%, respectively.

4.2 Impact of Reasoning Mode and Model Scale

We performed an ablation study to analyze the effects of a reasoning step ("Thinking Mode") and model scale. As shown in Table 2, activating Thinking Mode dramatically reduces hallucinations, especially in adhering to query constraints. This con-

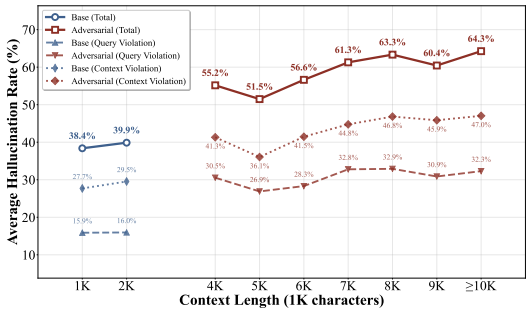

Figure 6: Effect of context type and length on the average hallucination rate across all models.

Model Name	Thinking Mode	Constraint-Violation(%) ↓	Context-Violation(%) ↓	Total Hallu(%) ↓
Qwen3-8B	On	**15.35**	**53.05**	**59.1**
	Off	53.15	76.45	88.15
Qwen3-14B	On	**14**	**43.2**	**50.5**
	Off	48.7	72.95	85.5
Qwen3-32B	On	**13.55**	**48.45**	**54.65**
	Off	42.5	54.5	72.9

Table 2: Ablation study on the effect of "Thinking Mode" on hallucination rates for various Qwen3 models. Lower percentages (indicated by ↓) are better.

firms that architectural approaches focused on reasoning are a key direction for improving reliability.

Crucially, our results also reveal that model scale is not a panacea. When Thinking Mode is enabled, performance does not scale monotonically with size, as the 14B model outperforms the 32B version. This finding, uncovered by our benchmark, challenges the prevailing "bigger is better" assumption and underscores the need to move beyond simple accuracy towards evaluating the multi-faceted reliability of models.

4.3 Impact of Context Length and Distractors

We analyzed the effect of context length on model reliability, testing on both our base context (Base) and our purpose-built distractor-injected context (Adversarial). The results, depicted in Figure 6, reveal a critical vulnerability of modern LLMs.

First, we observe a clear trend: as the context length increases, the average hallucination rate rises. This holds true for both base and adversarial settings, indicating that even models with long-context capabilities struggle with the "lost in the middle" problem (Liu et al., 2023) in this task. The model's ability to retrieve information accurately degrades as the input document grows.

Second, the results powerfully validate our benchmark's design. The presence of programmatically injected distractors in the adversarial context

Model Name	Table Format	Constraint-Violation(%) ↓	Context-Violation(%) ↓	Total Hallu(%) ↓
o3-mini	Markdown	9.25	30.25	**37**
	Json	**7.25**	**28**	34
	Tex	17	34.25	41.5
Llama-3.3-70B-Instruct	Markdown	42.5	32.75	60
	Json	**20.75**	37.5	**50**
	Tex	30	**31.5**	51.5
Qwen3-32B	Markdown	12.75	**54.5**	**60.75**
	Json	**12**	60.25	67
	Tex	15.5	58.75	65.5

Table 3: Performance evaluation across different table formats (Markdown, Json, Tex) for various models. Lower percentages (indicated by ↓) are better.

causes a dramatic surge in hallucinations. Notably, while this increase is primarily driven by a significant rise in Context-Violation, Query-Violation also exhibits an upward trend. This demonstrates that models are being successfully confused by the distractors. This finding showcases the diagnostic power of our benchmark to pinpoint specific failure modes that simple accuracy scores would obscure.

4.4 Influence of Output Format

We investigated whether the requested output format itself affects model reliability. We prompted three different models to generate tables in Markdown, JSON, and LaTeX (`Tex`) formats. As shown in Table 3, the choice of format has a significant, model-dependent, impact on hallucination rates.

Our key finding is that no single format is universally optimal. For example, o3-mini and Qwen3-32B perform best with Markdown, a simple and common format. In contrast, Llama-3.3-70B-Instruct is most reliable when generating JSON. This suggests that a model's performance is heavily influenced by the prevalence of certain data structures in its training corpus. Unsurprisingly, the syntactically complex `Tex` format often increased *Constraint-Violation*, as models struggled with its intricate syntax.

This analysis underscores the importance of a multi-faceted evaluation. It reveals that reliability is not just about a model's core reasoning ability but is also tied to its fluency in specific output schemas. This provides practical insight for users choosing a generation format and highlights a direction for future work: training models to be robust across a wider range of structured data representations.

4.5 Mitigation Strategies

We investigated whether table hallucinations can be mitigated through inference-time prompting or if they require training-time alignment. We evaluated

Method	Query-Violation(%) ↓	Context-Violation(%) ↓	Total Hallu(%) ↓
Base	15.35	53.05	59.10
Careful Emp.	15.40	52.30	58.50
Follow Emp.	15.25	<u>45.60</u>	<u>52.20</u>
COT	<u>15.15</u>	52.05	58.05
SFT	**13.85**	**13.90**	**24.45**

Table 4: Comparison of different methods on Qwen3-8B(in thinking mode). "Careful Emp." denotes prompts emphasizing careful verification of alignment between the response, query, and context; "Follow Emp." implies strict adherence to the query and context. Best results are **bolded** and second-best are <u>underlined</u>.

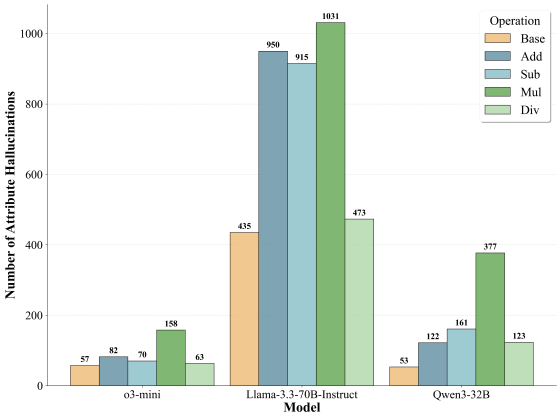

Figure 7: Performance on different basic arithmetic subsets. The y-axis denotes the number of samples with attribute hallucinations.

three prompting strategies—emphasizing verification ("Careful Emp."), strict adherence ("Follow Emp."), and Chain-of-Thought (COT)(Wei et al., 2022)—against Supervised Fine-Tuning (SFT) using our synthesized dataset.

As shown in Table 4, prompt engineering proved largely ineffective. Strategies like COT and "Careful Emp." yielded negligible gains, suggesting that mere instruction cannot overcome the model's sensitivity to noise. While "Follow Emp." offered a slight improvement by explicitly constraining the search space, it failed to address the root cause. In contrast, SFT achieved a transformative reduction in *Total Hallu* , driven primarily by a massive drop in *Context-Violation*. This result highlights the critical value of our data generation pipeline: by fine-tuning on samples containing programmatically injected adversarial noise, the model effectively learns to distinguish signal from noise, acquiring a robustness against distractors that prompting alone cannot instill. Detailed prompts can be found in Appendix E.

4.6 Beyond Information Extraction: In-Table Computational Failures

To push beyond simple information extraction, we evaluated the models' ability to perform basic arithmetic during table generation. We used a data subset with at least two numeric columns and prompted the models to add a final column containing the result of a simple operation (addition, subtraction, multiplication, or division) between two source columns. Errors in this computed column were classified as *Attribute Hallucinations*.

The results in Figure 7 reveal a striking failure in this capability. For all tested models, requiring an in-table calculation led to a massive increase in attribute hallucinations compared to the non-calculation baseline. For instance, the number of errors for Llama-3.3-70B-Instruct and Qwen3-32B more than doubled when performing multiplication. This indicates the cognitive load of simultaneously managing table structure, retrieving data, and executing a mathematical operation severely degrades their reliability.

This poor performance is particularly notable given that state-of-the-art LLMs have demonstrated high accuracy on standard arithmetic reasoning benchmarks (Cobbe et al., 2021; Anil et al., 2023). The stark contrast between their success in conversational math and their failure in structured computational tasks highlights a critical, previously under-explored vulnerability. Our findings suggest that future work must focus not just on improving reasoning in isolation, but on reliably integrating it within complex generation formats—a challenge our benchmark is designed to measure.

