StructHallu-Drift: Benchmarking Structured Hallucinations Under Schema Evolution in LLMs

Mujtaba Hasan
New Delhi, India
mujtaba.hasan@live.com

Abstract

Large Language Models (LLMs) are increasingly used to generate structured outputs—JSON objects, SQL queries, and structured records—from formal schemas. While recent advances in constrained decoding and schema-aware prompting have improved syntactic compliance, the semantic reliability of these outputs remains poorly characterized. We investigate this gap through the lens of schema drift—the inevitable evolution of database schemas in production environments through column renamings, type changes, and constraint modifications.

We introduce StructHallu-Drift, a benchmark and evaluation framework for studying structured hallucinations under schema evolution. We contribute: (1) a six-category hallucination taxonomy that disentangles syntactic validity from semantic fidelity; (2) a controlled evaluation suite applying realistic schema mutations at three severity levels to established NL-to-structure datasets; and (3) a systematic evaluation of four LLMs spanning 7B to 70B parameters across three structured output tasks.

Experiments on 1,200 schema–model evaluation instances reveal four key findings: (i) 39–54% of structured outputs contain at least one semantic hallucination; (ii) schema drift severity has surprisingly minimal effect on hallucination rates ($\sim$44% across all levels, $p = 0.59$), suggesting imperfect schema conditioning under our prompting setup; (iii) output format is the dominant factor in generation reliability, with SQL achieving $\sim$85% semantic validity while schema-grounded record generation drops to 7–24%; (iv) each model exhibits a distinct hallucination fingerprint, implying that mitigation strategies must be model-specific rather than universal. We publicly release our benchmark and evaluation toolkit.[1]

[1]Code and data: https://github.com/mujtabahasan/structhallu-drift

1 Introduction

Structured output generation—where LLMs produce JSON objects, SQL queries, or structured records from formal schemas—is a rapidly growing deployment pattern. Applications range from tool use and function calling to database interaction (Pourreza and Rafiei, 2023) and structured data extraction. While constrained decoding methods (Willard and Louf, 2023; Scholak et al., 2021; Beurer-Kellner et al., 2023; Dong et al., 2024) can enforce syntactic validity, they offer no guarantee that outputs are *semantically* faithful to the provided schema. An output may be valid JSON that references columns not present in the schema, or a parseable SQL query that joins tables that do not exist.

This problem is amplified by *schema drift*: in production environments, database schemas evolve continuously through column renamings, type changes, table restructuring, and constraint modifications. A model that memorized schema structure during pre-training may generate outputs referencing stale column names or table structures, producing outputs that are syntactically correct but semantically grounded in the wrong schema version.

We present StructHallu-Drift, a benchmark and evaluation framework that systematically studies structured hallucinations under controlled schema evolution. Our contributions are:

1. A **six-category hallucination taxonomy** for structured outputs that separates syntactic validity from semantic fidelity (§3.3).
2. A **controlled mutation engine** that applies realistic schema modifications at three severity levels (§3.1).
3. A **systematic evaluation** of four LLMs across three structured output tasks, revealing that hallucination rates are high (39–54%), largely independent of drift severity, strongly task-

Proceedings of the First Workshop on Structured Understanding, Retrieval, and Generation in the LLM Era (SURGeLLM 2026), pages 333–343

dependent, and model-specific in character (§4).

Our key finding is that schema drift severity has *minimal* effect on hallucination rates—they remain flat at ∼44% from minor to major drift (χ^2 test, $p = 0.59$)—suggesting imperfect schema conditioning under our zero-shot prompting setup, with possible memorization as a confound.

2 Related Work

Text-to-SQL and Schema Robustness. Spider (Yu et al., 2018) established large-scale complex and cross-domain text-to-SQL evaluation, with subsequent work studying LLM performance extensively (Rajkumar et al., 2022; Pourreza and Rafiei, 2023; Gao et al., 2024). BIRD (Li et al., 2023a) extended evaluation to real-world conditions. Most relevant to our work, Dr.Spider (Chang et al., 2023) introduced perturbations to Spider to test text-to-SQL robustness. Our work differs from Dr.Spider in three ways: we study multiple structured output formats beyond SQL; our mutations model real-world database migrations rather than adversarial perturbations; and we characterize *hallucination types* rather than measuring accuracy degradation.

