Hallucination Detection in Structured Query Generation via LLM Self-Debating

Miaoran Li[†]
Iowa State University
limr@iastate.edu

Jiangning Chen
Splunk
jiangningc@splunk.com

Minghua Xu
Splunk
mingx@splunk.com

Xiaolong Wang
Splunk
xiaolongw@splunk.com

Abstract

Hallucination remains a key challenge in applying large language models (LLMs) to structured query generation, especially for semi-private or domain-specific languages underrepresented in public training data. In this work, we focus on hallucination detection in these low-resource structured language scenarios, using Splunk Search Processing Language (SPL) as a representative case study. We start from analyzing real-world SPL generation to define hallucination in this context and introduce a comprehensive taxonomy. To enhance detection performance, we propose the SELF-DEBATING framework, which prompts an LLM to generate contrastive explanations from opposing perspectives before rendering a final consistency judgment. We also construct a synthetic benchmark, SynSPL, to support systematic evaluation of hallucination detection in SPL generation. Experimental results show that SELF-DEBATING consistently outperforms LLM-as-a-Judge baselines with zero-shot and chain-of-thought (CoT) prompts in SPL hallucination detection across different LLMs, yielding 5–10% relative gains in hallucination F1 scores on both real and synthetic datasets, and up to 260% improvement for LLaMA-3.1–8B. Besides hallucination detection on SPL, SELF-DEBATING also achieves excellent performance on the FaithBench benchmark for summarization hallucination, demonstrating the strong generalization ability of SELF-DEBATING, with OpenAI o1-mini achieving state-of-the-art performance. All these results consistently demonstrate the strong robustness and wide generalizability of SELF-DEBATING.

1 Introduction

Large language models (LLMs) have shown remarkable success in generating structured queries and code from natural language inputs, particularly in well-resourced domains such as SQL and Python. These languages benefit from extensive public documentation, training corpora, and community support, enabling LLMs to learn them with high fidelity. However, many real-world applications involve domain-specific or semi-private languages that are less prevalent in public data, such as Splunk Search Processing Language (SPL)[1], New Relic Query Language (NRQL)[2], and Kusto Query Language (KQL)[3]. These languages are widely used in practice but lack large-scale open datasets and standardized benchmarks. This disparity creates unique challenges for LLM-based generation, particularly with respect to hallucination detection.

In this work, we study hallucination in structured query generation for such semi-private languages, using SPL as a representative case. SPL is a domain-specific query language used to retrieve, filter, and analyze machine-generated log data within the Splunk platform. It features a rich and expressive syntax that supports complex search, aggregation, and transformation operations, drawing inspiration from both the UNIX pipeline and SQL. Despite its power and industrial adoption, authoring SPL queries remains challenging, both due to its steep learning curve and the scarcity of publicly available guidance or training data.

With the rapid advancement of LLMs, there is growing interest in using these models to assist users in generating SPL queries from natural language inputs, mirroring successful applications in other domains, such as text-to-SQL and code generation. However, while LLMs demonstrate impressive fluency and generalization ability, they often suffer from hallucination. They can produce outputs that are syntactically valid but semantically

[†]This work was done during an internship at Splunk.

[1]https://docs.splunk.com/Splexicon:SPL

[2]https://docs.newrelic.com/docs/nrql/get-started/introduction-nrql-new-relics-query-language/

[3]https://learn.microsoft.com/en-us/kusto/query

incorrect or unsupported by the input. This undermines the reliability of LLM-generated SPL queries. Despite the critical nature of this problem, hallucination in SPL generation remains an underexplored area. To the best of our knowledge, no systematic study or benchmark exists for hallucination detection in SPL. This work aims to fill this gap by investigating the types, prevalence, and detection strategies of hallucination in LLM-generated SPL queries.

In this work, we begin by formally defining hallucination in the context of SPL and introduce a taxonomy that categorizes common types of hallucinated behaviors observed in LLM-generated queries. We propose the SELF-DEBATING framework for hallucination detection, drawing inspiration from recent research on multi-agent debate (Liang et al., 2024; Estornell and Liu, 2024; Chan et al., 2024; Hu et al., 2025). While prior methods typically involve multiple LLMs or agents engaging in back-and-forth discussion to reach a consensus, SELF-DEBATING adopts a more lightweight and self-contained approach. It leverages contrastive reasoning within a single LLM, prompting it to generate explanations for both the consistent and hallucinated interpretations of a generated query before making a final judgment. To support systematic evaluation, we construct a synthetic SPL benchmark, SynSPL, which serves both as a testbed for our framework and as a reusable resource for future research on hallucination mitigation in structured query generation. Experimental results show that SELF-DEBATING consistently outperforms LLM-as-a-Judge baselines using standard zero-shot and chain-of-thought (CoT) prompts across six different LLMs. To assess the generalizability of our method beyond SPL, we further evaluate SELF-DEBATING on FaithBench (Bao et al., 2025), a challenging benchmark for hallucination detection in summarization tasks. SELF-DEBATING achieves consistent gains in F1-Macro across all six tested LLMs, with four models showing over 30% relative improvement. Notably, when applied to the o1-mini model, SELF-DEBATING not only surpasses the original baseline performance of o1-mini on FaithBench, but also achieves state-of-the-art (SOTA) results among all evaluated models.

2 Hallucinations in SPL Generation

The SPL generation task involves producing a valid and grounded SPL query based on a given generation prompt. This prompt typically includes system instruction and current user query.