5 Conclusion

We introduce **TableHallu**, the first diagnostic benchmark for query-driven table generation, constructed via a scalable pipeline for programmatic distractor injection. Our evaluation reveals that LLMs—regardless of scale—are highly susceptible to hallucinations, failing at ordering constraints and in-table arithmetic despite strong standalone performance. While fine-tuning on our dataset mitigates these errors, our findings underscore a significant gap between models' general capabilities and their reliability in structured generation. We hope **TableHallu** steers future research toward achieving verifiable and multi-faceted reliability.

Limitations

We identify two primary limitations in this study. **First**, our scope is strictly focused on table generation. While the proposed taxonomy regarding *Constraint* and *Context* violations likely generalizes to other structured generation tasks (e.g., code generation), we do not empirically verify its applicability beyond tabular formats. **Second**, to guarantee the rigorousness of our diagnostic evaluation and the precision of ground truth labels, **TableHallu** is constructed via a synthetic pipeline. Although this design choice effectively eliminates the noise and label errors prevalent in existing datasets—and allows for the controlled injection of adversarial distractors—there remains a potential distribution shift between our synthesized scenarios and the uncurated, heterogeneous nature of data found in real-world applications.

Ethical Considerations

We strictly adhere to ethical research practices. First, given the synthetic nature of TableHallu, we conducted rigorous manual and automated checks to ensure the dataset is free of personally identifiable information (PII) and offensive content. Second, for human verification, we recruited expert annotators (aged 18–30) who were explicitly informed of the data's intended research use and provided consent. Third, all participants were compensated at an hourly rate significantly exceeding the local minimum wage, ensuring fair labor practices appropriate to their demographics.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, and 8 others. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Thang Luong, Wei Ping, Sharan Ramachandran, and 1 others. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Tejas Anvekar, Juhna Park, Aparna Garimella, and Vivek Gupta. 2025. Tabrex : Tabular referenceless explainable evaluation. *Preprint*, arXiv:2512.15907.

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

Zheye Deng, Chunkit Chan, Weiqi Wang, Yuxi Sun, Wei Fan, Tianshi Zheng, Yauwai Yim, and Yangqiu Song. 2024. Text-tuple-table: Towards information integration in text-to-table generation via global tuple extraction. *arXiv preprint arXiv:2404.14215*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Nouha Dziri, Ehsan Kamalloo, Sivan Milton, Osmar Zaiane, Mo Yu, Edoardo M Ponti, and Siva Reddy. 2022. Faithdial: A faithful benchmark for information-seeking dialogue. *Transactions of the Association for Computational Linguistics*, 10:1473–1490.

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

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Muhammad Salar Khan and Hamza Umer. 2024. Chatgpt in finance: Applications, challenges, and solutions. *Heliyon*, 10(2).

Rémi Lebret, David Grangier, and Michael Auli. 2016. Generating text from structured data with application to the biography domain. *CoRR, abs/1603.07771*.

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. Halueval: A large-scale hallucination evaluation benchmark for large language models. *arXiv preprint arXiv:2305.11747*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Michael Xieyang Liu, Frederick Liu, Alexander J Fiannaca, Terry Koo, Lucas Dixon, Michael Terry, and Carrie J Cai. 2024b. " we need structured output": Towards user-centered constraints on large language model output. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–9.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *Preprint*, arXiv:2307.03172.

Ben Malin, Tatiana Kalganova, and Nikoloas Boulgouris. 2024. A review of faithfulness metrics for hallucination assessment in large language models. *arXiv preprint arXiv:2501.00269*.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251*.

Siyuan Mu and Sen Lin. 2025. A comprehensive survey of mixture-of-experts: Algorithms, theory, and applications. *Preprint*, arXiv:2503.07137.

OpenAI. 2025a. Introducing gpt-4.1 in the api.

OpenAI. 2025b. Openai o3-mini.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Vipula Rawte, Amit Sheth, and Amitava Das. 2023. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922*.

Abraham Silberschatz, Henry F Korth, Shashank Sudarshan, and 1 others. 2002. *Database system concepts*, volume 5. Mcgraw-hill New York.

Xiangru Tang, Yiming Zong, Jason Phang, Yilun Zhao, Wangchunshu Zhou, Arman Cohan, and Mark Gerstein. 2023. Struc-bench: Are large language models really good at generating complex structured data? *arXiv preprint arXiv:2309.08963*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. 2023. Large language models in medicine. *Nature medicine*, 29(8):1930–1940.

Ritam Upadhyay, Naman Ahuja, Rishabh Baral, Aparna Garimella, and Vivek Gupta. 2025. Cmt-bench: Cricket multi-table generation benchmark for probing robustness in large language models. *Preprint*, arXiv:2510.18173.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.

Xueqing Wu, Jiacheng Zhang, and Hang Li. 2021. Text-to-table: A new way of information extraction. *arXiv preprint arXiv:2109.02707*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Jialin Yang, Dongfu Jiang, Lipeng He, Sherman Siu, Yuxuan Zhang, Disen Liao, Zhuofeng Li, Huaye Zeng, Yiming Jia, Haozhe Wang, and 1 others. 2025b. Structeval: Benchmarking llms' capabilities to generate structural outputs. *arXiv preprint arXiv:2505.20139*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and 1 others. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, and 1 others. 2025. siren's song in the ai ocean: A survey on hallucination in large language models. *Computational Linguistics*, pages 1–46.

A Pipeline Details

This appendix elaborates on the automated generation pipeline for the **TableHallu** dataset. The pipeline is engineered to systematically generate test cases for diagnosing hallucinations in Large Language Models (LLMs) for query-driven table generation tasks in a scalable, controllable, and high-fidelity manner. The entire process is meticulously designed into three core stages: **STEP 1. Ground-Truth Kernel Generation**, **STEP 2. Contextual Perturbation with Adversarial Distractors**, and **STEP 3. Multi-Stage Verification**. Throughout the pipeline, generation tasks are performed by **OpenAI GPT-4.1**, while the verification stage employs the **Qwen3-235B-A22B-Thinking-2507** model. The specific prompts guiding the LLMs at each stage are shown in the final pages.

A.1 STEP 1. Ground-Truth Kernel Generation

The objective of this stage is to construct a completely self-consistent and flawless "Ground-Truth Kernel". This kernel consists of three components: a structured **Ground-Truth Table**, a perfectly corresponding **Base Context**, and a **Natural Language Query** that can be uniquely answered from the context. This stage ensures that the foundation for our subsequent adversarial testing is absolutely reliable.

A.1.1 Topic Pool Curation

To ensure the diversity and breadth of the generated data, we first curated a structured, multi-level topic pool.

- **Hierarchical Structure**: The topic pool is organized in a JSON-like format, covering **12 high-level domains** (e.g., "Technology & Business", "Finance & Economics") and **24 fine-grained sub-categories** (e.g., "Technological Innovation & Adoption").
- **Template-based Generation**: Within each sub-category, we predefined multiple "Topic Templates". These templates contain placeholders (e.g., `{entity1}`, `{start_year}`) that allow the LLM to randomly fill them from a predefined list of entities. This approach facilitates the large-scale creation of semantically rich and realistic topics.

A.1.2 Paired Table-Text Instance Generation

In this phase, we leverage an LLM to generate paired tables and texts from the topic pool. Guided by a selected topic, the LLM is instructed to:

1. **Generate the Ground-Truth Table**: The LLM first generates a structured and factually accurate table in Json format. The generation is constrained to a size of **5-7 columns** and **5-8 rows**. Critically, the prompt mandates that **at least one cell must be null** to represent missing information and that the table rows must be **pre-sorted** by a logical column.
2. **Generate the Base Context**: Subsequently, using the generated table as the sole factual basis, the LLM writes a concise text. The prompt strictly enforces that every non-null value in the table must be mentioned, and crucially, any attribute corresponding to a null cell must be **completely omitted** from the narrative, without using phrases like "information unknown". This ensures a perfect, closed-world correspondence between the table and the context.

