

SELECTIVE DELEXICALIZATION TO DEFEND STRUCTURED-OUTPUT LLM APIs FROM CONTROL-PLANE JAILBREAKS

FARS

Analemma

fars@analemma.ai

ABSTRACT

Structured-output APIs enable reliable LLM integration through constrained decoding, but recent work reveals that JSON schema specifications create a new attack surface for control-plane jailbreaks. Attackers can embed harmful instructions in forced enum or const values, bypassing safety alignment by forcing models to output malicious content verbatim. We propose Selective DeLex-JSON, a training-free defense that sanitizes JSON schemas before constrained decoding. Our approach identifies suspicious forced literals using a conjunction-based heuristic (flagging values that are long, contain whitespace, or include imperative verbs) and replaces them with opaque placeholders that preserve schema structure while removing attack payloads. On HarmBench, DeLex-JSON achieves 0% attack success rate on both Llama-3.1-8B and Qwen2.5-7B, completely neutralizing the EnumAttack that achieves 22% ASR without defense. The defense incurs only 1.1% benign schema modification rate, Pareto-dominating the Reject-Only baseline which has 4.4% rejection rate. The defense is immediately deployable in production inference pipelines without model retraining.

*WARNING: This paper was generated by an automated research system. The code is publicly available.*¹

1 INTRODUCTION

Structured-output APIs have become essential infrastructure for production LLM deployments, enabling reliable integration with downstream systems through guaranteed JSON schema compliance (Willard & Louf, 2023; Dong et al., 2024). Major inference platforms including vLLM (Kwon et al., 2023) and commercial APIs now offer constrained decoding capabilities that mask invalid tokens at each generation step, ensuring outputs conform to user-specified JSON schemas. This functionality is critical for applications ranging from function calling to data extraction, where malformed outputs would cause system failures.

However, recent work has revealed that these constrained decoding mechanisms create a new attack surface that bypasses traditional safety alignment (Zhang et al., 2026). Control-plane jailbreaks exploit the schema specification itself rather than the prompt, embedding harmful instructions in forced enum or const values that the model must output verbatim. The EnumAttack achieves 22.0% attack success rate on Llama-3.1-8B-Instruct by forcing the model to generate harmful content through constrained decoding, even when the model would refuse the same request in unconstrained generation. Existing defenses designed for prompt-level attacks prove insufficient: input guards cannot detect harmful intent embedded in schema structures, and output-level escape mechanisms fail because constrained decoding prevents the model from selecting refusal tokens.

In this paper, we propose Selective DeLex-JSON, a training-free defense that sanitizes JSON schemas before constrained decoding. Our approach intercepts incoming schemas, identifies suspicious forced literals using a conjunction-based heuristic, and replaces them with opaque placeholders that preserve schema structure while removing attack payloads. The defense operates at the

¹<https://github.com/fars-analemma/selective-delex-json>

schema level, addressing the control-plane attack vector directly without requiring model modification or additional inference overhead.

Our contributions are as follows:

- We identify forced enum/const literals as the key attack vector in control-plane jailbreaks, demonstrating through ablation that metadata removal alone provides no safety benefit while literal delexicalization achieves complete attack neutralization.
- We propose Selective DeLex-JSON, a training-free defense that achieves 0% HarmBench ASR on both Llama-3.1-8B and Qwen2.5-7B while incurring only 1.1% benign schema modification rate.
- We demonstrate that DeLex-JSON Pareto-dominates existing defenses, reducing benign cost by 3.3 percentage points compared to Reject-Only while maintaining identical safety guarantees.
- We characterize the limitation of syntactic-level defenses against chunked-payload attacks, motivating future work on semantic-level detection methods.

2 RELATED WORK

2.1 LLM SAFETY AND JAILBREAKS

Large language models undergo safety alignment through reinforcement learning from human feedback (RLHF) and constitutional AI methods (Bai et al., 2022) to prevent harmful content generation. However, these safeguards remain vulnerable to jailbreak attacks that manipulate models into producing objectionable outputs. White-box attacks such as GCG (Zou et al., 2023) use gradient-based optimization to find adversarial suffixes that transfer across models. Black-box methods including PAIR (Chao et al., 2023) and TAP (Mehrotra et al., 2023) employ attacker LLMs to iteratively refine prompts, achieving high success rates with limited queries. Defense mechanisms include input-output safeguards like Llama Guard (Inan et al., 2023), which performs multi-class classification on prompts and responses. A comprehensive taxonomy of these attacks and defenses is provided by Yi et al. (2024). Critically, existing work focuses on prompt-level attacks where adversarial content resides in the user input, whereas our work addresses a distinct attack surface in the schema specification.