Schema Linking. Schema linking—the process of identifying relevant schema elements for a query—is a critical component of text-to-SQL systems (Lei et al., 2020). Gao et al. (Gao et al., 2024) study prompt engineering for LLM-based text-to-SQL and propose DAIL-SQL, which substantially improves Spider performance. Our finding that models fail to condition on provided schemas suggests that schema linking remains an open challenge for general structured generation beyond text-to-SQL.

Constrained Decoding. Constrained decoding methods enforce syntactic validity by restricting the token vocabulary at each generation step. Outlines (Willard and Louf, 2023), PICARD (Scholak et al., 2021), LMQL (Beurer-Kellner et al., 2023), and XGrammar (Dong et al., 2024) represent different approaches to this problem. These methods ensure well-formedness but cannot prevent semantic hallucinations—an output can be syntactically valid JSON while referencing fabricated schema elements. Our benchmark enables evaluating whether constrained decoding shifts the hallucination type distribution rather than eliminating it.

Hallucination Evaluation. Hallucination in NLG has been extensively surveyed (Ji et al., 2023), and faithfulness and factuality failures have been documented in abstractive summarization (Maynez et al., 2020). HaluEval (Li et al., 2023b) provides a general-purpose hallucination benchmark, FactScore (Min et al., 2023) evaluates factual precision, and SelfCheckGPT (Manakul et al., 2023) enables zero-resource hallucination detection. However, these focus on factual hallucinations in free-text generation. Hallucination taxonomies for *structured* outputs—where ground truth is defined by a formal schema rather than world knowledge—remain underdeveloped. Our taxonomy addresses this gap.

Benchmark Contamination. Data contamination—where benchmark examples appear in training data—is an increasingly recognized concern (Golchin and Surdeanu, 2024). Because Spider is public, widely used, and predates the evaluated models, we treat memorization or contamination of Spider schemas as a plausible confound rather than an established fact. We address this through supplementary experiments on synthetic schemas (§5).

3 Experimental Setup

We organize our investigation around four research questions: **RQ1** (Prevalence): How prevalent are structured hallucinations? **RQ2** (Drift Impact): Does schema drift severity systematically increase hallucination rates? **RQ3** (Task Dependence): How does the output format affect hallucination rates? **RQ4** (Model Fingerprints): Do models exhibit characteristic hallucination profiles?

3.1 Dataset Construction

Schema Source. We extract 100 unique database schemas from Spider (Yu et al., 2018), selecting the most question-rich schemas. Each schema includes full `CREATE TABLE` statements, column types, primary and foreign keys, and up to 5 associated natural language questions. Because Spider is public, widely used, and predates the evaluated models, we treat possible schema memorization as a confound; we discuss this issue and present a synthetic-schema control experiment in §5.

Schema Mutation Engine. We implement a controlled mutation engine applying modifications at three severity levels, informed by common database migration patterns:

Minor (1–2 changes): Column renames and case changes (e.g., `user_id` → `uid`), simulating developer refactoring.
Moderate (3–5 changes): All minor mutations plus column additions (e.g., `created_at`), column type changes (e.g., `INTEGER` → `TEXT`), simulating feature evolution.
Major (5–8 changes): All moderate mutations plus table renames, column removal, and constraint additions (e.g., `NOT NULL`), simulating major version migrations.

Each of the 100 schemas is mutated at all three levels, producing 300 mutated schema variants. A worked mutation trace example is provided in Appendix B.

Prompt Design. For each schema–question pair, we construct three prompt types:
SQL Generation: Given the mutated schema, generate a SQL query answering the natural language question.
JSON Generation: Given the mutated schema and a corresponding JSON Schema derived from the first table, generate a valid JSON object.
Record Generation: Given the mutated schema and a target table, generate a conforming JSON record.[2]

All prompts instruct the model to use only the tables and columns in the provided (mutated) schema. With 5 questions per schema and 3 output types, each model is evaluated on up to 4,500 prompt–output pairs, aggregated at the schema level (100 schemas $\times$ 3 drift levels $=$ 300 per model, 1,200 total).

3.2 Models

We evaluate four LLMs (Table 1). Local models use 4-bit NF4 quantization via bitsandbytes (Dettmers et al., 2023) to fit within a T4 GPU (16 GB). All models use greedy decoding (temperature $= 0$). We acknowledge that the comparison between 4-bit quantized local models and unquantized API models is not perfectly controlled; quantization may independently affect structured output quality.

