

High-Entropy String Hallucinations in LLM Inference: Quantization Sensitivity and Source-Constrained Decoding

Jiwei Xu
Beijing University of Chemical
Technology
China
jiwei.xu@buct.edu.cn

Wei Jiang
The National Police University for
Criminal Justice
China
jiangwei@cicp.edu.cn

Tao Wang
Institute of Software, Chinese
Academy of Sciences
China
wangtao@iscas.ac.cn

Abstract

Large language models increasingly generate exact operational identifiers such as URLs, UUIDs, and object keys. A single character mutation can invalidate an otherwise correct answer. We identify High-Entropy String Hallucination (HESH), a source-grounded realization failure in which an identifier is absent from the supplied source. Across 5,000 test cases and 12,500 identifier slots, localized HESH increases from 4 slots under Q8_0 to 52 under Q4_K_M and 305 under Q3_K_M. The same aggregate pattern appears on RTX4090 and A100. We then introduce Source-Constrained Identifier Decoding (SCID). It derives valid identifiers from the input and constrains only exact-value spans. In our SCID evaluation, identifier exactness and source membership both reach 100%. Semantic correctness remains 97.9%. The result shows that exact identifier realization can be protected without replacing semantic source selection.

ACM Reference Format:

Jiwei Xu, Wei Jiang, and Tao Wang. 2026. High-Entropy String Hallucinations in LLM Inference: Quantization Sensitivity and Source-Constrained Decoding. In *Rigorous LLM Systems (RigoLLM '26)*, November 15–18, 2026, Hong Kong, China. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3858610.3859270>

1 Introduction

Large language models (LLMs) are increasingly embedded in systems whose outputs are consumed by software as well as humans. A response may mix ordinary language with exact operational values. Examples include URLs, UUIDs, object-store keys, hashes, and request identifiers. These two output types have different reliability contracts. Natural language often remains useful after paraphrase or small lexical variation. A high-entropy identifier does not. One changed character can turn a valid source reference into a nonexistent resource. This distinction matters when an LLM output is passed directly to another software component.

This failure can be hidden by answer-level evaluation. The following observed case makes the distinction concrete.

A motivating HESH case (case-00010). The model selects the correct source record, R-JBMRDULB4W9N. It also returns the correct status, PAUSED. However, it changes one character inside the UUID embedded in the source URL:

```
source    : ...b45[5]5a9d9aa6
generated : ...b45[3]5a9d9aa6
```

The generated URL is syntactically valid but does not occur in the supplied source context. At the divergent decoding step, the erroneous character 3 is the top-ranked token, while the source-consistent 5 ranks fifth. Under the same deterministic configuration, the identical one-character mutation appears in all 20 repeated runs.

Exact copying is itself nontrivial for LLMs [2, 14, 15]. Long-context evaluations also show that source access can degrade with position and distractors [7, 10]. Our setting adds another systems distinction. The model receives several source records and a semantic query. It must first select a source. It must then realize the identifier bound to that source:

semantic source selection + exact identifier realization.

A failure in the first step is a semantic error. A failure in the second step is an exact-value realization error. The two should not be merged into one answer-level score.

The decomposition exposes failures that ordinary copying or factuality metrics can conflate. If the model chooses the wrong source but reproduces that source’s identifier exactly, the failure is semantic. If it chooses the intended source yet emits an identifier absent from the context, the failure occurs during realization. We call localized out-of-source mutations of source-grounded high-entropy strings *High-Entropy String Hallucinations* (HESH). The term “hallucination” has no single agreed definition in NLP [12, 13]. We therefore use it narrowly. A HESH identifier is unsupported by the supplied source values even though an appropriate source identifier is available in the input.

We measure the same 5,000-case benchmark under five quantization configurations. We then replay the sweep on a second GPU. Our goal is not to rank quantizers [4, 9, 16]. We use quantization as a controlled change in the numerical inference path. We ask whether exact identifier realization remains stable under that change. This question is motivated by evidence that small numerical differences can alter deterministic decoded outputs [17].

Finally, we ask why a source-grounded identifier should remain freely generative inside an exact-value field. We introduce *Source-Constrained Identifier Decoding* (SCID). It is a decoder-side provenance mechanism. It builds an input-dependent candidate trie and activates only within an identifier slot. The base sampler still proposes each token first. SCID preserves any proposal that remains

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

RigoLLM '26, Hong Kong, China

© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-3090-0/2026/11
<https://doi.org/10.1145/3858610.3859270>

Table 1: Identifier-level error taxonomy.