A.1.3 NLQ Formulation

Finally, based on the (Table, Text) pair, we instruct the LLM to construct a complex Natural Language Query (NLQ). The LLM is required to:

- **Define Scope and Structure**: The query must naturally integrate the table's subject and scope (e.g., a date range). It must also explicitly define the required table schema, listing all **column headers in their exact sequence**.
- **Specify Constraints**: The query must include the precise **sorting constraint** (column and order) derived from the table's metadata.
- **Maintain Abstraction**: The prompt explicitly forbids referencing any specific cell values, ensuring the query is about structure and scope, not data points.

A.2 STEP 2. Contextual Perturbation with Adversarial Distractors

This stage is the core of TableHallu's adversarial design. Our goal is not to modify the ground-truth

kernel directly but to "contaminate" only the **Base Context** by injecting "Distractors" that are inconsistent with the query's constraints.

A.2.1 Query-Driven Constraints Identification

Before injecting distractors, we prompt the LLM to analyze the NLQ and extract its constraints into a structured JSON format. The extracted fields include: `entity_scope`, `numerical_range`, `sort_column`, and `sort_order`. This structured output serves as a precise guide for the next step.

A.2.2 Contextual Perturbation with Adversarial Distractors

Based on the identified constraints, we guide the LLM to augment the base context using the detailed prompt that enforces a strict protocol for generating the **Adversarial Context**:

- **Mandatory Constraint Violation**: The core instruction is that every piece of distractor content **must violate at least one query constraint**. The prompt provides explicit examples for temporal, geographic, numerical, and categorical violations to ensure the distractors are effective.
- **Strict Prohibition Rules**: The LLM is forbidden from creating distractor entities that could satisfy all query constraints, be mistaken for ground-truth entries, or have names similar to ground-truth entities.
- **Original Context Dispersal**: The prompt mandates that the original context's key information be scattered throughout the expanded text, preventing simple paragraph-level retrieval.
- **Null Value Protection Protocol**: A critical instruction is to **completely avoid** providing information for attributes that were null in the ground-truth table, ensuring distractors do not accidentally fill these informational gaps.

A.3 STEP 3. Multi-Stage Verification

To ensure the highest data quality, we designed a verification protocol involving two LLM-based verifiers followed by a final human audit.

A.3.1 LLM Verifier 1 – Verify the Ground-Truth Kernel

Before any adversarial perturbation, the first LLM verifier reviews the kernel generated in STEP 1. It will check:

- **Context Quality and Consistency**: The context must be internally consistent and sufficient to answer the query.
- **Unique Derivation**: The table must be unambiguously and uniquely derivable from the context.
- **Content Correspondence**: The context must correspond exactly to the table, with no extra or missing information.

Only kernels that pass this check as "CORRECT" proceed to the next stage.

A.3.2 LLM Verifier 2 – Verify the Adversarial Context

After the adversarial context is generated, a second, two-part verification is performed:

1. **Ground-Truth Preservation**: The verifier confirms that all information from the original context is fully preserved in the new, longer adversarial context.
2. **Distractor Compliance**: The verifier ensures that the newly added distractor information does not inadvertently satisfy the query constraints, which would compromise the ground-truth table as the unique correct answer.

An instance must pass both checks to be considered valid.

A.3.3 Human Verification

As the final quality control measure, all samples that pass the automated LLM verification are ultimately audited by human experts. The annotators conduct a holistic review of the entire data instance (table, adversarial context, query) to assess its logical rigor, adversarial effectiveness, and textual naturalness. This stringent process ensures an accuracy of over 95% for the TableHallu dataset.

B Human Verification Process

To ensure the reliability and robustness of the TableHallu dataset, we conducted a rigorous manual verification process involving 10 qualified annotators.

All annotators possessed proficient bilingual (English and Chinese) reading comprehension skills and strong logical reasoning capabilities. The verification workflow was structured into three stages: training, a pilot phase, and formal annotation. During the pilot phase, each annotator processed 50 samples to align with the evaluation criteria, followed by a standard adjustment session to unify acceptance standards before the formal verification of the full dataset.

The manual verification protocol enforced a strict three-step assessment for each data sample. First, annotators evaluated the **Query** for clarity, verifying the presence of essential constraints (e.g., category, column headers, and sorting rules). Second, the **Ground Truth (GT) Table** was checked against the GT Context to ensure factual accuracy, correct formatting, and strict adherence to row-sorting constraints. Third, and most critically, the **Distractor Context** was scrutinized to verify that (1) it fully encompassed the GT Context information, and (2) no added distractor information inadvertently satisfied the query constraints, thereby guaranteeing the uniqueness of the GT Table as the solution. Ambiguous samples were subjected to cross-validation and group discussion to maintain high-quality standards.

C Hallucination Detection

To systematically evaluate the factuality of the generated tables, we employ the detection logic detailed in Algorithm 1. This procedure hierarchically compares the model-generated table, T_A, against a ground-truth table, T_{GT}, and a provided textual context, C, to identify and categorize five distinct types of hallucinations. The process is as follows:

1. Format Hallucination. The initial and most fundamental check is for a **Format Hallucination**. This error category is triggered under two primary conditions. First, it occurs if the model's response is not rendered in a parsable format, such as producing plain text instead of a table or using malformed Markdown syntax, which prevents successful data extraction. This represents a complete failure to adhere to the task's structural requirements. Second, if the table is successfully parsed, it is still flagged for a Format Hallucination if its column structure (i.e., the names or order of its columns) deviates from that of the ground-truth table T_{GT}. In either case, a structural mismatch is considered a critical failure that precludes any meaningful cell-wise comparison, causing the algorithm to terminate its analysis.

2. Row Matching. Following structural validation, the algorithm performs a row-matching procedure. Its purpose is to establish a one-to-one correspondence between entities in T_A and T_{GT}, typically by matching values in a designated primary key column. This step is crucial as it partitions the rows of T_A into two distinct sets for subsequent analysis: matched rows (M) and unmatched rows (U).

3. Blank Fabrication. For each matched row pair in the set M, we first inspect for a **Blank Fabrication** error. This hallucination type occurs when the model generates a value for a cell (c_A) that is intended to be empty in the ground truth (c_{GT}). It represents the unwarranted invention of a piece of information for a correctly identified entity and is checked before other attribute errors.

4. Attribute Hallucination. If a cell is not a blank fabrication, we then check for an **Attribute Hallucination**. This error is logged if a cell value c_A in the generated table does not match the corresponding non-empty cell c_{GT} in the ground truth. This signifies that the model has retrieved or inferred incorrect information for a specific attribute of a known entity. The matching criteria are strict for numerical values and employ semantic similarity for textual content.

5. Scope Hallucination. This check is performed on the set of unmatched rows, U. An unmatched row r_U is classified as a **Scope Hallucination** if its primary key exists within the broader textual context C, but is not present in the ground-truth table T_{GT}. This indicates that the model has included a factually existing entity that falls outside the precise scope of the user's query (e.g., returning a fourth item when only a top-three list was requested).

6. Entity Fabrication. Finally, if an unmatched row's primary key cannot be found even in the provided context C, it is classified as an **Entity Fabrication**. This is the most severe type of hallucination, representing the creation of a completely fictitious entity that has no basis in the provided information sources.

This hierarchical and multi-faceted approach allows for a fine-grained and comprehensive eval-

uation of a model's ability to generate factually accurate and relevant tabular data.

C.1 Implementation Details

To operationalize the detection logic described above, we defined several key implementation details regarding data standardization, primary key identification, and the cell-matching criteria.

Cell Standardization. To ensure robust comparison and account for the inherent variability of Large Language Model (LLM) outputs, we first perform a cell standardization pre-processing step. Recognizing that LLMs may generate dates in numerous different formats, we employ a comprehensive set of pattern-matching rules to normalize all date-like strings into a single, consistent `yyyy-mm-dd` format. Similarly, a wide array of expressions for empty or unavailable data (e.g., 'n/a', 'null', '—', 'not available', etc.) are identified from a predefined list of patterns and standardized to a canonical representation ('-'). This pre-processing is critical to prevent superficial formatting differences from being misclassified as attribute errors.