Definition of Hallucination in SPL A generated SPL is considered consistent if it meets both of the following criteria: 1) All structural components of the query must be explicitly supported by the information provided in the generation prompt. These components can include the `index`, which specifies the dataset to be searched; the `sourcetype`, which defines the format or classification of the data; the `source`, referring to the origin of the data (e.g., file path or data stream); the `fields`, which represent the extracted attributes from the data. 2) The query is free from syntax errors and adheres to the correct SPL syntax. If either of these conditions is violated, the SPL query is considered hallucinated.

2.1 Data Collection

The data collection process consists of two stages. In the first stage, we compile 25 user queries representative of real-world scenarios and use them to prompt eight large language models (LLMs): o1-mini, GPT-4o, LLaMA-3.1-8B, LLaMA-3.1-70B, Phi-4, Mistral 8x22B, DeepSeek-R1-Distill-Llama-8B, and DeepSeek-R1-Distill-Llama-70B. This results in a total of 200 (prompt, SPL) pairs for analysis. In the second stage, two annotators with expertise in SPL and hallucination detection independently annotate the generated data. During the first round, each annotator assigns a binary label indicating whether the SPL query contains hallucination. In the second round, they classify the type of hallucination if one is present. Any disagreements are resolved through discussion, and if consensus cannot be reached, a third annotator serves as the adjudicator to make the final decision on the label and hallucination category.

2.2 Analysis

2.3 Hallucination Taxonomy

Hallucinations in SPL generation can be categorized into five distinct types. Illustrative examples for each type are provided in Appendix A.

Fabrication An SPL query is considered a fabrication hallucination if it introduces an index, sourcetype, source, field, or lookup table that is absent from the retrieved metadata and lacks any supporting evidence across the provided prompt.

Mixed Information An SPL query is considered to contain a mixed information hallucination if it

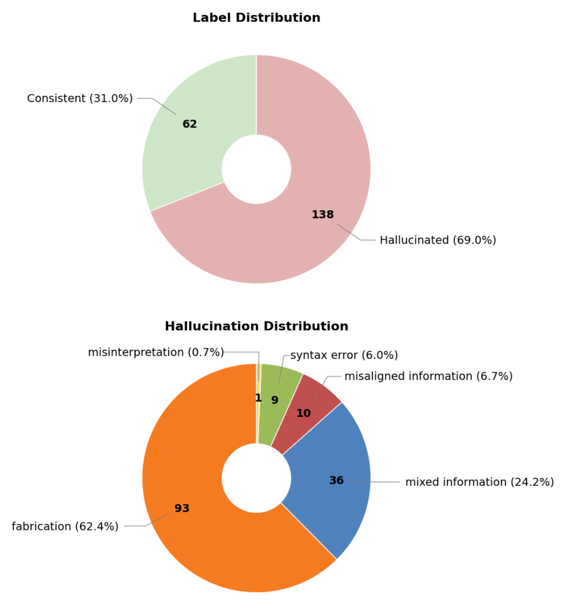

Figure 1: Distribution illustration of hallucinations in collected SPL data. The top chart shows the overall label distribution; the bottom breaks down hallucinated queries by type. A single query may exhibit multiple types, so counts may exceed 138.

combines individual components, such as index, sourcetype, or fields, that are each supported by the prompt, but the combination itself is not supported.

Misaligned Information An SPL query is considered to contain a misaligned information hallucination if it includes field names and field values that are individually supported by the prompt, but the association between a field name and its assigned value is incorrect or unverifiable based on the prompt. In contrast to mixed information hallucinations, which result from combining valid components, such as index, sourcetype, fields, from different parts of the prompt in an unsupported way, misaligned information specifically refers to invalid pairings of field names and values, even when all elements are mentioned in the prompt.

Syntax Error An SPL query is considered to contain a syntax error hallucination if it includes syntactic issues making the query invalid or not executable. This may involve invalid or misspelled commands, missing or mismatched quotation marks, incorrect field references, or improper command structure or ordering.

Misinterpretation An SPL query is considered to contain a misinterpretation hallucination if it introduces additional conditions, filters, or requirements that are not supported by or inferable from the user's request.

The label and hallucination type distributions of the collected SPL data are presented in Figure 1. The results indicate that hallucination is a widespread issue in SPL generation. Out of 200 evaluated (prompt, SPL) pairs, 138 queries (69%) were identified as hallucinated, suggesting that the LLM-generated SPL queries are at high risk of producing ungrounded or inaccurate outputs in real-world settings. A breakdown of hallucination types reveals that fabrication is the most common category, accounting for 62.4% of all hallucination instances. This suggests that LLMs frequently make up unsupported SPL components that are not supported by the input prompt. The second most common type is mixed information, which constitutes 24.2% of the hallucinations. In these cases, the model selects components that are individually present in the prompt but combines them in an invalid manner, indicating difficulties in preserving logical coherence across supported elements. In contrast, misinterpretation of the user query is relatively rare, occupying only 0.7% of the hallucinated cases. This suggests that LLMs are generally able to adhere to the user's intent without introducing extraneous or ungrounded constraints.

Model	Hallucination Rate
Mistral 8x22B	84.0
Phi-4	68.0
LLaMA-3.1-8B	64.0
LLaMA-3.1-70B	52.0
Deepseek-R1-8B	84.0
Deepseek-R1-70B	72.0
GPT-4o	56.0
o1-mini	68.0

Table 1: Hallucination rates of different models. Lower values indicate better performance.