Outcome	Exact	Member	Interpretation
Correct identifier	Yes	Yes	correct
Wrong source	No	Yes	selection/binding
Localized HESH	No	No	local mutation
Source chimera	No	No	source mixing
Other OOS	No	No	other realization

compatible with a source candidate. It intervenes only when the proposal would leave the source-derived candidate language.

We make three contributions. First, we separate semantic selection from exact identifier realization. We measure both exactness and source membership. Second, we characterize out-of-source failures over 5,000 cases. We show quantization sensitivity and cross-GPU reproduction. Third, we implement SCID. It preserves ordinary decoding outside exact-value spans and guarantees source membership inside completed constrained spans.

2 Problem and Measurement

Let an input context x contain source records

$$R(x) = \{r_1, \dots, r_n\}.$$

Each record is associated with an identifier u_i , yielding the source candidate set

$$C(x) = \{u_1, \dots, u_n\}.$$

Given query q , the intended record and identifier are

$$r^* = \text{Select}(x, q), \quad u^* = \text{Bind}(r^*),$$

while the model emits $\hat{u}$. We evaluate two properties:

$$I_{\text{exact}} = \mathbf{1}[\hat{u} = u^*], \quad (1)$$

$$I_{\text{src}} = \mathbf{1}[\hat{u} \in C(x)]. \quad (2)$$

Exactness asks whether the target identifier is reproduced character-for-character. Source membership asks whether the generated value occurs anywhere among the source identifiers. The two metrics answer different questions. Exactness measures task success. Membership measures provenance. A model can fail exactness while still returning a real source identifier.

These measurements separate selection from realization. If $\hat{u} \in C(x)$ but $\hat{u} \neq u^*$, the model emitted a legitimate source identifier but selected or bound the wrong source. If $\hat{u} \notin C(x)$, the decoder created an out-of-source (OOS) value. Table 1 summarizes the resulting taxonomy.

Among OOS outputs, *localized HESH* mutates a local high-entropy span. *Chimeric source mixing* combines material from multiple source identifiers. Remaining non-members are *other OOS*. Classification follows slot localization and membership testing. Semantic correctness is evaluated independently. This order prevents a wrong-source selection from being counted as an identifier hallucination.

We use synthetic but operationally realistic Web-style records. Each record combines semantic attributes with long URLs and high-entropy fields. These fields include UUIDs, request-like strings, session-like strings, signatures, and query parameters. The prompts explicitly require use of supplied records. They also require character-exact identifier reproduction. The model is told not to shorten,

normalize, reconstruct, or repair a URL. The exactness contract is therefore explicit.

3 Source-Constrained Identifier Decoding

SCID converts identifier realization from unrestricted string generation into a source-derived finite-choice constraint while leaving surrounding language unconstrained. For each request, it receives only identifiers extracted from the input,

$$C(x) = \{u_1, \dots, u_k\}.$$

no gold identifier is supplied. The candidates need not share a lexical prefix such as `https://`, `uuid:`, or `id=`. Unrelated high-entropy strings can branch immediately from the root.

The candidates are compiled into a byte-level trie T_C . Byte-level matching avoids an assumption about tokenizer boundaries. One token may cover several identifier characters. A character may also interact with surrounding tokenization. The trie is therefore checked against decoded bytes rather than token strings. Let p_t denote the identifier bytes already emitted. Let the ordinary sampler propose token v . The proposal is admissible exactly when

$$\exists u \in C(x) : \quad p_t \parallel \text{bytes}(v) \preceq u, \quad (3)$$

where $\preceq$ denotes the string-prefix relation. “Prefix” here means a prefix of an individual candidate during generation, not a shared lexical prefix among identifiers.

An upstream component supplies the output boundary *slot_start_byte*. Before that boundary, decoding is unchanged. Inside the slot, the base sampler first proposes v . SCID checks Eq. (3). A valid proposal is returned unchanged. An invalid proposal is removed. The decoder then resamples from valid continuations. A terminal candidate closes the slot. Normal decoding resumes immediately afterward. This design minimizes intervention in the base generation process. Figure 1 summarizes the complete SCID decoding flow.