Primary Key Identification. The identification of a primary key is essential for the row-matching and source-analysis steps. Our methodology employs a heuristic to automatically designate a primary key for any given table: we select the first non-numeric column encountered when scanning from left to right. In the case that all columns are determined to be numeric, the first column is used as the primary key by default. This key is subsequently used to match entities between the generated table T_A and the ground-truth table T_{GT} and, for unmatched rows, to query the context C in order to differentiate between Scope Hallucinations and Entity Fabrications.

Hierarchical Cell Matching. Our cell-matching logic follows a hierarchical strategy to accommodate different data types and minor textual variations. A distinction is first made between numeric and string-based cells. For numeric cells, a strict equality check is performed. For string cells, we first test for mutual substring inclusion (i.e., if one cell's content is contained within the other's). This rule effectively handles cases where the LLM provides the correct information but appends it with extraneous or conversational text. If the substring check does not yield a match, we then proceed to semantic similarity matching. Specifically, we encode the cell contents using the `all-mpnet-base-v2` Sentence Transformer model and compute the cosine similarity between the resulting embeddings. A cell pair is considered a match if their similarity score exceeds a threshold of 0.8. We found through empirical observation of samples that this relatively lenient threshold is highly effective at correctly identifying semantically equivalent cells that are phrased differently.

C.2 Evaluation Metrics

To quantify model performance and characterize the different failure modes, we define a suite of evaluation metrics based on the hallucination types identified by our methodology. These are divided into foundational and aggregated metrics.

Foundational Metrics. We first establish six foundational metrics, each corresponding directly to one of the hallucination categories. For each metric, we calculate the percentage of total samples in the evaluation set that are flagged with that specific type of error. The six metrics are: **FormatError (%)**, **RangeHallu (%)** (for Scope Hallucinations), **OrderMismatch (%)** (for Sorting Hallucinations), **AttributeHallu (%)**, **BlankFilling (%)**, and **EntityFabrication (%)**.

Aggregated Metrics. To provide a higher-level, more interpretable view of model behavior, we group the foundational errors into two macro categories and define an overall hallucination rate.

- **Constraint-Violation (%):** This metric captures errors related to a model's failure to adhere to the explicit or implicit constraints of a query. A sample is counted as a constraint violation if it exhibits a **FormatError**, **RangeHallu**, or **OrderMismatch**. The final metric is the percentage of samples that contain at least one of these three error types.
- **Context-Violation (%):** This metric measures a model's failure to faithfully represent the factual content provided in the context. A sample is counted as a context violation if it contains an **AttributeHallu**, **BlankFilling**, or **EntityFabrication**. This is calculated as the percentage of samples with one or more of these three content-related errors.
- **TotalHallu (%):** This represents the overall hallucination rate. It is defined as the percentage of samples that contain at least one

hallucination of any of the six foundational types. A sample with multiple types of errors is counted only once for this metric, providing a clear measure of the proportion of flawed outputs in the entire evaluation set.

D Experimental Details

D.1 General Implementation

We deploy all Large Language Models (LLMs) using the `vllm` inference library. To mitigate the issue of repetitive outputs, we set the generation `temperature` to 0.6. Furthermore, we configure the maximum number of output tokens (`max_tokens`) to 30,000. This high token limit is intended to prevent premature truncation, especially for queries that require extended reasoning chains, thereby ensuring the completeness of the generated tables. The specific prompts utilized for the main experiments and subsequent studies are detailed in the *Prompts* section at the end of this document.

Models Our evaluation encompasses a diverse range of state-of-the-art large language models to ensure a comprehensive analysis. To promote reproducibility and transparency, we have primarily utilized open-source models publicly available on the Hugging Face Hub. Table **??** provides a complete list of all models employed in our experiments, along with direct hyperlinks to their official sources.

D.2 Ablation and Format Experiments (Table 2 & 3 in Main Text)

For the ablation study on the "Thinking Mode" presented in Table 2 of the main paper, we disabled the default "Thinking Mode" of the Qwen3 model by appending the `/no_think` flag to the input prompt.

For the experiments on different table formats, as shown in Table 3, we instructed the LLMs via prompts to generate tables in various formats (e.g., Markdown, LaTeX, JSON). Crucially, each prompt included a format-specific example to guide the model's output structure. During the evaluation phase, all generated tables, regardless of their original format, were standardized by converting them into `pandas DataFrame` objects. This uniform data structure is essential for the consistent application of our hallucination detection and classification pipeline.

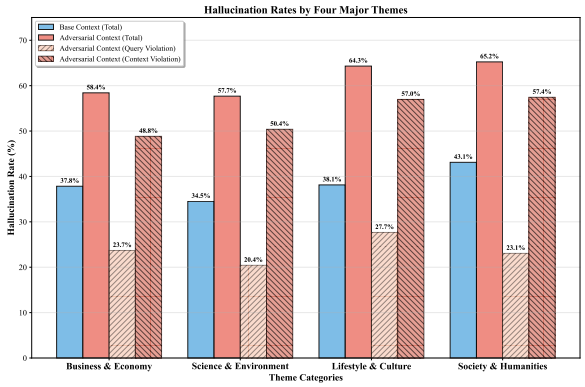

Figure 8: Average hallucination rates across four major thematic categories, providing a high-level overview of domain-specific challenges.

D.3 Basic arithmetic subsets(Figure 7 in Main Text)

For the table calculation experiment detailed in Figure 7 of the main paper, we constructed a specialized subset of 195 samples, each guaranteed to have no empty cells. To create ground-truth samples involving basic arithmetic and ensure the correctness of the results, we programmatically extracted two numerical columns from the ground-truth table (`gt_table`) using regular expressions and a Python script. These columns were then used to compute a third result column. This calculated column was named **"Composite Value"** and appended as the final column to the `gt_table`.

Subsequently, we employed an LLM to revise the corresponding user query. The revised query explicitly described the calculation method for the "Composite Value" column, specifying which two source columns were used and the arithmetic operation performed. This methodology minimizes direct LLM involvement in the ground-truth data creation process, thereby preventing the introduction of factual hallucinations into the evaluation data itself.

D.4 Thematic Analysis: Uncovering Domain-Specific Vulnerabilities

To investigate whether models' susceptibility to hallucination is domain-dependent, we analyzed their average performance at both a macro level (four major themes) and a micro level (12 subcategories). The results, presented in Figure 8 and Figure 9, reveal several critical insights into the nature of model failures.

First, a universal trend observed at both levels is the dramatic increase in total hallucination rates

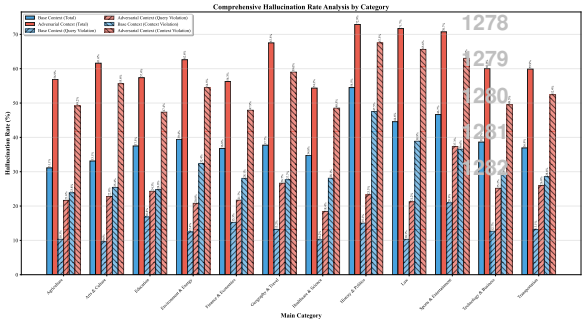

Figure 9: A detailed breakdown of hallucination rates across 12 sub-categories. This granular view reveals extreme performance disparities and pinpoints specific areas of model vulnerability, such as History & Politics and Law.

when models are exposed to adversarial contexts. The introduction of targeted distractors consistently caused a significant performance degradation compared to the base context baseline, validating the effectiveness of our adversarial data construction in exposing model weaknesses.

From a high-level perspective (Figure 8), the analysis uncovers distinct domain-specific vulnerabilities. Models exhibit the highest overall hallucination rates in the **Lifestyle & Culture (64.3%)** and **Society & Humanities (65.2%)** categories. This suggests that topics involving more narrative, subjective, or nuanced human-centric information are significantly more challenging for models to ground accurately.

A deeper, more granular analysis across 12 subcategories (Figure 9) pinpoints these weaknesses with greater precision. The highest error rates are found in abstract and reasoning-intensive domains like **History & Politics (72.9%)** and **Law (71.7%)**. This indicates that models struggle immensely when tasks require understanding complex causal relationships, abstract legal or political concepts, and dense historical narratives. In contrast, more structured domains like **Technology & Business (60.2%)** show comparatively lower, yet still substantial, error rates.