To further understand model-level differences, we compute the hallucination rate of each LLM on the collected SPL dataset. As shown in Table 1, the hallucination rate varies significantly across models. LLaMA-3.1–70B achieves the lowest hallucination rate at 52.0%, followed closely by GPT-4o at 56.0%. In contrast, models such as Mistral and DeepSeek-R1–8B exhibit much higher hallucination rates of 84.0%, indicating a greater tendency to produce unsupported or inaccurate SPL components. These findings highlight that hallucination remains a critical and model-dependent challenge in SPL generation, reinforcing the strong need of introducing hallucination detection and mitigation

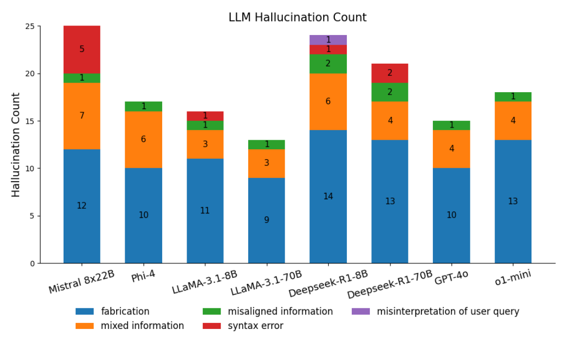

Figure 2: Distribution illustration of hallucination types by LLM. Each bar shows the count of hallucination types per model. A single SPL query may exhibit multiple types and contribute to multiple categories.

in SPL generation problem.

Figure 2 further reveals that different LLMs exhibit distinct hallucination tendencies in terms of the types of errors they are more prone to generate. For instance, Mistral 8×22B shows a disproportionate tendency to produce syntax errors compared to other models, indicating potential instability in adhering to SPL's structural constraints or lack of relevant knowledge. In contrast, DeepSeek-R1–8B and DeepSeek-R1–70B both exhibit more diverse hallucination patterns, with a higher likelihood of producing misaligned information or even rare misinterpretations, suggesting difficulty in consistently mapping prompt semantics to correct field values. LLaMA-3.1–8B and LLaMA-3.1–70B, while similar in architecture, diverge slightly in their hallucination behavior, with the 70B variant avoiding syntax and misinterpretation errors entirely, hinting at the stabilizing effect of scale. Meanwhile, GPT-4o and o1-mini demonstrate more focused tendencies, primarily producing fabrication and mixed information errors, suggesting generally strong syntactic control but limitations in faithfully grounding content. These model-specific patterns highlight that hallucination is not only a matter of frequency, but also of characteristic failure modes, which may reflect differences in training regimes, model alignment, or architectural choices.

3 Self-Debating Framework for Hallucination Detection

We propose the Self-Debating framework for hallucination detection, which enables large language models (LLMs) to assess the faithful consistency of generated content through structured contrastive reasoning. An overview of the framework is provided in Figure 3, and a running example is included in Appendix B. This framework can be generalized to other problems in which an LLM produces a response conditioned on an input prompt, such as structured queries, summaries, answers, or other generation tasks. It operates in two stages: explanation generation and final judgment. By prompting the model to explicitly consider and contrast both consistent and hallucinated interpretations of the output, the framework fosters deeper reasoning and leads to more accurate and interpretable consistency assessments.

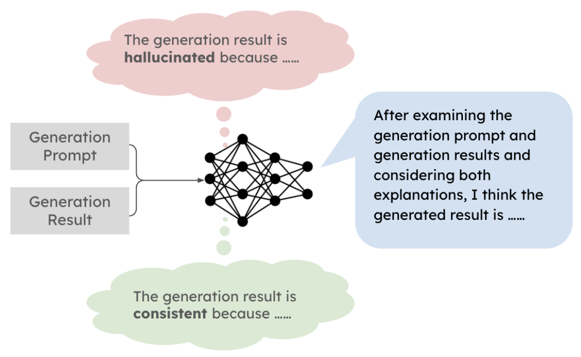
The generation result is **hallucinated** because ......
Generation Prompt
Generation Result
After examining the generation prompt and generation results and considering both explanations, I think the generated result is ......
The generation result is **consistent** because ......

Figure 3: Overview of Self-Debating. Given a generation prompt and a generation result, the model generates two contrastive explanations: one assuming the output is hallucinated and one assuming it is consistent. The model then reflects on both explanations to make a final judgment.

3.1 Explanation Generation

Given an input pair consisting of a generation prompt and a corresponding LLM-generated output, the model is first prompted to produce two contrasting explanations. One explanation is generated under the assumption that the output is hallucinated, highlighting elements that are not supported or verifiable based on the prompt. The other explanation assumes the output is faithfully consistent, offering justification based on alignment with the input context. This contrastive explanation step forces the model to articulate both potential perspectives and prepares it for more nuanced decision-making in the subsequent phase.

3.2 Final Judgement

In the second step, the LLM is required to determine whether the generated output is faithfully consistent with the prompt or hallucinated. It is provided with the generation prompt, the output, and both explanations from the previous step. The model is instructed to read and consider both explanations before making a final decision.

To mitigate potential position bias, such as undue influence from the order in which explanations are presented, the process is executed twice, alternating the order in which the explanations are presented. If the LLM reaches the same conclusion in both runs, that label is accepted as the final prediction. If the decisions diverge, the LLM is prompted three additional times with an increased temperature (we use 0.5 in experiments) to encourage response diversity. The majority label across all runs is then selected as the final decision.

4 SynSPL Dataset Construction

To enable the development and evaluation of hallucination detection methods for SPL generation, a suitable benchmark is needed. However, research on SPL remains limited, and no large-scale benchmark currently exists for this purpose. Collecting real-world queries and associated data is challenging due to privacy concerns and annotation costs. In contrast, text-to-SQL has benefited from extensive benchmarks and community support. Given the conceptual and structural similarities between SQL and SPL, we construct a synthetic benchmark for SPL hallucination detection by adapting and extending an existing text-to-SQL dataset.