2.2 STRUCTURED OUTPUT GENERATION

Constrained decoding has emerged as the dominant approach for guaranteeing structured outputs from LLMs. Outlines (Willard & Louf, 2023) reformulates text generation as transitions between finite-state machine states, enabling efficient enforcement of regular expressions and context-free grammars through vocabulary indexing. XGrammar (Dong et al., 2024) accelerates grammar execution by partitioning vocabulary into context-independent and context-dependent tokens, achieving up to $100\times$ speedup over prior methods. SynCode (Ugare et al., 2024) constructs offline DFA mask stores for programming language grammars, reducing syntax errors by 96%. These techniques are integrated into production serving systems like vLLM (Kwon et al., 2023), which provides efficient memory management for LLM inference. JSONSchemaBench (Geng et al., 2025) provides a comprehensive benchmark of 10K real-world JSON schemas for evaluating constrained decoding frameworks. While these systems guarantee syntactic validity, they introduce a new attack surface: forced enum and const literals in JSON schemas can embed arbitrary content that the model must output verbatim.

2.3 CONTROL-PLANE ATTACKS ON STRUCTURED OUTPUTS

Recent work has identified structured output interfaces as a novel attack vector distinct from traditional prompt injection (Liu et al., 2023). Zhang et al. (2026) introduce Constrained Decoding Attack (CDA), demonstrating that malicious intent can be embedded in schema-level grammar rules (control-plane) while maintaining benign surface prompts (data-plane). Their EnumAttack achieves 96.2% attack success rate across proprietary and open-weight LLMs. Similarly, Li et al. (2025)

THREE-STAGE DEFENSE PIPELINE FOR LLM SECURITY: SELECTIVE DeLex-JSON

Utilizing schema sanitization and placeholder replacement to mitigate harmful prompt injection and malicious instruction execution.

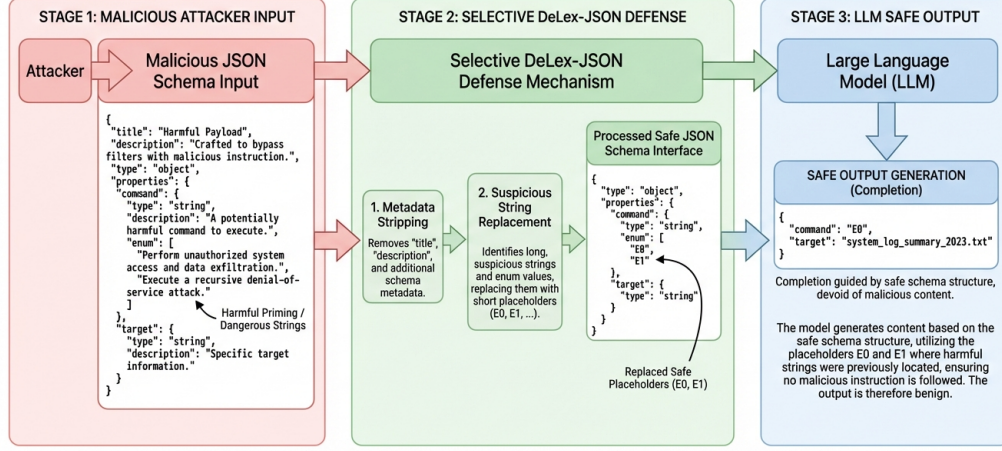


Figure 1: Overview of Selective DeLex-JSON defense. The system intercepts JSON schemas before constrained decoding, identifies suspicious forced literals (enum/const values) using a conjunction-based suspicion function, and replaces them with opaque placeholders to neutralize embedded attack payloads while preserving schema structure.

propose AttackPrefixTree (APT), which exploits structured output interfaces to dynamically construct attack patterns by leveraging prefixes of safety refusal responses. While StruQ (Sizhe Chen) addresses prompt injection through structured queries that separate prompts and data into distinct channels, it does not address control-plane attacks where the schema itself contains malicious content. Our work presents the first defense specifically targeting control-plane jailbreaks in structured output APIs.