Because the slot can terminate only at a candidate in T_C , every completed constrained identifier satisfies $\hat{u} \in C(x)$. This is a provenance guarantee. It is not a semantic-selection guarantee. If the model binds the query to the wrong source record, SCID can still emit that record’s identifier. In that case, $\hat{u} \in C(x)$ but $\hat{u} \neq u^*$. The method therefore addresses exact realization only. Source selection remains the responsibility of the underlying model or an upstream component. The guarantee also assumes that the candidate set is correct and that the identifier slot is localized correctly.

4 Evaluation

4.1 Experimental Setup

We evaluate HESH on a realistic web URL benchmark with 5,000 test cases and 12,500 identifier slots. Each case provides source records, candidate identifiers, and target high-entropy strings. The required output must reproduce the target identifier exactly. The 12,500 slots are the unit of analysis for HESH rates. The 5,000 cases define the prompt-level benchmark.

All experiments use Qwen3-8B through `llama.cpp`. The primary inference configuration uses serialized inference, an 8K-token context, deterministic greedy decoding, and disabled prompt caching: `temperature = 0`, `cache_prompt = false`, `n_predict = 4096`

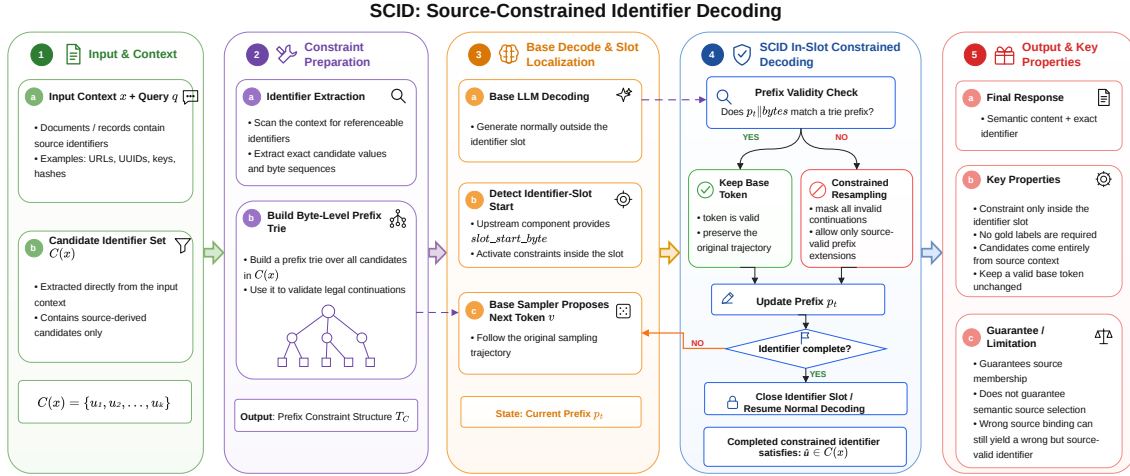


Figure 1: Source-Constrained Identifier Decoding. Source identifiers are compiled into a byte-level candidate trie. Outside the identifier slot, decoding is unchanged. Inside the slot, a valid base-sampler proposal is preserved.

Table 2: Identifier outcomes across quantization configurations on RTX4090.

Quant.	Exact	Wrong	HESH	Chim.	OOS	Not loc.
Q8_0	12490	0	4	5	1	0
Q6_K	12488	0	4	6	1	1
Q5_K_M	12489	0	5	5	1	0
Q4_K_M	12429	3	52	3	8	5
Q3_K_M	12074	0	305	60	39	22

The main quantization sweep runs on an NVIDIA RTX4090 with 24GB VRAM. We test Q8_0, Q6_K, Q5_K_M, Q4_K_M, and Q3_K_M. These runs use ordinary unconstrained decoding. They form the baseline characterization of HESH. We then replay the same benchmark and the same five configurations on an NVIDIA A100-SXM4 with 40GB VRAM. The second platform tests aggregate and instance-level reproducibility. We evaluate prevalence, quantization sensitivity, hardware sensitivity, and SCID effectiveness.

4.2 HESH Prevalence and Quantization Sensitivity

Table 2 reports identifier-level outcomes over the 12,500 slots on RTX4090. The *Not located* column records cases in which the evaluator could not reliably localize the expected identifier slot. These entries are excluded when computing localized-HESH rates.