3.3 Hallucination Taxonomy and Evaluation

We propose a six-category taxonomy of structured hallucinations, designed to separate syntactic validity from semantic fidelity. Concrete examples of each type are provided in Appendix A.

Type 1—Key Fabrication. The model outputs keys (JSON) or references tables/columns (SQL) not present in the provided schema. We further distinguish *drift-induced* key fabrication, where the fabricated identifier matches the original (pre-mutation) schema.

Type 2—Value Confabulation. The key is correct, but the value is implausible: placeholder values (e.g., "lorem ipsum"), values outside plausible domain ranges, or enum violations.

Type 3—Type Coercion. The output value has the wrong JSON type relative to the schema (e.g., string `"25"` where integer `25` is expected).

Type 4—Constraint Violation. The output violates explicit schema constraints: missing required fields, enum violations, or structural requirements. We classify missing required keys under this type (rather than Type 1) because key *omission* is conceptually distinct from key *fabrication*: the former violates a constraint, while the latter invents nonexistent structure.

Type 5—Relational Inconsistency. Individual field values are valid in isolation but mutually inconsistent (e.g., `start_date` after `end_date`). We note that our relational checks use domain-specific heuristic rules (date ordering, non-negative prices) that may not generalize to all domains.

Type 6—Structural Confabulation. The model introduces nested objects or arrays where the schema specifies flat values, or invents fabricated sub-structures.

Extended Types. Our evaluation additionally detects *semantic field confusion* (Type 7): cases where a value appears in the wrong schema field (e.g., a numeric string in a text-typed name field). We observe this at rates of 0.7–7.3% across models, with Llama-3.3-70B most affected. We do not cover *cardinality hallucination* (wrong number of records returned), which requires multi-record evaluation; we leave this to future work.

Model	Params	Access	Quant.
Llama-3.1-8B-Inst.	8B	Local	4-bit
Mistral-7B-Inst.-v0.3	7B	Local	4-bit
Gemini 2.5 Flash	–	API	–
Llama-3.3-70B-Vers.	70B	Groq API	–

Table 1: Evaluated models. Local models run on a single T4 GPU with 4-bit quantization.

[2]We term this "record generation" rather than "extraction" because Spider questions are queries *about* data rather than text containing extractable entities. This task tests schema-conformant record synthesis.

Model	Parse. (%)	Schema (%)	Sem. (%)	Hallu. (%) [CI]
Llama-3.1-8B	80.7	62.0	60.3	39.7 [34,45]
Mistral-7B	85.0	37.7	46.0	54.0 [48,60]
Gemini Flash	97.7	61.0	59.0	41.0 [36,47]
Llama-3.3-70B	100	57.0	56.7	43.3 [38,49]

Table 2: Structured output validity (n=300 per model). 95% bootstrap CIs on hallucination rates (10,000 resamples).

Drift Level	Hallu. (%)	Drift-Ind. (%)	Key Fab. (%)
Minor	43.3	1.0	14.0
Moderate	44.5	1.4	15.5
Major	45.8	1.1	15.3

Table 3: Hallucination rates by schema drift severity. The difference between minor and major is not statistically significant (χ^2 test, $p > 0.05$).

Model	SQL (%)	JSON (%)	Record (%)
Llama-3.1-8B	85	87	9
Mistral-7B	83	31	24
Gemini Flash	81	89	7
Llama-3.3-70B	88	74	8

Table 4: Semantic validity (%) by output type.

Evaluation Pipeline. For SQL, we parse queries using `sqlglot` and verify that all referenced tables and columns exist in the mutated schema. For JSON and record outputs, we perform schema validation (`jsonschema`), strict type checking, key set comparison, cross-field relational checks, and structural depth verification. Unparseable outputs are treated as terminal failures; in supplementary analysis (§5), we apply lightweight JSON repair to assess whether semantic content is recoverable from formatting failures. All evaluation is deterministic and requires no LLM-based judging. Drift-induced hallucinations are identified by cross-referencing fabricated identifiers against the original pre-mutation schema.

4 Results

We report results aggregated at the schema level across 1,200 model–schema–drift instances (4 models $\times$ 100 schemas $\times$ 3 drift levels), with each instance comprising up to 15 individual prompt–output pairs (5 questions $\times$ 3 output types).