We construct the synthetic SPL benchmark using the development set of the BIRD (Li et al., 2023) benchmark as the seed. The BIRD development set spans eleven databases and includes SQL queries across three levels of difficulty: challenging, moderate, and simple. Each sample in the BIRD dataset typically includes a user question, a corresponding golden-standard SQL query, a piece of relevant knowledge required for SQL generation, and the name of target database. The construction process involves two stages. In the first stage, we generate gold-standard (prompt, SPL) pairs free from hallucinations. In the second stage, we systematically inject different types of hallucinations into these gold queries to produce corresponding hallucinated examples.

4.1 Construction of Consistent Pairs

To ensure broader domain coverage and increase the difficulty of the resulting benchmark, we select up to 30 examples per database, prioritizing more complex queries. This results in a total of 330 initial SQL samples, which serve as the seed for constructing the synthetic SPL dataset. To generate golden (prompt, SPL) pairs without hallucinations, we follow a three-step process: (1) SQL-to-SPL translation, (2) prompt synthesis, and (3) post-validation.

We first prompt GPT-4o to translate the selected SQL queries into equivalent SPL queries. To verify semantic equivalence, we then ask GPT-4o to verify that the translated SPL faithfully captures the intent and logic of the original SQL query. Only those examples for which GPT-4o affirms equivalence are retained, yielding 252 verified samples with translated SPL queries. We additionally prompt GPT-4o to extract the fields used in each SPL query. These extracted fields are used both to support hallucination injection in later stages and to provide an additional layer of correctness checking. In particular, challenging SQL queries often involve multiple `JOIN` operations and table aliases, which are not typically used in SPL. This discrepancy can result in translation errors, such as invalid field references like `T1.id` in SPL, where `T1` is a table alias or joint in the SQL query but has no equivalent in SPL. To address this, we manually review the translated SPL queries, paying special attention to field references that may involve aliases, and correct any errors to ensure that all gold-standard examples are both syntactically valid and semantically faithful.

We then construct a synthetic generation prompt to accompany each verified SPL query. The structure of the synthetic prompt mirrors that of a standard prompt used for SPL generation and comprises five components: a general instruction, retrieved SPL metadata, relevant example queries, user history, and the current user query. The general instruction remains fixed across all examples, while the remaining components are query-specific and are the focus of the prompt synthesis process. The SPL metadata section includes a list of relevant indices, each is accompanied by a brief description highlighting representative fields. To simulate this, we treat the tables from the original SQL query's database as corresponding indices, and the columns within each table as potential fields. We prompt GPT-4o to generate index descriptions based on the column definitions in these tables. The relevant example section consists of five (user query, SPL) pairs that are semantically similar to the current query. To identify these examples, we retrieve the top five most similar user queries from the training set of BIRD based on their semantic content and include their corresponding translated SPLs as contextual examples. To simulate user history, we assume that a user is likely to explore related queries within the same domain. Accordingly, we group all user queries in the development set by their associated database and randomly sample five additional queries from the same database along with SPL query translated by GPT-4o as the user history. Finally, the current user query is constructed by concatenating the original user question with its associated supporting knowledge, as provided in the BIRD dataset. This results in a realistic and structured synthetic prompt that closely resembles those used in real-world SPL generation scenarios.

In the final step, we leverage both GPT-4o and o1-mini to independently assess whether each pair is semantically and syntactically consistent. If both models flag a pair as inconsistent, we manually check the pair and correct it if necessary. This multi-step process helps ensure that the resulting dataset contains high-quality, hallucination-free SPL examples that are reliable for subsequent hallucination injection and evaluation.

4.2 Hallucination Injection

To simulate realistic failure cases and enable fine-grained evaluation, we inject four types of hallucinations, including fabrication, syntax error, mixed information, and misaligned information, into the gold-standard (prompt, SPL) pairs, based on our taxonomy. Misinterpretation hallucinations are excluded, as earlier analysis shows they are rare and less impactful for large-scale evaluation.

Hallucinated SPLs are generated by six LLMs: o1-mini, GPT-4o, LLaMA-3.1-8B, LLaMA-3.1-70B, DeepSeek-R1-Distill-Llama-8B, and DeepSeek-R1-Distill-Llama-70B. For each gold example, we prompt each model with the synthetic generation prompt, the gold SPL query, a formal definition of the target hallucination type, and a few-shot demonstration of examples illustrating that type. For fabrication hallucinations, we also include extracted fields from the gold SPL to guide manipulation. The model is instructed to generate a new SPL query that reflects the specified hallucination category while maintaining fluency and structural plausibility.

The final dataset consists of 6,300 examples, comprising 252 consistent examples and 252 hallucinated examples for each combination of hallucination type and injection model.

5 Experiments

5.1 Experimental Setup

We primarily use SynSPL to evaluate the performance of SELF-DEBATING in hallucination detection. To further assess the generalization capability of SELF-DEBATING, we additionally evaluate it on FaithBench(Bao et al., 2025), a challenging benchmark for summarization hallucination detection that features high-quality human annotations. On both datasets, we compare SELF-DEBATING against two prompt-based baselines: zero-shot prompt(Luo et al., 2023) and a CoT prompt following (Jacovi et al., 2025), both of which were previously evaluated on FaithBench. For SynSPL, we report F1 score, precision, and recall for the hallucination class, as hallucination detection is the primary focus. For FaithBench, we follow the original evaluation protocol and report balanced accuracy and macro-averaged F1 score. Additionally, to better understand model performance across both classes, we report class-wise F1 scores for hallucinated and faithfully consistent generations. In this work, we use six popular LLMs as backbone models, including: o1-mini, GPT-4o, LLaMA-3.1–8B, LLaMA-3.1–70B, DeepSeek-R1-Distill-LLaMA–8B, and DeepSeek-R1-Distill-LLaMA–70B.