Finally, and most importantly, the detailed breakdown confirms the dominant failure pattern. Across nearly all 12 sub-categories, **Context Violation is consistently the primary driver of hallucinations** in the adversarial setting. For instance, in the most challenging "History & Politics" theme, context-violation errors are rampant. This indicates that when faced with conflicting information, models are far more likely to disregard the provided source text and fabricate information rather than violate explicit query constraints. This persistent pattern across diverse topics highlights a fundamental weakness in current models' grounding capabilities when under cognitive load.

E Prompts and Cases

Prompt for Paired Table-Text Instance Generation (Part 1: Instructions)

You are an expert data curator and content creator. Your task is to generate a ground-truth dataset for a text-to-table task based on the specific topic. This dataset must contain a JSON object with two main keys: "gt_table" and "gt_context".
Follow these instructions precisely:

1. **Topic Specification:** Use the following specific topic: {topic}
   Main Category: {main_category}
2. **Create the Ground-Truth Table ('gt_table'):**
   - Design a table with 4-7 columns and 5-8 rows about the specified topic.
   - Each cell must contain only a single value: This can be a single number, a single entity (e.g., a person, place, or thing), or null.
   - **Crucially, at least 1 cell in the table must be EMPTY (null).** This represents information that is not available.
   - **The table rows must be sorted by one of the columns in a logical order** (e.g., alphabetical order for text values, numerical order for numbers, chronological order for dates). Choose the most appropriate column for sorting based on the topic and data type.
   - The table data must be accurate and factual based on your chosen topic.
   - Format the table and its metadata strictly according to the provided JSON structure.
3. **Create the Ground-Truth Context ('gt_context'):**
   - Write a clear, factual paragraph that describes **only** the information present in the 'gt_table'.
   - Every non-null value in the table must have a direct corresponding mention in the paragraph.
   - **Crucially, for cells that are 'null' (empty), you must completely omit any mention of that attribute for that entity.** For example, if a product's price is 'null', your description of that product will cover its other details but will not mention the price at all. **Do not** use phrases like "price is unknown" or "information was not disclosed". Ensure the context is written so that the missing information cannot be logically inferred.
   - Do not include any information that is not present in the 'gt_table'.
4. **Output Format:** Provide the final output as a single JSON object, adhering strictly to the format provided in the following prompt box.

Prompt for Paired Table-Text Instance Generation (Part 2: JSON Structure)

JSON Output Structure:

```
{
  "gt_table": {
    "table": {
      "<Table Name: String>": [
        {
          "<Column Header 1: String>": "<Value: String/Number/null>",
          "<Column Header 2: String>": "<Value: String/Number/null>",
          ...
          "<Column Header N: String>": "<Value: String/Number/null>"
        }
      ]
    },
    "metadata": {
      "table_info": {
        "<Single Table Name: String>": {
          "rows": "<Integer>",
          "columns": "<Integer>",
          "topic": "{main_category}",
          "row_headers": [
            "<String: Header for row 1>",
            "<String: Header for row 2>",
            ...
          ],
          ...
        }
      }
    }
  }
}
```

```
        "column_headers": [
            "<String: Column Header 1>",
            "<String: Column Header 2>",
            ...
          ],
          "sort_column": "<String: Column header used for sorting>",
        "sort_method": "<String: Sorting method - 'ascending' or 'descending'>"
        }
      }
    }
  },
  "gt_context": "<Generated descriptive paragraph here>"
}
```

Now, please generate one complete instance based on the given topic.

Prompt for NLQ Formulation (Part 1: Instructions)

Role: You are an expert data analyst. Your task is to write a precise, clear, and natural language query for an AI assistant. This query must instruct the assistant to extract information from a ground-truth context and format it into the ground-truth table according to your specified structure and scope.
Objective: Based on the provided 'gt_table' (ground-truth table) and 'gt_context' (ground-truth context), generate a high-quality, natural language query.
Instructions:

1. **Identify Subject and Scope:**
    - First, identify the core **subject** of the table from the 'gt_table's' title or the 'gt_context' (e.g., "China's Key Financial Indicators", "Flagship Smartphone Lineup").
    - Next, determine the overall **scope/range** of the data. For time-series data, specify the start and end period (e.g., "from 2018 to 2023"). For categorical data, this could be a descriptive group (e.g., "tech companies in Silicon Valley"). For numerical data, it could be a range (e.g., "revenue between 100M and 1B USD").
    - Naturally integrate the subject and scope/range into the beginning of your query.
2. **Define Table Schema:**
    - You **must** explicitly list all required **column headers** and arrange them in the **exact order** they appear in the 'gt_table'.
    - Use clear phrasing, such as: "The columns should be, in order: ..." or "The table should have the following columns in this sequence: ...".
3. **Specify Sorting Constraint:**
    - Use the sorting information provided in the 'gt_table' metadata ('sort_column' and 'sort_method') to determine the exact sorting requirement.
    - State this sorting rule as an explicit constraint in the query. For example: "sort the table by [sort_column] in [sort_method] order" where [sort_method] should be either "ascending" or "descending".
4. **Avoid Specific Cell Values:**
    - Your query is about **scope, structure and format**, not specific data points.
    - **Do not** reference any specific cell values in your query (e.g., do not write "...where the GDP growth in 2018 was 6.7%" or "...which includes the value 2494").
5. **Maintain Natural Language:**
    - Write the entire query as a single, coherent, human-like command or request. It should be clear and unambiguous.

Prompt for NLQ Formulation (Part 2: Examples & Execution)

Example: If 'gt_table' is about multi-year financials for several companies:

- **Good Query:** "Please create a table summarizing the financial performance of tech companies in Silicon Valley from 2022 to 2023, based on the context. The table must include the columns 'Company Name', 'Year', and 'Revenue (USD Billions)', in that exact order and sort the results by 'Company Name' in ascending order."
- **Bad Query:** "Give me the 2022 revenue for Apple and Google, and also their 2023 revenue." (Too specific, lacks structural definition).

Your Task: Strictly follow the five rules above to generate a high-quality, natural language 'query' for the provided 'gt_table' and 'gt_context'.
Output Format: Directly output the generated natural language 'query' as raw text. Do not add any explanations, titles, or extra formatting.

Input:
{output_of_step1}

Prompt for Query-Driven Constraint Identification (Part 1: Task Definition)

Role: You are an expert at analyzing table generation queries and extracting constraints. Your task is to identify and extract various constraints from user queries for text-to-table tasks.
Given a user query that requests table generation, you need to identify:

1. **Entity Scope**: The specific subject matter, domain, or entities that the table should focus on (e.g., "venture capital funding rounds for DeFi startups", "institutional AI investments", "XR patent filings in the USA and China").
2. **Numerical Range**: Time periods, date ranges, or numerical constraints that limit the data scope. This can be:
   - Single time points (e.g., "2023", "1960", "2022-Q4")
   - Time ranges (e.g., "2018 to 2021", "1960 and 1980", "Nov 2023 to Dec 2023")
   - Other numerical constraints (e.g., price ranges)
   - Set to "None" if no numerical constraints are specified
3. **Sort Column**: The column name by which the table should be sorted.
4. **Sort Order**: Either "ascending" or "descending".