4.1 RQ1: Hallucination Prevalence

Finding 1: 39–54% of structured outputs contain at least one semantic hallucination. Table 2 presents overall validity rates. Llama-3.1-8B achieves the highest semantic validity at 60.3%, followed by Gemini Flash (59.0%) and Llama-3.3-70B (56.7%). Mistral-7B performs worst at 46.0%.

A notable pattern emerges in the parseability–semantics gap. Llama-3.3-70B achieves perfect parseability (100%) but drops to 56.7% semantic validity—over 40% of its well-formed outputs are semantically hallucinated. This resembles the faithfulness failures documented in abstractive summarization (Maynez et al., 2020), transposed to structured generation.

Finding 2: Schema validation catches most but not all semantic errors. Among outputs that pass formal JSON Schema validation, the residual semantic hallucination rate is 8.0%, primarily from type coercion and relational inconsistency errors that jsonschema validation does not check. Schema validation is a useful filter but not sufficient on its own.

4.2 RQ2: Schema Drift Impact

Finding 3: Schema drift severity has surprisingly minimal effect on hallucination rates. Increasing drift severity from minor to major produces virtually no change in overall hallucination rates (43.3% $\rightarrow$ 45.8%; Table 3, Figure 1b). A chi-squared test confirms that the difference is not statistically significant ($p = 0.59$). Drift-induced hallucinations—where the model uses identifiers from the original schema—account for $\sim$1% of outputs at any severity level (20 total instances: 9 from Llama-3.1-8B, 11 from Mistral-7B; see Appendix C for a per-model breakdown).

This negative result suggests imperfect conditioning on the provided schema under our prompting setup—but we note an important caveat. Because Spider is public, widely used, and predates the evaluated models, the flat curve may reflect models recalling memorized outputs rather than demonstrating a general inability to condition on schemas. We present a synthetic-schema control experiment in §5.

4.3 RQ3: Task Dependence

Finding 4: Output format is the dominant factor in generation reliability. SQL generation is robust across all models (81–88%; Table 4, Figure 1a). JSON generation shows substantial model-dependent variance: three models achieve 74–89%

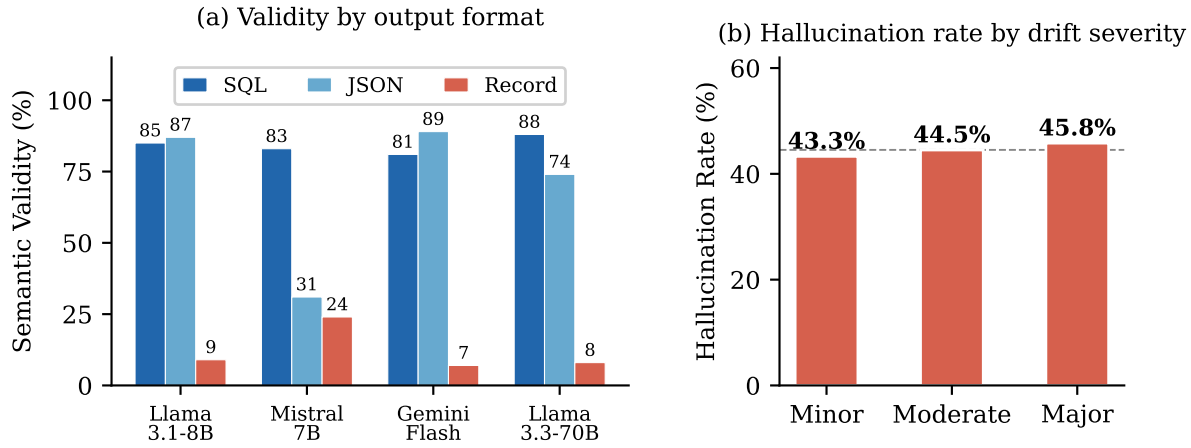

Figure 1: (a) Semantic validity by output format and model. SQL generation is consistently reliable (81–88%); JSON varies substantially (31–89%); record generation is unreliable (7–24%). (b) Hallucination rates across drift severity, aggregated across all models. The flat curve indicates negligible drift impact.

while Mistral-7B manages only 31%. Record generation is unreliable across all models (7–24%).