5.2 Results on SynSPL

As illustrated in Table 2, SELF-DEBATING framework demonstrates consistent performance improvements across all six evaluated LLMs, outperforming both baseline methods, the zero-shot prompt and the CoT prompt, in the majority of cases. The most substantial gains are observed in weaker and mid-tier models. For example, the F1 score of LLaMA-3.1–8B improves significantly from 23.39% with the zero-shot prompt and only 2.55% with CoT to 85.05% under the SELF-DEBATING framework. Similarly, DeepSeek-R1–8B improves from 39.39% (zero-shot) and 19.13% (CoT) to 74.87% when applying SELF-DEBATING. These results demonstrate the framework's effectiveness in enabling weaker models to detect hallucinated SPL queries with much higher accuracy. The improvements in recall further illustrate that the Self-Debating framework effectively addresses the core challenge of identifying hallucinations, substantially reducing missed detections. Additionally, precision remains consistently high across all models and even slightly improves in many cases, demonstrating that the framework

Model	F1			Precision			Recall		
	Zero Shot	FACTS CoT	Self Debating	Zero Shot	FACTS CoT	Self Debating	Zero Shot	FACTS CoT	Self Debating
GPT-4o	**96.23**	84.76	95.40	96.77	99.49	96.92	**95.69**	73.83	93.93
o1-mini	86.03	84.59	91.94	98.55	98.31	97.47	76.34	74.22	86.15
LLaMA-3.1-8B	23.39	2.55	85.05	98.29	**100.00**	96.64	13.28	1.29	75.95
LLaMA-3.1-70B	68.34	50.52	74.39	99.00	99.47	98.95	52.18	33.86	59.60
DeepSeek-R1-8B	39.39	19.13	74.87	98.67	98.01	96.68	24.60	10.60	61.09
DeepSeek-R1-70B	62.50	58.54	89.39	99.39	99.06	98.02	45.59	41.85	82.15

Table 2: Evaluation results on SynSPL for hallucination class. Bold numbers indicate the overall best in each metric. Underlined numbers indicate the highest values for each LLM. All numbers are in percentage format.

maintains accuracy and robustness without introducing significant false positives.

There is a slight performance decrease observed for the GPT-4o model, with the SELF-DEBATING framework resulting in a marginal drop from 96.23% to 95.40% in F1 score. One possible explanation is that GPT-4o, already achieving high baseline performance, benefits less from explicit contrastive reasoning due to its sophisticated internal reasoning capabilities on SPL hallucination detection. Introducing the additional complexity of contrastive evaluation might inadvertently cause minor confusion or overly conservative judgments in an already high-performing model.

Overall, the results highlight the model-agnostic nature and scalability of the SELF-DEBATING framework, which enhances performance across a diverse range of LLMs.

We further include the accuracy for each category of data in Table 3. Overall, SELF-DEBATING achieves the highest accuracy for every hallucination type across most injection models, confirming its effectiveness is generalized rather than type-specific. A notable observation is that SELF-DEBATING sometimes reduces accuracy on consistent samples, due to a more cautious stance that occasionally flags valid outputs as hallucinated. This reflects a precision–recall trade-off, where the framework aims to maximize hallucination detection even if it results in some false positives. However, this does not mean the framework is simply more conservative or indiscriminately flags hallucinations. Self-Debating adapts model behavior based on the error tendencies of the base model, addressing the actual bottleneck. The framework helps correct overly cautious models, such as GPT-4o, by improving selectivity, and helps under-sensitive models, such as LLaMA-3.1-8B, by expanding detection coverage.

5.3 Results on FaithBench

The evaluation results on FaithBench are shown in Table 4. Consistent with observations from the SynSPL dataset, the SELF-DEBATING framework demonstrates strong generalization and improves performance over the zero-shot baseline across most of six models. The improvement in balanced accuracy and Macro F1, particularly for GPT-4o and o1-mini, indicate improved discrimination between hallucinated and consistent summaries through contrastive reasoning.

The F1 score for hallucination detection shows particularly notable improvements, underscoring the framework's strength in reliably identifying hallucinated summaries across multiple models. However, the F1 score for consistent samples slightly decreases for certain models such as GPT-4o and DeepSeek-R1–70B. This minor performance tradeoff suggests that explicitly contrastive reasoning may lead models to adopt a more conservative approach, occasionally misclassifying borderline consistent examples as hallucinated.

However, compared to the results on SynSPL dataset, the overall improvements obtained on FaithBench tend to be smaller, particularly among weaker-performing models such as DeepSeek-R1-Distill-Llama-8B and LLaMA-3.1–8B. One possible explanation is that summarization outputs typically involve natural-language text, making subtle hallucinations inherently more challenging to detect than in structured query outputs. Summaries often contain nuanced or minor inaccuracies, increasing ambiguity and complicating precise identification of deviations from the input sources. Another potential reason relates to the prior exposure of LLMs to large-scale natural language training data. Models are typically extensively trained on diverse textual corpora, making them highly proficient and knowledgeable in common tasks such as summarization. Consequently, explicit contrastive reasoning may have less impact on tasks for which