Prompt for Query-Driven Constraint Identification (Part 2: Few-Shot Examples)

Here are some examples:

Example 1:

Query: "Please create a table presenting a comparative timeline of the British and Mongol Empires... Ensure the table is sorted by 'Year' in ascending order."

```
Output:
{
  "constraints": {
    "entity_scope": ["British Empire", "Mongol Empire"],
    "numerical_range": "None",
    "sort_column": "Year",
    "sort_order": "ascending"
  }
}
```

Example 2:

Query: "Please generate a table summarizing venture capital funding rounds for DeFi startups in 2022... Make sure to sort the table by 'Amount Raised (USD Million)' in descending order."

```
Output:
{
  "constraints": {
    "entity_scope": ["Venture capital funding rounds", "DeFi startups"],
    "numerical_range": "2022",
    "sort_column": "Amount Raised (USD Million)",
    "sort_order": "descending"
  }
}
```

Example 3:

Query: "Please create a table summarizing the Humira revenue breakdown by region for Roche in 2021... Make sure to sort the table by 'Region' in ascending order."

```
Output:
{
  "constraints": {
    "entity_scope": ["Humira revenue breakdown", "Roche"],
    "numerical_range": "2021",
    "sort_column": "Region",
    "sort_order": "ascending"
  }
}
...
```

Prompt for Query-Driven Constraint Identification (Part 3: Format & Execution)

Output your analysis only in the following JSON format without extra content or explanation:

```
{
  "constraints": {
    "entity_scope": "[extracted entity scope]",
    "numerical_range": "[extracted numerical range or 'None']",
    "sort_column": "[column name for sorting]",
    "sort_order": "[ascending or descending]"
  }
}
```

Important Guidelines:

- Extract the entity scope as precisely as possible, focusing on entities with clear boundaries and distinguishing characteristics.
- For `numerical_range`, capture time periods, years, or any numerical constraints mentioned.
- Use exact column names as specified in the query for `sort_column`.
- If no numerical constraints are mentioned, set `numerical_range` to "None".

Now analyze the following query and extract the constraints:
{query}

Prompt for Contextual Perturbation (Part 1: Task Setup & Logic)

Role: You are an expert at generating distractor content for text-to-table tasks. Your objective is to create expanded context that includes additional information which does NOT satisfy the query constraints, while preserving all original context information to make table extraction more challenging for LLMs.
Input Parameters:

- **Query**: {query}
- **Original Context**: {gt_context}
- **Ground Truth Table**: {gt_table_json}
- **Query Constraints**: {provided_query_constraints}
- **Difficulty Level**: {provided_difficulty_level}

CRITICAL CONSTRAINT VIOLATION REQUIREMENTS:
Mandatory Distractor Rules:
ABSOLUTE REQUIREMENT: Every piece of distractor content MUST violate at least one query constraint.
Constraint Violation Examples:

- **Temporal**: If constraint specifies "2020" $\rightarrow$ include 2019, 2021, or other years (NEVER 2020).
- **Geographic**: If constraint specifies "Europe" $\rightarrow$ include Asia, Africa, Americas (NEVER European entities).
- **Entity Type**: If constraint specifies "electric vehicles" $\rightarrow$ include gasoline cars, motorcycles (NEVER electric vehicles).
- **Numerical Range**: If constraint specifies "price under $200" $\rightarrow$ include items over $200 (NEVER items under $200).
- **Categorical**: If constraint specifies categories "A, B, C" $\rightarrow$ include categories "D, E, F" (NEVER A, B, or C).

Prompt for Contextual Perturbation (Part 2: Strategy & Guidelines)

Strict Prohibition Rules:

1. **Never create entities that satisfy ALL query constraints simultaneously.**
2. **Never include data that could be mistaken for ground truth entries.**
3. **Never use ambiguous entities that might partially match constraints.**
4. **Never include entities with similar names to ground truth entities.**

Distractor Generation Strategy:
For each constraint type, generate content that CLEARLY violates the specified constraints.
Example: For constraint "European countries in 2023":

- ✓**CORRECT**: Add Asian countries' data from 2023 (violates geographic constraint).
- ✓**CORRECT**: Add European countries' data from 2022/2024 (violates temporal constraint).
- ✓**CORRECT**: Add other regions' data from different years (violates both constraints).
- × **WRONG**: Add any European country data from 2023 (satisfies all constraints).

Original Context Dispersal:
Distribute original context's key information throughout the expanded content. Never cluster Original Context in a single paragraph. This ensures the extraction task remains challenging while preserving all necessary data.
Null Value Protection Protocol:

- **Completely avoid** providing information for attributes with null values.
- **Never include** direct statements, implications, or negative statements about missing data.
- **Ensure** distractor entities don't fill gaps in ground truth table data.

Difficulty Scaling

- **Easy**: Expand to ∼10x original context length
- **Medium**: Expand to ∼20x original context length
- **Hard**: Expand to ∼30x original context length

Writing Standards
Produce professional, coherent content resembling high-quality journalism:

- Smooth paragraph transitions
- Natural narrative flow
- Appropriate connecting phrases
- Human-like writing patterns

Final Validation Checklist
Before submitting, ensure:

1. ✓All distractor entities violate at least one query constraint
2. ✓No distractor entity could be confused with a valid table entry
3. ✓Original context is preserved and dispersed
4. ✓Content flows naturally and professionally
5. ✓Null value protection is maintained

Output Requirements
Provide only the final expanded context. Do not include explanations, meta-commentary, or process descriptions.

Prompt for LLM Verifier 1 – Verify the ground-truth kernel

Please evaluate whether the markdown table is correctly derived from the given context according to the query.

Query:
{query}

Context:
{gt_context}

Table:
{gt_table}

Please thoroughly check these aspects:

1. **Context Quality and Consistency**:
    - Is the context internally consistent without contradictions?
    - Does the context contain all necessary information required by the query?
    - Are there any logical inconsistencies or conflicting data points in the context?
    - Is the context complete enough to answer the query requirements?
2. **Unique Derivation**:
    - Can the table unambiguously be derived from the context based on the query requirements?
    - Is there sufficient information in the context to generate exactly this table?
    - Are all data points in the table clearly supported by the context?
3. **Content Correspondence**:
    - Does the content in the context exactly correspond to the table (no more, no less)?
    - The table should not contain:
        - Any additional columns or rows not supported by context
        - Any duplicate rows or redundant information
            - Example:
              Context: Brazil set a target to reduce greenhouse gas emissions by 37% from 2005 levels by 2025, and as of 2022, had reported a 29% reduction. For 2030, Brazil aims for a 43% reduction from 2005 levels, with a net zero target year of 2050. China has a 2030 target to reduce emissions by 60% from 2005 levels, and by 2022, had achieved a 51% reduction. China also has a more ambitious 2030 target of a 65% reduction from 2005 levels, with a net zero target year of 2060.
              Table:
              ...
              Result: INCORRECT
        - Any data that contradicts the context
    - Are all values in the table accurate according to the context?

Please answer "CORRECT" only if ALL aspects are satisfied. Answer "INCORRECT" if ANY aspect fails.

Format:
Result: [CORRECT / INCORRECT]
Reason: [Explanation of your assessment, if INCORRECT]

Prompt for LLM Verifier 2 – Verify the Adversarial Context - Step 1

Please analyze the following two texts and determine whether all information from the Original Context is completely preserved in the Full Context.

Original Context:
{gt_context}

Full Context:
{distractor_context}

Please carefully analyze:

1. Whether EVERY specific data point, fact, and number from the original context can be found in the full context
2. Whether the key information maintains its original accuracy and completeness
3. Whether the content itself remains consistent even if the information in original context appears in different positions within the full context

Please only answer "YES" or "NO" and provide a brief explanation.

Format:
Result: [YES/NO]
Reason: [Brief explanation]

Prompt for LLM Verifier 2 – Verify the Adversarial Context - Step 2

Please analyze whether the distractor information in the expanded context would compromise the uniqueness of the ground-truth table as the correct answer.

Background:

- The Original Context contains information that can be used to generate the Ground-Truth Table according to the Query requirements
- The Expanded Context includes the Original Context plus additional distractor information
- The goal is to ensure that the distractor information does NOT satisfy the query constraints, so it won't interfere with the Ground-Truth Table being the unique correct answer

Query:
{query}

Query Constraints:
{query_constraints}

Original Context:
{gt_context}

Ground-Truth Table (correct answer derived from Original Context):
{gt_table}

Expanded Context (Original Context + Distractor Information):
{distractor_context}

Please analyze:

1. Identify the distractor information (content in Expanded Context that is NOT present in Original Context)
2. Check if any distractor information satisfies the query constraints (time range, entity scope, data requirements, etc.)
3. Determine if the distractor information could be used to generate a valid alternative table that competes with the Ground-Truth Table

Validation Criteria:

- VALID: Distractor information does NOT satisfy query constraints, ensuring Ground-Truth Table remains the unique correct answer
- INVALID: Distractor information DOES satisfy query constraints, potentially creating competing valid answers

Please answer "VALID" or "INVALID" and provide detailed reasoning.