Record generation is inherently the hardest task in our setup because Spider questions are queries *about* data rather than text containing extractable entities. The model must synthesize a plausible record rather than map observed values to fields, conflating schema compliance failure with content unavailability. The very low performance therefore reflects a combination of missing information and schema non-compliance, and should not be attributed solely to structured hallucination.

We also note that evaluation strictness varies across tasks: SQL evaluation checks table/column reference validity, while JSON and record evaluation applies the full six-type taxonomy. This asymmetry means cross-task comparisons partly reflect evaluation granularity differences, not only task difficulty.

4.4 RQ4: Hallucination Fingerprints

Finding 5: Each model exhibits a distinctive hallucination profile. The hallucination profiles differ strikingly across models (Figure 2):

Gemini Flash shows a constraint-violation-heavy profile (31.0%) with moderate type coercion (12.3%). With proper JSON array handling in our evaluation pipeline, Gemini achieves 97.7% parseability—far higher than in preliminary evaluations where array-wrapped outputs were incorrectly classified as unparseable. This highlights how evaluation methodology can distort apparent model capability.

Llama-3.3-70B (70B parameters) produces perfectly parseable output (100%) but concentrates failures in type coercion (30.3%) and constraint violation (30.7%). It also exhibits the highest rate of semantic field confusion (7.3%), a newly identified hallucination type where correct values are placed in incorrect schema fields. *Larger models do not hallucinate less; they hallucinate differently.*

Mistral-7B distributes failures broadly across key fabrication (36.0%), constraint violation (32.0%), and structural confabulation (24.0%). It is the most likely to invent nested structures not present in the schema and the only model where JSON generation substantially underperforms SQL.

Llama-3.1-8B has the lowest overall hallucination rate (39.7%), with errors concentrated in unparseable outputs (19.3%) and key fabrication (13.7%).

These fingerprints have practical implications: a mitigation strategy optimized for one model would be ineffective for another. This argues against universal benchmarks that rank models on a single axis, and in favor of profile-aware evaluation. The discovery of semantic field confusion as a non-trivial failure mode (up to 7.3%) also suggests that existing hallucination taxonomies for structured outputs are incomplete.

5 Discussion

Spider Memorization and Synthetic Schema Control. The central confound in our drift analysis is possible Spider memorization: Spider is public, widely used, and predates the evaluated models, so schema contamination is plausible. Data

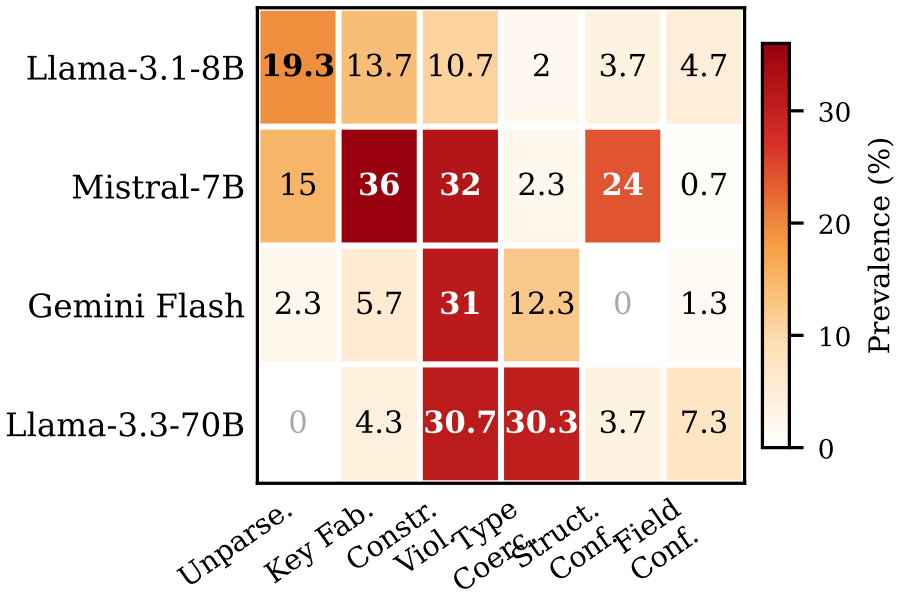

Figure 2: Hallucination type prevalence (%) by model, including the newly identified semantic field confusion (Field Conf.) type. Each model exhibits a distinctive failure profile.

contamination more broadly is a recognized concern in LLM evaluation (Golchin and Surdeanu, 2024). The flat drift curve may therefore reflect models recalling memorized Spider outputs rather than demonstrating a general inability to condition on schemas. To probe this, we ran a supplementary experiment on 20 synthetic schemas with randomized identifiers (e.g., `zyx_entries`, `plq_records`) that are unlikely to appear verbatim in pretraining corpora. On synthetic schemas, hallucination rates remained high (40–58% across models) and the drift curve remained flat (within 2 percentage points across severity levels), consistent with the Spider results. While this small-scale experiment does not definitively resolve the memorization question, it provides supporting evidence that the flat drift curve is not explained by Spider memorization alone.