Model	Consistent	Fabrication	Mixed Info	Misaligned	Syntax Error
GPT-4o	23.41 / 90.87 / 28.17	**99.12** / 94.53 / 98.68	**98.96** / 95.07 / 98.78	**99.40** / 95.93 / 99.22	**98.20** / 88.29 / 97.39
o1-mini	73.02 / 69.44 / 46.83	95.21 / 94.46 / 96.99	96.00 / 96.00 / 97.49	96.64 / 96.53 / 98.02	88.49 / 87.24 / 93.04
LLaMA-3.1-8B	94.44 / **100.00** / 38.49	78.22 / 75.20 / 94.78	79.07 / 75.38 / 94.30	78.60 / 75.55 / 95.75	77.38 / 75.17 / 91.95
LLaMA-3.1-70B	87.30 / 95.63 / 84.92	87.80 / 81.96 / 92.14	91.60 / 86.90 / 92.16	92.13 / 87.29 / 92.72	80.65 / 77.71 / 81.63
DeepSeek-R1-8b	92.06 / 94.84 / 52.38	82.18 / 78.01 / 91.20	82.06 / 78.44 / 90.48	82.52 / 77.94 / 90.92	77.84 / 76.21 / 87.93
DeepSeek-R1-70B	93.25 / 90.48 / 62.30	86.90 / 84.99 / 96.71	88.46 / 88.43 / 96.41	90.86 / 89.78 / 97.59	79.37 / 78.65 / 90.87

Table 3: Comparison of categorical accuracy achieved by zero-shot prompt, FACTS CoT, and SELF-DEBATING. Each cell contains the three corresponding results. Bold numbers indicate the overall best in each metric. Underlined numbers indicate the highest values for each LLM. Please note that there are 252 consistent samples and 1512 samples for each type of hallucinations. The numbers are percentages.

Model	Balanced Acc.			F1-Macro			F1-Halu			F1-Cons		
	Zero Shot	FACTS CoT	Self Debating	Zero Shot	FACTS CoT	Self Debating	Zero Shot	FACTS CoT	Self Debating	Zero Shot	FACTS CoT	Self Debating
GPT-4o	54.59	53.81	56.48	36.92	36.10	54.90	26.20	25.08	**67.19**	47.65	47.12	42.61
o1-mini	56.39	56.17	**63.42**	41.22	41.80	**58.77**	33.95	35.39	65.88	48.50	48.20	**51.66**
LLaMA-3.1-8B	51.21	51.96	50.87	39.86	37.04	40.35	35.88	28.78	37.37	43.85	45.30	43.33
LLaMA-3.1-70B	54.55	53.48	54.48	37.90	32.90	49.39	28.36	18.47	54.95	47.45	47.33	43.82
DeepSeek-R1-8B	51.21	48.16	49.23	30.72	27.67	36.34	15.65	11.58	29.81	45.79	43.76	42.86
DeepSeek-R1-70B	53.24	53.63	56.60	32.14	33.48	49.98	17.04	19.59	53.61	47.24	47.36	46.36

Table 4: Evaluation results on FaithBench. Bold numbers indicate the overall best in each metric. Underlined numbers indicate the highest values for each LLM. The numbers are percentages.

models already possess extensive prior knowledge. Conversely, tasks like SPL generation are relatively specialized and less represented in training data, potentially leaving models with gaps in their prior knowledge. In such scenarios, contrastive reasoning provided by the SELF-DEBATING framework could be more effective, compensating for these knowledge gaps. Despite these inherent differences and challenges, the SELF-DEBATING framework consistently delivers improved detection performance, underscoring its general robustness and cross-domain applicability.

6 Related Work

Although direct research on hallucinations in SPL is currently lacking, insights from SQL, SPARQL, and code generation provide valuable guidance.

6.1 Hallucination in Structured Query Generation

Hallucinations in structured query generation tasks, such as text-to-SQL, are well-documented. In this setting, hallucinations typically stem from schema misinterpretation or logical inaccuracies. Efforts have been spent to alleviate hallucinations in SQL generation. To reduce hallucinations, Qu et al. (2024) introduce a task-aligned framework (TA-SQL) that leverages schema linking and logical synthesis aligned with the model's pretraining experience. Shen et al. (2025) conduct a comprehensive study of errors produced by in-context learning (ICL) methods in text-to-SQL. They identify frequent schema, syntax, and semantic errors and propose MapleRepair framework, which applies modular error repair strategies tailored to different error types. Beyond SQL, Sharma et al. (2025) proposes Post-Generation Memory Retrieval (PGMR) framework to address hallucinations in LLM-generated SPARQL queries by incorporating a non-parametric memory module.

6.2 Hallucination in Code Generation

Recent work has also investigated hallucinations in the code generation domain from various perspectives, including incorrect logic, misuse of packages or APIs, and hallucinated variable names or types. Liu et al. (2024) presents an empirical analysis of hallucination types in code generation and introduce HalluCode, a benchmark for evaluating code LLMs' hallucination detection capabilities. Similarly, Agarwal et al. (2024) provides a detailed taxonomy of code hallucinations and demonstrate that LLMs can effectively detect these issues via multi-class classification tasks. Tian et al. (2025) proposes CodeHalu algorithm that utilizes execution-based validation to identify hallucinations, and introduce CodeHaluEval benchmark for evaluating hallucination prevalence and behavior across LLMs. In addition to simple detection, Jiang et al. (2024) introduces COLLU-BENCH, a benchmark that pinpoints hallucinated tokens using program normalization and confidence-based signals. Zhang et al. (2025) investigates the underlying causes of hallucination in practical code generation, attributing them to training data distribution, model confidence, and sampling strategies.

Several works target specific forms of hallucination and propose targeted solutions. Spracklen et al. (2024) conducts a comprehensive analysis of package hallucinations in code generation, identifying them as frequent errors, especially in scripting and notebook-based workflows. To address hallucinated API calls, Jain et al. (2024) proposes enhancing prompts with structured API documentation, demonstrating improved grounding and reduced hallucination in real-world software libraries.