Format:
Result: [VALID/INVALID]
Reason: [Detailed analysis of distractor information and its impact on answer uniqueness]

Base Prompt (Markdown) for Table Generation

Context:
{distractor_context}

Query:
{query}

Only output a table in Markdown format, without any additional information or explanation. Please format your response using standard Markdown formatting syntax (columns separated by `|` and `---` as the separator line after the header row). The final table MUST BE put after `####`.

CoT Prompt for Table Generation

Context:
{distractor_context}

Query:
{query}

Only output a table in Markdown format, without any additional information or explanation. Please format your response using standard Markdown formatting syntax (columns separated by `|` and `---` as the separator line after the header row). The final table MUST BE put after `####`. Let's think step by step.

Following emphasis Prompt

Context:
{distractor_context}

Query:
{query}

Only output a table in Markdown format, without any additional information or explanation. Please format your response using standard Markdown formatting syntax (columns separated by `|` and `---` as the separator line after the header row). The final table MUST BE put after `####`. IMPORTANT: You must strictly adhere to the provided context and query, and only extract information that is explicitly mentioned.

Careful Prompt

Context:
{distractor_context}

Query:
{query}

Only output a table in Markdown format, without any additional information or explanation. Please format your response using standard Markdown formatting syntax (columns separated by `|` and `---` as the separator line after the header row). The final table MUST BE put after `####`. Before providing your final answer, please carefully review and double-check the accuracy of all data in your table to ensure it correctly reflects the information from the context.

Prompt for JSON Output Format

Context:
{distractor_context}

Query:
{query}

Only output a table in JSON format, without any additional information or explanation. Please format your response using JSON formatting syntax. The response should follow this format:

```
{
    "<Table Name: String>": [
        {
            "<Column Header 1: String>": "<Value: String/Number>",
            "<Column Header 2: String>": "<Value: String/Number>",
            ...
            "<Column Header N: String>": "<Value: String/Number>"
        }
    ]
}
```

Prompt for LaTeX `tabular` Output

Context:
{distractor_context}

Query:
{query}

Only output a table in LaTeX tabular format, without any additional information or explanation. Please format your response using LaTeX tabular formatting syntax. The response should follow this format:

```
\begin{tabular}{|l|c|r|}
\hline
\textbf{Column1} & \textbf{Column2} & ... & \textbf{ColumnN} \\
\hline
Value1 & Value2 & ... & ValueN \\
Value4 & Value5 & ... & ValueN \\
\hline
\end{tabular}
```

Bad Case: Format Error

Query:

```
Please create a table summarizing institutional AI investments in 2021, based on the provided context. The table should have the following columns in this sequence: "Institution", "Country", "Investment Amount (USD Million)", "Number of Deals", "Top AI Company Invested", and "Quarter of Largest Deal". Sort the table by "Investment Amount (USD Million)" in ascending order. Only output a table in Markdown format, without any additional information or explanation.
```

Ground Truth Table:

```
| Institution            | Country   | Investment Amount (USD Million) | Number of Deals | Top AI Company Invested | Quarter of Largest Deal |
|------|-----|-----|-------|-------|--------|-----|
| Temasek Holdings       | Singapore | 800                             | 5                    | Trax                    | -                       |
| Sequoia Capital        | USA       | 950                             | 7                    | Databricks              | Q1                      |
| Andreessen Horowitz    | USA       | 1100                            | 6                    | OpenAI                  | Q2                      |
| BlackRock              | USA       | 1200                            | 8                    | UiPath                  | Q2                      |
| Tencent Holdings       | China     | 1800                            | 9                    | CloudWalk               | Q3                      |
| SoftBank Vision Fund   | Japan     | 2500                            | 10                   | SenseTime               | Q4                      |
```

Response:

```
| Institution            | Country   | Investment Amount (USD Million) | Number of Deals | Top AI Company | Quarter of Largest Deal |
|------|-----|-----|-------|-------|--------|-----|
| Temasek Holdings       | Singapore | 800                             | 5                    | Trax           |                         |
| Sequoia Capital        | USA       | 950                             | 7                    | Databricks     | First Quarter           |
| Andreessen Horowitz    | USA       | 1100                            | 6                    | OpenAI         | Second Quarter          |
| BlackRock              | USA       | 1200                            | 8                    | UiPath         | Second Quarter          |
| Tencent Holdings       | China     | 1800                            | 9                    | CloudWalk      | Third Quarter           |
| SoftBank Vision Fund   | Japan     | 2500                            | 10                   | SenseTime      | Fourth Quarter          |
```

Bad Case: Attribute Hallucination and Blank Filling

Query:

```
Please create a table summarizing publishing and digital platform revenues from 2019 to 2021, based on the provided context. The table should have the following columns in this sequence: "Year", "Traditional Publishing Revenue (USD Billion)", "YouTube Revenue (USD Billion)", "Amazon Kindle Revenue (USD Billion)", "Traditional Publishing YoY Growth (%)", "YouTube YoY Growth (%)", and "Amazon Kindle YoY Growth (%)". Make sure to sort the table by "Year" in ascending order. Only output a table in Markdown format, without any additional information or explanation.
```

Ground Truth Table:

```
| Year | Traditional Publishing Revenue (USD Billion) | YouTube Revenue (USD Billion) | Amazon Kindle Revenue (USD Billion) | Traditional Publishing YoY Growth (%) | YouTube YoY Growth (%) | Amazon Kindle YoY Growth (%) |
|------|-----|-----|-------|-------|--------|-----|
| 2019 | 25.7 | 15.1 | 3.3 | 1.2 | 35.8 | 13.8 |
| 2020 | 26.2 | 19.8 | 4.0 | 1.9 | 31.1 | 21.2 |
| 2021 | 27.0 | 28.8 | - | 3.1 | 45.5 | - |
```

Response:

```
| Year | Traditional Publishing Revenue (USD Billion) | YouTube Revenue (USD Billion) | Amazon Kindle Revenue (USD Billion) | Traditional Publishing YoY Growth (%) | YouTube YoY Growth (%) | Amazon Kindle YoY Growth (%) |
|------|-----|-----|-------|-------|--------|-----|
| 2019 | 25.7 | 15.1 | 3.3 | 1.2 | 35.8 | 13.8 |
| 2020 | 26.2 | 19.8 | 4.0 | 1.9 | 31.1 | 21.2 |
| 2021 | 27.0 | 28.8 | 4.0 | 3.1 | 45.5 | 0.0 |
```

Bad Case: Range Hallucination

Query:

```
Please create a table summarizing the discoveries and milestones up to the Nobel Prize in Chemistry 2021, as described in the context, covering the period from 2000 to 2021...
```

Context:

```
......By 2003, research in South Korea led to the development of
bifunctional thiourea catalysts for Michael addition reactions, with
notable publications appearing in Chemical Communications. This
innovation contributed to the broader understanding of non-metal
catalysis, though it was not directly connected to the discoveries
leading to the Nobel Prize in Chemistry 2021. ......
```

Response:

```
| Year | Discovery | Scientist(s) | Country | Catalyst Type | Application Area | Notable Publication |
|------|-----|-----|-------|-------|--------|-----|
| 2000 | First asymmetric organocatalysis using proline | Benjamin List | Germany | proline | pharmaceutical synthesis | Journal of the American Chemical Society |
| 2000 | Enamine catalysis for asymmetric synthesis | David W. C. MacMillan | United States | imidazolidinone | asymmetric Diels-Alder reactions | Journal of the American Chemical Society |
| 2003 | Bifunctional thiourea catalysts for Michael addition reactions | Scientists in South Korea | South Korea | thiourea | Michael addition reactions | Chemical Communications |
| 2005 | Iminium catalysis | David W. C. MacMillan | United States | imidazolidinone | organocatalytic transformations | Science |
| 2005 | Organocatalysis expanded to cascade reactions | Benjamin List | Germany | proline | cascade reactions | Angewandte Chemie International Edition |
| 2006 | N-heterocyclic carbene (NHC) catalysis for benzoin condensation | Research group in Switzerland | Switzerland | N-heterocyclic carbene (NHC) | synthesis of fine chemicals | European Journal of Organic Chemistry |
| 2008 | Copper-based catalysts for synthesis of heterocyclic compounds | Scientists in India | India | copper-based | synthesis of heterocyclic compounds | Green Chemistry |
| 2010 | Chiral sulfinamide catalysts for asymmetric epoxidation | Team in Brazil | Brazil | sulfinamide | agrochemical synthesis | Tetrahedron Letters |
| 2012 | Visible-light photoredox catalysis using ruthenium complexes | Researchers in China | China | ruthenium complexes | organic synthesis | Journal of Organic Chemistry |
| 2015 | Iron-based catalysts for hydrogenation of alkenes | Group in Russia | Russia | iron-based | petrochemical processing | Russian Journal of Applied Chemistry |
...
```