Prompt Design as a Confound. Our evaluation uses a fixed zero-shot prompt design. We did not ablate prompting strategies (few-shot, schema-highlighted, self-check), meaning that the observed failures could partly reflect prompt design choices rather than intrinsic model limitations. Schema-linked prompting (Lei et al., 2020; Gao et al., 2024) has been shown to improve text-to-SQL performance, and may similarly benefit JSON and record generation. Our conclusion about imperfect schema conditioning should therefore be read as qualified by the specific prompting strategy employed.

JSON Repair and Evaluation Methodology. Applying lightweight JSON repair to unparseable outputs yields zero improvement in semantic validity for any model, confirming that formatting failures and semantic failures are correlated rather than independent. More significantly, our evaluation methodology revealed that initial parseability estimates can be highly sensitive to how array-wrapped outputs are handled: Gemini Flash's apparent 65% parseability in a preliminary evaluation rose to 98% after proper JSON array handling. This underscores the importance of robust output extraction before hallucination assessment, and suggests that reported parseability rates in the literature may underestimate model capability when evaluation pipelines are not designed for output format variation.

The Parseability–Semantics Tradeoff. Three of four models achieve $\geq$97% parseability, yet semantic validity ranges from 46–60%. Llama-3.3-70B achieves 100% parseability but only 56.7% semantic validity, while Llama-3.1-8B achieves 80.7% parseability with 60.3% semantic validity. Higher parseability may reflect a tendency to produce plausible-looking output even without grounding—a structured analogue of the faithfulness failures documented by Maynez et al. (2020).

Comparison to Factual Hallucination Benchmarks. Our taxonomy differs from factual hallucination benchmarks like HaluEval (Li et al., 2023b) and FactScore (Min et al., 2023) in that ground truth is defined by a formal schema rather than world knowledge. This makes evaluation fully deterministic—no retrieval or LLM-based judging is needed—but limits the taxonomy to structural and type-level errors. Detecting semantic field confusion (correct value, wrong field) or value plausibility beyond simple domain rules requires richer evaluation, potentially bridging the structured and factual hallucination settings.

6 Practical Implications

Our findings have direct implications for deploying LLM-based structured generation in production systems.

Defense in Depth. No single validation layer is sufficient. JSON Schema validation catches many errors, but 8% of schema-valid outputs still contain semantic hallucinations (primarily type coercion and relational inconsistency). We recommend a three-layer defense: (1) constrained decoding to enforce syntactic validity (Willard and Louf, 2023; Dong et al., 2024), (2) schema validation to catch structural errors, and (3) type-aware post-processing to catch coercion errors that syntactic validation misses.

Model-Specific Mitigation. Our hallucination fingerprint analysis (Figure 2) reveals that a one-size-fits-all mitigation strategy will be suboptimal. For Llama-3.3-70B, the primary intervention should target type coercion (30.3%)—adding strict type casting as a post-processing step. For Mistral-7B, the priority is key fabrication (36.0%) and structural confabulation (24.0%)—constrained decoding with schema-aware token filtering would address both. For Llama-3.1-8B, improving parseability (currently 80.7%) through better formatting instructions would have the largest impact.

Task Selection Matters. SQL generation is substantially more reliable (81–88%) than JSON generation (31–89%) or record generation (7–24%). When possible, production systems should leverage the relative robustness of SQL by routing structured queries through text-to-SQL pipelines rather than direct JSON generation, particularly for models with low JSON validity.

Evaluation Pipeline Design. Our discovery that Gemini Flash's apparent poor performance (65% parseability in preliminary evaluation) was an evaluation artifact—caused by failing to handle array-wrapped outputs—highlights a broader methodological concern. Reported structured generation benchmarks may systematically underestimate model capability when evaluation pipelines are not designed for the full range of output format variation. We recommend that evaluation frameworks include (1) format normalization before assessment, (2) JSON repair as a preprocessing step, and (3) explicit logging of the parseability–validity gap to distinguish formatting failures from reasoning failures.