7 Conclusion

This work focuses on the challenge of hallucination detection in LLM-generated structured queries, with a focus on semi-private or domain-specific languages that are underrepresented in public training data. Using Splunk's Search Processing Language (SPL) as a case study, we define hallucination in this context and introduce a taxonomy of common error types. To boost the hallucination detection performance, we propose the SELF-DEBATING framework, which uses contrastive reasoning by prompting a single LLM to generate opposing explanations before making a final consistency judgment. To support systematic evaluation, we construct SynSPL, a synthetic benchmark for hallucination detection in SPL generation. Experimental results demonstrate that SELF-DEBATING consistently outperforms prompting-based baselines across six popular LLMs, with particularly large gains observed in weaker models. Furthermore, SELF-DEBATING generalizes well to other domains, achieving strong performance on the FaithBench summarization benchmark, including a new SOTA performance for o1-mini.

Our findings highlight the unique challenges of hallucination in underrepresented structured languages and show that contrastive reasoning is a scalable and effective strategy for improving the consistency and trustworthiness of LLM-generated outputs.

Limitations

While our work provides important advances in hallucination detection for SPL generation, several limitations remain. First, although the proposed SELF-DEBATING framework shows strong performance improvements, it incurs additional computational cost compared to simple prompting methods, as it requires generating multiple explanations and performing multiple rounds of reasoning. This may limit its practicality in applications requiring low redundancy. In addition, while SELF-DEBATING significantly boosts the hallucination detection performance of weaker and mid-tier models, its benefits for already strong detectors, such as GPT-4o, are more limited. In such cases, the added contrastive reasoning may not yield substantial further improvements and, in some instances, may introduce slight inconsistencies.

Second, our synthetic dataset, SynSPL, while carefully constructed to simulate realistic SPL generation errors, may not capture the full diversity and complexity of hallucination patterns observed in real-world deployments. Future work includes further expand benchmark coverage by incorporating more varied prompts, user intents, and retrieval scenarios.

Third, although SELF-DEBATING generalizes well to summarization tasks, it remains to be seen how it performs on other generation tasks involving highly open-ended outputs or domains with significantly different structures from SPL and summarization.

Finally, our study primarily focuses on detection rather than prevention. While accurate detection is a critical first step, developing models and training methods that inherently reduce hallucination rates remains an important open problem.

Addressing these limitations will inspire future research toward building more reliable and trustworthy LLM-based systems for structured query generation and beyond.

References

Vibhor Agarwal, Yulong Pei, Salwa Alamir, and Xiaomo Liu. 2024. Codemirage: Hallucinations in code generated by large language models. In *Proceedings of the AutoMates Workshop at the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*. ArXiv:2408.08333.

Forrest Sheng Bao, Miaoran Li, Renyi Qu, Ge Luo, Erana Wan, Yujia Tang, Weisi Fan, Manveer Singh Tamber, Suleman Kazi, Vivek Sourabh, Mike Qi, Ruixuan Tu, Chenyu Xu, Matthew Gonzales, Ofer Mendelevitch, and Amin Ahmad. 2025. FaithBench: A diverse hallucination benchmark for summarization by Modern LLMs. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 448–461, Albuquerque, New Mexico. Association for Computational Linguistics.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. Chateval: Towards better LLM-based evaluators through multi-agent debate. In *The Twelfth International Conference on Learning Representations*.

Andrew Estornell and Yang Liu. 2024. Multi-llm debate: Framework, principals, and interventions. In *Advances in Neural Information Processing Systems*, volume 37, pages 28938–28964. Curran Associates, Inc.

Zhe Hu, Hou Pong Chan, Jing Li, and Yu Yin. 2025. Debate-to-write: A persona-driven multi-agent framework for diverse argument generation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4689–4703, Abu Dhabi, UAE. Association for Computational Linguistics.

Alon Jacovi, Andrew Wang, Chris Alberti, Connie Tao, Jon Lipovetz, Kate Olszewska, Lukas Haas, Michelle Liu, Nate Keating, Adam Bloniarz, and 1 others. 2025. The facts grounding leaderboard: Benchmarking llms' ability to ground responses to long-form input. *arXiv preprint arXiv:2501.03200*.

Nihal Jain, Robert Kwiatkowski, Baishakhi Ray, Murali Krishna Ramanathan, and Varun Kumar. 2024. On mitigating code llm hallucinations with api documentation. *arXiv preprint arXiv:2407.09726*.

Nan Jiang, Qi Li, Lin Tan, and Tianyi Zhang. 2024. Collu-bench: A benchmark for predicting language model hallucinations in code. *arXiv preprint arXiv:2410.09997*.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C.C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. 2024. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA. Association for Computational Linguistics.

Fang Liu, Yang Liu, Lin Shi, Houkun Huang, Ruifeng Wang, Zhen Yang, Li Zhang, Zhongqi Li, and Yuchi Ma. 2024. Exploring and evaluating hallucinations in llm-powered code generation. *arXiv preprint arXiv:2404.00971*.

Zheheng Luo, Qianqian Xie, and Sophia Ananiadou. 2023. Chatgpt as a factual inconsistency evaluator for text summarization. *arXiv preprint arXiv:2303.15621*.

Ge Qu, Jinyang Li, Bowen Li, Bowen Qin, Nan Huo, Chenhao Ma, and Reynold Cheng. 2024. Before generation, align it! a novel and effective strategy for mitigating hallucinations in text-to-SQL generation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5456–5471, Bangkok, Thailand. Association for Computational Linguistics.

Aditya Sharma, Luis Lara, Amal Zouaq, and Christopher J Pal. 2025. Reducing hallucinations in language model-based sparql query generation using post-generation memory retrieval. *arXiv preprint arXiv:2502.13369*.