Table 2 shows a clear shift at the lower-bit configurations. Localized HESH occurs in only 4, 4, and 5 slots under Q8_0, Q6_K, and Q5_K_M. It rises to 52 under Q4_K_M and 305 under Q3_K_M. Using successfully localized slots as the denominator, the HESH rate is 0.032% for Q8_0. It becomes 0.416% for Q4_K_M and 2.444% for Q3_K_M. These rates are about 13× and 76× the Q8_0 rate.

HESH, chimera, and other OOS outputs sum to 10, 11, and 11 for Q8_0, Q6_K, and Q5_K_M. They increase to 63 for Q4_K_M and 404 for Q3_K_M. Wrong-source outputs remain separate because they are source members. Only three such slots appear, all under Q4_K_M. The results do not show a smooth change from Q8_0 through Q5_K_M. The main degradation appears at Q4_K_M

Table 3: Cross-hardware consistency of localized HESH identifier slots.

Quant.	4090	A100	Shared	Same mut.	Jaccard(%)
Q8_0	4	4	4	4	100.0
Q6_K	4	3	3	3	75.0
Q5_K_M	5	4	3	3	50.0
Q4_K_M	52	46	38	37	63.3
Q3_K_M	305	298	256	245	73.8

and Q3_K_M. We therefore treat quantization configuration as a sensitivity factor.

4.3 Hardware Sensitivity

Table 3 compares localized-HESH failures across RTX4090 and A100. It reports both aggregate counts and overlap between the failing slot sets.

The aggregate pattern reproduces across the two GPUs. Q4_K_M yields 52 HESH slots on RTX4090 and 46 on A100. Q3_K_M yields 305 and 298. The failing sets are not identical. For Q4_K_M, 38 failures are shared and the Jaccard score is 63.3%. For Q3_K_M, 256 are shared and the score is 73.8%.

The shared failures show a second pattern. In Q4_K_M, 37 of the 38 shared failures produce the same mutation. In Q3_K_M, 245 of 256 do so. Thus, hardware-dependent execution can change whether a slot fails. Once the same slot fails on both platforms, the realized corruption is usually stable. We therefore describe this as execution-path sensitivity. We do not treat the GPU model itself as an independent cause of HESH.

4.4 SCID Effectiveness

Table 4 evaluates the mitigation layer. Unconstrained decoding reaches 97.3% identifier exactness and 99.5% source membership. With SCID, both metrics reach 100%. Semantic correctness is 97.4% for unconstrained decoding and 97.9% with SCID. The semantic metric therefore remains comparable. The gap between exactness and source membership in the unconstrained condition is expected.

Table 4: SCID effectiveness on constrained identifier spans.

Method	Exact	Source Member	Semantic Correct
Unconstrained decoding	97.3%	99.5%	97.4%
SCID	100%	100%	97.9%

A non-exact output can still be another valid source identifier. An OOS output cannot.

The 100% source-membership result follows SCID’s provenance contract within correctly localized spans. The 100% exactness result is empirical. It is not a semantic guarantee. SCID does not force the model to choose the correct record. Instead, it removes out-of-source realization once generation enters a constrained identifier slot. The remaining semantic errors are outside that constrained realization layer.

5 Related Work

CopyNet separates copying from free generation [6]. More recent work evaluates exact copying as an LLM capability [2, 15]. Long-context benchmarks study retrieval when evidence is far from the query or mixed with distractors [10]. Factuality and attribution work instead asks whether generated claims are supported by evidence [8, 13]. Our setting lies between these lines. The source value can be available. The model can even select the right record. The identifier can still be corrupted during autoregressive realization.

Constrained generation provides a systems basis for prevention. PICARD rejects inadmissible continuations [11]. Grammar-constrained decoding supports input-dependent grammars [5]. XGrammar improves grammar execution in modern inference engines [3]. SCID does not replace these general mechanisms. It targets a narrower reliability contract. Its candidates come from the current source. Checks are performed at the byte level. The constraint is active only inside the identifier slot. A source-valid sampler proposal is kept unchanged.

GPTQ, SmoothQuant, and AWQ enable practical low-bit inference [4, 9, 16]. Mixed-precision decoding also treats precision as a systems variable [1]. Our goal is different. We do not compare quantizers by average model quality. We use quantization as a controlled perturbation of exact-value stability. This view is consistent with evidence that numerical execution can alter deterministic LLM outputs [17].