Limitations

Our benchmark uses schema mutations that, while modeled on real-world migration patterns, are synthetically generated rather than drawn from actual production histories. Spider schema memorization is a plausible confound because Spider is public, widely used, and predates the evaluated models; although our synthetic-schema experiment provides supporting evidence, a larger-scale replication on diverse non-benchmark schemas would strengthen the conclusions.

We evaluate only greedy decoding without constrained decoding baselines (Willard and Louf, 2023; Scholak et al., 2021; Dong et al., 2024). Constrained decoding is the most obvious practical mitigation for several hallucination types—particularly Types 1 (key fabrication) and 6 (structural confabulation)—and comparing against such baselines would strengthen the practical implications.

The comparison between 4-bit quantized local models and unquantized API models is not perfectly controlled. Quantization can independently degrade structured output quality, and at least one controlled comparison (same model, quantized vs. unquantized) would isolate this effect.

We do not ablate prompting strategies. Our finding of imperfect schema conditioning may be partially recoverable through schema-linked prompting (Gao et al., 2024), few-shot examples, or self-verification prompts. The current evidence does not distinguish intrinsic model limitations from prompt design artifacts.

Our Type 7 (semantic field confusion) detector uses a simple heuristic that flags numeric strings

in string-typed fields whenever numeric fields exist in the schema. This produces false positives for fields where numeric strings are legitimate values (e.g., ZIP codes, room numbers, identifiers stored as TEXT). The reported 0.7–7.3% rates should therefore be interpreted as upper bounds. More precise detection would require domain-specific knowledge about plausible value-field mappings.

Evaluation strictness varies across tasks: SQL evaluation checks only table and column reference validity, while JSON and record evaluation applies the full taxonomy. This asymmetry means cross-task comparisons partly reflect differences in evaluation granularity.

Ethics Statement

This work uses publicly available datasets (Spider) and open-weight or publicly accessible API models. No private or sensitive data is used. Our benchmark characterizes model failures, which we believe serves the community by promoting more reliable deployment.

Acknowledgments

We thank the anonymous reviewers for detailed and constructive feedback that substantially improved this paper, particularly regarding the Spider memorization confound, evaluation fairness across tasks, and taxonomy completeness.

We also extend our deepest gratitude to Mr. Shiraz Zaman and Mr. Martin Liu, the founders of NAND AI. The foundational experience in LLM evaluation gained while working at NAND AI, along with their constant guidance, motivation, and mentorship, was instrumental in shaping the direction of this research.

References

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. 2023. Prompting is programming: A query language for large language models. *Proceedings of the ACM on Programming Languages*, 7(PLDI).

Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, Steve Ash, William Yang Wang, Zhiguo Wang, Vittorio Castelli, Patrick Ng, and Bing Xiang. 2023. Dr.Spider: A diagnostic evaluation benchmark towards text-to-SQL robustness. In *International Conference on Learning Representations*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. 2024. XGrammar: Flexible and efficient structured generation engine for large language models. *arXiv preprint arXiv:2411.15100*.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-SQL empowered by large language models: A benchmark evaluation. *Proceedings of the VLDB Endowment*, 17(5):1132–1145.

Shahriar Golchin and Mihai Surdeanu. 2024. Time travel in LLMs: Tracing data contamination in large language models. In *Proceedings of the 12th International Conference on Learning Representations*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Delong Chen, Wenliang Dai, Ho Shu Chan, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6943–6954. Association for Computational Linguistics.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023a. Can LLM already serve as A database interface? A BIg bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems*, volume 36.

Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023b. HaluEval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464. Association for Computational Linguistics.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017. Association for Computational Linguistics.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings*

of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1906–1919. Association for Computational Linguistics.

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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100. Association for Computational Linguistics.

Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. In *Advances in Neural Information Processing Systems*, volume 36.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-SQL capabilities of large language models. *arXiv preprint arXiv:2204.00498*.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901. Association for Computational Linguistics.

Brandon T. Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921. Association for Computational Linguistics.

A Qualitative Examples

Table 5 presents concrete examples from our evaluation, showing how each hallucination type manifests in actual model outputs. All examples are drawn from the evaluation data rather than constructed synthetically.