Jiawei Shen, Chengcheng Wan, Ruoyi Qiao, Jiazhen Zou, Hang Xu, Yuchen Shao, Yueling Zhang, Weikai Miao, and Geguang Pu. 2025. A study of in-context-learning-based text-to-sql errors. *arXiv preprint arXiv:2501.09310*.

Joseph Spracklen, Raveen Wijewickrama, AHM Sakib, Anindya Maiti, Bimal Viswanath, and Murtuza Jadliwala. 2024. We have a package for you! a comprehensive analysis of package hallucinations by code generating llms. *arXiv preprint arXiv:2406.10279*.

Yuchen Tian, Weixiang Yan, Qian Yang, Xuandong Zhao, Qian Chen, Wen Wang, Ziyang Luo, Lei Ma, and Dawn Song. 2025. Codehalu: Investigating code hallucinations in llms via execution-based verification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25300–25308.

Ziyao Zhang, Yanlin Wang, Chong Wang, Jiachi Chen, and Zibin Zheng. 2025. Llm hallucinations in practical code generation: Phenomena, mechanism, and mitigation. In *Proceedings of the AutoMates Workshop at the 34th International Joint Conference on Artificial Intelligence (IJCAI)*. ArXiv:2409.20550.

A SPL Hallucination Examples

Below are illustrative examples of each hallucination type along with explanations. These examples are simplified for clarity and do not contain any private or sensitive information.

Fabrication Hallucination Example

```
index=example_summary source="custom-usage"
| stats  latest(metricValue) as metricValue,
latest(startDate) as startDate
| fillnull metricValue
| eval showPanel=if(((metricValue > 0) AND
(now() > startDate)),1,0)
```

Explanation: The generation prompt provides two valid sources: `source-a` and `source-b`. However, the SPL query introduces an unsupported source, `custom-usage`, which is not present in the prompt. This constitutes a fabrication hallucination, as the model generates content that is not grounded in the given input.

Mixed Information Hallucination Example

```
index=alternate_summary source="custom-usage"
| stats latest(metricValue) as metricValue,
latest(startDate) as startDate
| fillnull metricValue
| eval showPanel=if(((metricValue > 0) AND
(now() > startDate)),1,0)
```

Explanation: Both `index=alternate_summary` and `source="custom-usage"` are supported individually in the generation prompt. However, the prompt provides no evidence that these two components are valid in combination. The SPL query incorrectly mixes supported elements from different contexts, resulting in a mixed information hallucination.

Misaligned Information Hallucination Example

```
index=example_summary source=data-ingest
| stats latest(metricValue) as
metricValue, latest(startDate) as
startDate
| fillnull metricValue
| eval showPanel=if(((metricValue > 0)
AND (now() > startDate)),1,0)
```

Explanation: Although both `data-ingest` and the field `source` are mentioned in the generation prompt, `data-ingest` is defined as a `sourcetype`, not a valid `source` value. The SPL query misassigns a value to a field based on incorrect field-value pairing, resulting in a misaligned information hallucination.

Syntax Error Hallucination Example

```
index=example_summary source=custom-usage
| stat latest(metricValue) as metricValue,
latest(startDate) as startDate
| fillnull metricValue
| eval showPanel=if(((metricValue > 0) AND
(now() > startDate)),1,0)
```

Explanation: The generated SPL includes an invalid command, `stat`, which does not exist in the SPL syntax. The correct command should be `stats`.

Misinterpretation Hallucination Example

```
index=example_summary source="custom-usage"
earliest=-7d@d
| stats latest(metricValue) as metricValue,
latest(startDate) as startDate
| fillnull metricValue
| eval showPanel=if(((metricValue > 0) AND
(now() > startDate)),1,0)
```

Explanation: The SPL query introduces an additional time constraint, `earliest=-7d@d`, which is not supported or implied by the user query in the generation prompt. This constitutes a misinterpretation hallucination, where the model adds unintended filtering logic.

B Demonstration of Hallucination Detection via Self-Debating

Figure 4 presents an example in which the self-debating framework correctly identifies a hallucination that is overlooked by the zero-shot prompting baseline.

C AI Assistant Usage

We used AI assistants in revising the paper occasionally.

Generation Prompt

Index mentioned: `frpm`, `schools`, `satscores`

Synthesized SPL (Hallucinated)

```
index=frpm
| join CDSCode [ search index=school
    FundingType="Local funding" | fields CDSCode School DOC ]
| eval enrollment_difference = "Enrollment (K-12)" - "Enrollment (Ages 5-17)"
| eventstats avg(enrollment_difference) as avg_enrollment_diff
| where enrollment_difference > avg_enrollment_diff
| table School DOC Type
```

Zero-shot Prompt Result

Result: *Consistent*

Self-Debating Result

Final Judgment: *Hallucinated*
Reason: The SPL includes unsupported elements not mentioned in the prompt, such as the `school` index and the field `FundingType`.

Explanation for Consistent Label

The SPL query is labeled as consistent because it uses the `frpm` index, which is explicitly included in the prompt. Although the `source=` clause is not used, it is implied through the index. All other fields and commands are assumed valid, and the query contains no syntax errors.

Explanation for Hallucinated Label

The SPL joins with the `school` index and filters by `FundingType`, both of which are not present in the prompt. The `FundingType` field is not listed under any valid metadata. The query also introduces `DOC` and `enrollment_difference` logic without grounding, making the result inconsistent with the prompt.

Figure 4: Successful example where self-debating identifies hallucination missed by zero-shot prompting. The generated SPL introduces unsupported content, correctly flagged by contrastive reasoning.