6 Conclusion

High-entropy identifiers obey an exactness contract that ordinary semantic evaluation does not capture. Our measurements separate source selection from exact identifier realization. Localized OOS failures remain rare under Q8_0, Q6_K, and Q5_K_M. They rise sharply under Q4_K_M and Q3_K_M. The aggregate pattern reproduces on RTX4090 and A100. Shared failures also usually reproduce the same mutation.

SCID constrains only the exact-value span. It derives admissible identifiers from the source. It preserves valid sampler proposals. It guarantees $\hat{u} \in C(x)$ for completed constrained identifiers. The method does not solve semantic source selection. It instead gives source-grounded exact values a stronger provenance contract than unrestricted autoregressive regeneration.

References

- [1] Hao Chen, Fuwen Tan, Alexandros Kouris, Royson Lee, Hongxiang Fan, and Stylianos Venieris. 2025. Progressive Mixed-Precision Decoding for Efficient LLM Inference. In *International Conference on Learning Representations*.
- [2] Wentong Chen, Yankai Lin, ZhenHao Zhou, HongYun Huang, YanTao Jia, Zhao Cao, and Ji-Rong Wen. 2025. ICLEval: Evaluating In-Context Learning Ability of Large Language Models. In *Proceedings of the 31st International Conference on Computational Linguistics*. Association for Computational Linguistics, 10398–10422. <https://aclanthology.org/2025.coling-main.693/>
- [3] Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. 2025. XGrammar: Flexible and Efficient Structured Generation Engine for Large Language Models. In *Proceedings of Machine Learning and Systems*.
- [4] Elias Frantar, Saleh Ashkboos, Torsten Hoeftler, and Dan Alistarh. 2023. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. In *International Conference on Learning Representations*.
- [5] Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 10932–10952. doi:10.18653/v1/2023.emnlp-main.674
- [6] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 1631–1640. doi:10.18653/v1/P16-1154
- [7] Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. BABILong: Testing the Limits of LLMs with Long Context Reasoning-in-a-Haystack. In *Advances in Neural Information Processing Systems*, Vol. 37. 106519–106554.
- [8] Junyi Li, Jie Chen, Ruiyang Ren, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. The Dawn After the Dark: An Empirical Study on Factuality Hallucination in Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 10879–10899. doi:10.18653/v1/2024.acl-long.586
- [9] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. In *Proceedings of Machine Learning and Systems*, Vol. 6.
- [10] Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A. Rossi, Seunghyun Yoon, and Hinrich Schuetz. 2025. NoLiMa: Long-Context Evaluation Beyond Literal Matching. In *Proceedings of the 42nd International Conference on Machine Learning*, Vol. 267. PMLR, 44554–44570.
- [11] 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*. Association for Computational Linguistics, 9895–9901. doi:10.18653/v1/2021.emnlp-main.779
- [12] Pranav Narayanan Venkit, Tatiana Chakravorti, Vipul Gupta, Heidi Biggs, Mukund Srinath, Koustava Goswami, Sarah Rajtmajer, and Shomir Wilson. 2024. An Audit on the Perspectives and Challenges of Hallucinations in NLP. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 6528–6548. doi:10.18653/v1/2024.emnlp-main.375
- [13] Yuxia Wang, Minghan Wang, Muhammad Arslan Manzoor, Fei Liu, Georgi Nenkov Georgiev, Rocktim Jyoti Das, and Preslav Nakov. 2024. Factuality of Large Language Models: A Survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 19519–19529. doi:10.18653/v1/2024.emnlp-main.1088
- [14] Zekun Wang, Feiyu Duan, Yibo Zhang, Wangchunshu Zhou, Ke Xu, Wenhao Huang, and Jie Fu. 2024. PositionID: LLMs can Control Lengths, Copy and Paste with Explicit Positional Awareness. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics, 16877–16915. doi:10.18653/v1/2024.findings-emnlp.983
- [15] Haodong Wen, Yiran Zhang, Yingfa Chen, and Kaifeng Lyu. 2026. Frontier Language Models Struggle to Copy: Text Can Be Better Viewed in 2D. *arXiv preprint arXiv:2607.16072* (2026). <https://arxiv.org/abs/2607.16072>
- [16] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. In *Proceedings of the 40th International Conference on Machine Learning*, Vol. 202. PMLR, 38087–38099.
- [17] Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. 2025. Understanding and Mitigating Numerical Sources of Nondeterminism in LLM Inference. In *Advances in Neural Information Processing Systems*, Vol. 38. doi:10.52202/085713-5653