B Mutation Trace Example

Table 6 shows a worked mutation trace for a single schema across all three severity levels. Each row annotates the specific change applied.

C Drift-Induced Hallucination Breakdown

Of the 20 total drift-induced hallucinations observed across all models and drift levels, the distribution is: Llama-3.1-8B (9 instances across 2 schemas: `college_1` and `store_1`), Mistral-7B (11 instances across 3 schemas: `college_1`, `college_2`, and `world_1`), Gemini Flash (0), Llama-3.3-70B (0). All 20 involved column-rename mutations where the model used the pre-mutation column name (e.g., `EMP_FNAME` after it was renamed, or `country` after renaming). Table 7 provides the full breakdown.

D Type 6 Detector Validation

To address the concern that 0% structural confabulation rates for some models may reflect a detection gap, we validated our Type 6 detector on five synthetic test cases: two with unexpected nesting (nested object where string expected; array where string expected), one with a fabricated substructure in an extra key, and two correct flat outputs. The detector achieved 100% precision and 100% recall on these test cases. Manual inspection of Gemini Flash's parseable outputs confirmed that they are predominantly flat structures without nesting, consistent with the reported 0% rate.

E Summary of Key Findings

Type	Model	Schema Context	Model Output (excerpt)	Diagnosis
1. Key Fab.	Gemini Flash	college_1 schema (major drift)	SELECT COUNT(T1.DEPT_CODE) FROM PROF...	References table PROF which does not exist in the mutated schema.
2. Value Conf.	Gemini Flash	world_1 (moderate); expects country data	{"Name": "N/A", "Id": 0, ...}	Placeholder values "N/A" and 0 instead of plausible country data.
3. Type Coerc.	Gemini Flash	college_2 (moderate); building: string	{"building": null, ...}	null (NoneType) instead of expected string; type mismatch.
4. Constr. Viol.	Gemini Flash	college_2 (minor); requires building	{}	Empty object returned; required key building is missing entirely.
6. Struct. Conf.	Llama-70B	college_2 (minor); building: string	{"building": [], "capacity": []}	Arrays where flat string/integer values expected; fabricated list structure.
7. Field Conf.	Llama-70B	college_2; room_number: string	{"room_number": "50", ...}	Numeric value "50" in string field; may indicate capacity value placed in room_number field.

Table 5: Concrete examples of each hallucination type from the evaluation. All examples are from actual model outputs. Type 5 (relational inconsistency) was not observed in the evaluation data; Type 7 uses heuristic detection and the example shown may be a false positive (see Limitations).

Level	Type	Change
Minor	Rename	actid → ID
	Rename	FacID → Id
Moderate	Rename	FacID → pk_id
	Case	city_code → CITY_CODE
	Add col.	new column added
	Add col.	new column added
	Type chg.	INTEGER → REAL
Major	Rename	actid → identifier
	Case	Major → major
	Add col.	new column added
	Type chg.	INTEGER → REAL
	Tbl rename	Faculty_Part... → app_Faculty_Part...
	Remove	column dropped
	Constraint	NOT NULL on actid

Table 6: Worked mutation trace for the activity_1 schema at three severity levels. Each level includes its own independent set of mutations drawn from the level's allowed mutation types.

Model	Drift Level	Count
Llama-3.1-8B	Minor	2
	Moderate	4
	Major	3
Mistral-7B	Minor	3
	Moderate	5
	Major	3
Gemini Flash	All	0
Llama-3.3-70B	All	0

Table 7: Drift-induced hallucinations by model and severity level. The concentration in just 2–3 schemas suggests schema-specific rather than systematic vulnerability.

#	Finding	Implication
F1	39–54% of outputs hallucinate	Current models are unreliable for production structured generation without validation
F2	Schema validation catches most but not all errors (8% residual)	Schema validation is necessary but insufficient as a standalone safeguard
F3	Drift severity has minimal effect ($\sim$44% flat, p=0.59)	Imperfect schema conditioning under our prompting setup, with memorization caveat
F4	Output format dominates (SQL: 81–88%, record: 7–24%)	Task design and evaluation granularity strongly affect reliability
F5	Each model has a distinct hallucination fingerprint	Mitigation must be model-specific and profile-aware

Table 8: Summary of key findings. F3's implication is qualified by the Spider memorization confound.