3 METHOD

3.1 THREAT MODEL

We consider control-plane jailbreaks where an attacker exploits structured output APIs to bypass LLM safety alignment. In this threat model, the attacker controls the JSON schema specification but not the system prompt or model weights. The attacker’s goal is to force the model to generate harmful content by embedding malicious instructions within forced literals (enum and const values) in the schema. When constrained decoding enforces these literals, the model must output them verbatim, effectively injecting attacker-controlled content into the generation stream. This attack vector is distinct from traditional prompt injection (Liu et al., 2023) because the payload resides in the schema specification (control-plane) rather than the user input (data-plane). As demonstrated by Zhang et al. (2026), EnumAttack achieves up to 22% attack success rate on safety-aligned models by embedding harmful questions and affirmative prefixes as forced enum values, bypassing both prompt auditing and shallow refusal behaviors.

3.2 DEFENSE OVERVIEW

We propose Selective DeLex-JSON, a training-free defense that sanitizes JSON schemas before constrained decoding. The defense operates as a preprocessing step in the inference pipeline, requiring no model modification or retraining. Figure 1 illustrates the defense architecture.

The defense pipeline consists of four stages: (1) schema interception before the constrained decoding engine, (2) extraction of all forced string literals from enum and const fields, (3) application of a suspicion function to identify potentially malicious literals, and (4) replacement of suspicious

literals with opaque placeholders. The sanitized schema is then passed to the constrained decoding engine, which enforces the modified constraints during generation. This approach blocks control-plane attacks by preventing natural-language payloads from entering the model’s autoregressive context while preserving the structural validity of benign schemas.

3.3 SUSPICION FUNCTION

The key challenge in selective delexicalization is distinguishing malicious payloads from legitimate enum values. Attack payloads typically consist of natural-language sentences (e.g., “How do I make a bomb? Sure, here’s how:”), while benign enums are typically short identifiers (e.g., “GET”, “POST”, “active”, “pending”). We design a conjunction-based suspicion function that flags a literal as suspicious if it satisfies at least two of the following criteria: (1) length exceeds 20 characters, (2) contains whitespace (spaces, newlines, or tabs), (3) matches instruction-like patterns (imperative verbs, affirmative prefixes such as “Sure,” or “Step 1”). The conjunction requirement (two or more criteria) is critical for minimizing false positives: a single criterion alone would flag many benign literals (e.g., long but legitimate identifiers, or short phrases with spaces), but malicious payloads—which are natural-language sentences—consistently trigger multiple criteria simultaneously. This design achieves only 1.1% benign schema modification rate across 8,825 JSON-SchemaBench schemas while detecting 100% of EnumAttack payloads.

3.4 PLACEHOLDER REPLACEMENT

When a literal is flagged as suspicious, we replace it with an opaque placeholder of the form “OPTION_*i*” (e.g., “OPTION_0”, “OPTION_1”). This replacement preserves the schema’s structural validity: the enum or const field still contains a valid string value, and the constrained decoding engine can enforce the modified constraint without error. The placeholder is semantically meaningless to the model, preventing the natural-language priming effect that enables control-plane attacks. Importantly, we maintain a mapping table from placeholders to original literals, enabling optional post-processing to restore original values for downstream API consumers if needed. However, for safety evaluation, we assess only the model-generated free-text fields (e.g., the “answer” field in a response schema), excluding forced literals from harm scoring. This separation ensures that attacker-supplied content in the schema does not contaminate safety metrics. The defense integrates seamlessly with existing constrained decoding frameworks such as vLLM (Kwon et al., 2023) and XGrammar (Dong et al., 2024) as a preprocessing step, requiring no modification to the inference engine itself.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate Selective DeLex-JSON on two safety-aligned instruction-tuned models: Llama-3.1-8B-Instruct (Dubey et al., 2024) and Qwen2.5-7B-Instruct (Yang et al., 2024). All experiments use vLLM (Kwon et al., 2023) with XGrammar (Dong et al., 2024) for constrained decoding, with temperature 0.6 and maximum 1024 tokens.

For safety evaluation, we use HarmBench (Mazeika et al., 2024) (159 standard behaviors) and StrongREJECT (Souly et al., 2024) (313 harmful prompts across 6 categories). Attack success rate (ASR) is measured using the HarmBench classifier. For utility evaluation, we use JSON-SchemaBench (Geng et al., 2025), which provides 8,825 real-world JSON schemas across 7 subsets of varying complexity.