Bad Case: Entity Fabrication

Context: no UAE in context

Ground Truth Table:

```
| Region        | Hybrid Vehicle Adoption Rate (%) | Charging Stations (2023) | Government Incentives | Major City    | Year of First Major Policy | Public Transit Hybrid Fleet (%) |
|---|---|---|---|---|---|---|
| Japan         | 19.4                             | 30000                    | Yes                   | Tokyo         | 2009                       | 40                              |
| China         | 8.5                              | 180000                   | Yes                   | Beijing       | 2014                       | 35                              |
| South Korea   | 7.1                              | 12000                    | Yes                   | Seoul         | 2012                       | -                               |
| United States | 6.6                              | 56000                    | Varies by state       | Los Angeles   | 2007                       | 28                              |
| Australia     | 6.2                              | 3500                     | Yes                   | Sydney        | 2017                       | 18                              |
| Canada        | 5.8                              | 7000                     | Yes                   | Toronto       | 2016                       | 22                              |
```

Response Table:

```
| Region         | Hybrid Vehicle Adoption Rate (%) | Charging Stations (2023) | Government Incentives | Major City     | Year of First Major Policy | Public Transit Hybrid Fleet (%) |
|---|---|---|---|---|---|---|
| Japan          | 19.4                             | 30000                    | robust                | Tokyo          | 2009                       | 40                              |
| China          | 8.5                              | 180000                   | strong                | Beijing        | 2014                       | 35                              |
| United Kingdom | 9.7                              | 36000                    | provided              | London         | 2010                       | 30                              |
| Sweden         | 8.9                              | 12500                    | N/A                   | Stockholm      | 2012                       | 27                              |
| Germany        | 10.2                             | 44000                    | substantial           | Berlin         | 2011                       | 25                              |
| United States  | 6.6                              | 56000                    | vary by state         | Los Angeles    | 2007                       | 28                              |
| South Korea    | 7.1                              | 12000                    | instrumental          | Seoul          | 2012                       | N/A                             |
| Australia      | 6.2                              | 3500                     | available             | Sydney         | 2017                       | 18                              |
| Canada         | 5.8                              | 7000                     | in place              | Toronto        | 2016                       | 22                              |
| Singapore      | 4.5                              | 1200                     | provided              | Singapore      | 2013                       | 15                              |
| India          | 3.9                              | 2200                     | provided              | Mumbai         | 2015                       | 12                              |
| New Zealand    | 3.2                              | 1000                     | in place              | Auckland       | 2018                       | 10                              |
| UAE            | 2.1                              | 900                      | provided              | Dubai          | 2016                       | 6                               |
```

Bad Case: Order Mismatch

Query:

```
Please generate a table summarizing the autonomous vehicle testing and regulatory status in Japan and the United States, as described in the context. The table should have the following columns in this sequence: "Country", "Testing Start Year", "Number of Testing Cities", "Major Testing Companies", "Regulatory Approval Status", and "First Regulatory Approval Year". Ensure that the table is sorted by "Testing Start Year" in ascending order. Only output a table in Markdown format, without any additional information or explanation.
```

Ground Truth Table:

```
| Country       | Testing Start Year | Number of Testing Cities | Major Testing Companies | Regulatory Approval Status | First Regulatory Approval Year |
|------|-----|-----|-------|-------|--------|-----|
| Japan         | 2013               | 7                        | Toyota                  | Conditional                | 2020                           |
| United States | 2015               | 15                       | Waymo                   | Permitted                  | 2018                           |
| United States | 2016               | 10                       | Cruise                  | Permitted                  | 2021                           |
| Japan         | 2017               | 2                        | Nissan                  | Pilot Only                 | -                              |
| United States | 2018               | 5                        | Aurora                  | Testing Only               | -                              |
```

Response:

```
| Country       | Testing Start Year | Number of Testing Cities | Major Testing Companies | Regulatory Approval Status      | First Regulatory Approval Year |
|------|-----|-----|-------|-------|--------|-----|
| Japan         | 2013               | 7                        | Toyota                  | conditional regulatory approval | 2020                           |
| United States | 2015               | 15                       | Waymo                   | permitted regulatory approval   | 2018                           |
| Japan         | 2017               | 2                        | Nissan                  | pilot-only regulatory approval  | N/A                            |
| United States | 2016               | 10                       | Cruise                  | permitted regulatory approval   | 2021                           |
| United States | 2018               | 5                        | Aurora                  | testing-only regulatory approval| N/A                            |
```

LLM Verifier 2 – Verify the Adversarial Context - Step 2

Please analyze whether the distractor information in the expanded context would compromise the uniqueness of the ground-truth table as the correct answer.

Background:

- The Original Context contains information that can be used to generate the Ground-Truth Table according to the Query requirements
- The Expanded Context includes the Original Context plus additional distractor information
- The goal is to ensure that the distractor information does NOT satisfy the query constraints, so it won't interfere with the Ground-Truth Table being the unique correct answer

Query:
{query}

Query Constraints:
{query_constraints}

Original Context:
{gt_context}

Ground-Truth Table (correct answer derived from Original Context):
{gt_table}

Expanded Context (Original Context + Distractor Information):
{distractor_context}

Please analyze:

1. Identify the distractor information (content in Expanded Context that is NOT present in Original Context)
2. Check if any distractor information satisfies the query constraints (time range, entity scope, data requirements, etc.)
3. Determine if the distractor information could be used to generate a valid alternative table that competes with the Ground-Truth Table

Validation Criteria:

- VALID: Distractor information does NOT satisfy query constraints, ensuring Ground-Truth Table remains the unique correct answer
- INVALID: Distractor information DOES satisfy query constraints, potentially creating competing valid answers

Please answer "VALID" or "INVALID" and provide detailed reasoning.

Format:
Result: [VALID/INVALID]
Reason: [Detailed analysis of distractor information and its impact on answer uniqueness]

Algorithm 1: Hallucination Detection Logic

Require: Answer Table T_A, Ground Truth Table T_{GT}, Context C
Ensure: Hallucination Counts H

▷ *1. Format Check*
if columns of $T_A \neq$ columns of T_{GT} **then**
    $H[\text{Format}] \leftarrow 1$
    return H
end if

▷ *2. Row Matching*
$M, U \leftarrow \text{MATCHROWSBYPRIMARYKEY}(T_A)$ ▷ M: Matched, U: Unmatched

▷ *3. Attribute Matching (for Matched Rows)*
for each matched pair $(r_A, r_{GT}) \in M$ **do**
    for each cell c_A in r_A (except primary key) **do**
        $c_{GT} \leftarrow$ corresponding cell in r_{GT}
        if c_{GT} is EMPTY **and** c_A is NOT EMPTY **then** ▷ Check for Blank Fabrication first
            $H[\text{Blank-Filling}] \leftarrow H[\text{Blank-Filling}] + 1$
        else if c_{GT} is NOT EMPTY **and** c_A does not match c_{GT} **then** ▷ Then check for Attribute Hallucination
            $H[\text{Attribute}] \leftarrow H[\text{Attribute}] + 1$
        end if
    end for
end for

▷ *4. Source Analysis (for Unmatched Rows)*
for each unmatched row $r_U \in U$ **do**
    $key \leftarrow$ primary key of r_U
    if key is in Context C **then**
        $H[\text{Scope}] \leftarrow H[\text{Scope}] + 1$
    else
        $H[\text{EntityFab}] \leftarrow H[\text{EntityFab}] + 1$
    end if
end for
return H