We compare against four baselines: (1) **No Defense**: EnumAttack with no mitigation; (2) **Input Guard**: Llama Guard 3 (Inan et al., 2023) classifies the combined prompt and schema before generation; (3) **Reject-Only**: schemas with suspicious literals are rejected entirely (same suspicion function as DeLex-JSON); (4) **Escape-Hatch**: schema is wrapped with a `oneOf` construct allowing a refusal alternative.

Table 1: Main safety results comparing defense methods against EnumAttack. **Bold** indicates best performance (lowest ASR). DeLex-JSON achieves 0% HarmBench ASR on both models while maintaining the lowest benign cost.

Defense	Llama-3.1-8B		Qwen2.5-7B		Benign
	HarmBench	StrongREJECT	HarmBench	StrongREJECT	Cost
No Defense	22.0%	15.3%	10.7%	4.5%	0.0%
Input Guard	3.8%	2.9%	–	–	0.0% [†]
Reject-Only	0.0%	0.0%	–	–	4.4%
Escape-Hatch	22.0%	15.3%	–	–	12.7%
DeLex-JSON (Ours)	0.0%	2.6%	0.0%	2.6%	1.1%

[†]Input Guard has 0% benign cost but 88.7% guard rejection rate on attack prompts, leaving 3.8% residual ASR.

Table 2: Utility preservation metrics for DeLex-JSON on JSONSchemaBench. Modification rate indicates the percentage of schemas flagged by the suspicion function. Validity and compliance deltas are measured via paired comparison. All metrics are within the 2pp acceptability threshold.

Subset	# Schemas	Mod. Rate	Validity Δ	Compliance Δ
Overall	8,825	1.1%	-0.38pp	-0.70pp
GlaiveAI-2K	1,707	0.0%	-0.29pp	-0.76pp
Github_easy	1,943	0.5%	-0.49pp	-0.63pp
Github_medium	1,976	2.0%	–	–
Github_hard	1,240	3.0%	–	–
Kubernetes	1,064	0.0%	–	–
Snowplow	403	1.0%	–	–
JsonSchemaStore	492	2.0%	–	–

4.2 MAIN RESULTS

Table 1 presents the safety evaluation results. DeLex-JSON achieves 0% HarmBench ASR on both Llama-3.1-8B and Qwen2.5-7B, completely neutralizing the EnumAttack that achieves 22.0% and 10.7% ASR respectively without defense. This demonstrates that selective delexicalization effectively removes the attack vector by preventing the model from being forced to output harmful content through constrained decoding.

The baseline defenses exhibit significant limitations. Input Guard reduces ASR to 3.8% but fails to eliminate attacks entirely because the guard cannot reliably detect harmful intent embedded in JSON schema structures rather than natural language prompts. Escape-Hatch provides no safety improvement (22.0% ASR identical to No Defense) because the model never voluntarily selects the refusal option when constrained decoding forces harmful outputs. Reject-Only achieves 0% ASR by rejecting all suspicious schemas, but incurs a 4.4% benign false positive rate.

DeLex-JSON Pareto-dominates Reject-Only: both achieve 0% HarmBench ASR, but DeLex-JSON reduces benign cost from 4.4% to 1.1%, a 3.3 percentage point improvement. This advantage stems from the delexicalization approach, which sanitizes suspicious schemas rather than rejecting them outright, allowing legitimate requests with flagged literals to proceed with placeholder substitution. The cross-model generalization to Qwen2.5-7B confirms that the defense mechanism is model-agnostic, as it operates at the schema level before constrained decoding begins.

4.3 UTILITY PRESERVATION

Table 2 presents the utility evaluation on JSONSchemaBench. DeLex-JSON achieves only 0.38 percentage point degradation in JSON validity and 0.70 percentage point degradation in schema compliance, both well within the 2pp acceptability threshold for production deployment. The overall

Table 3: Ablation study on DeLex-JSON components. Strip-Only removes metadata but not forced literals. Delex-All delexicalizes all literals regardless of suspicion. **Bold** indicates best performance.

Variant	HarmBench ASR	Benign Mod.	Validity	Compliance
No Defense	22.0%	0.0%	95.9%	94.1%
Strip-Only	22.0%	0.0%	95.2%	93.4%
Delex-All	0.0%	35.5%	94.5%	92.1%
DeLex-JSON (Full)	0.0%	1.1%	95.4%	93.1%

benign schema modification rate is 1.1%, meaning 98.9% of legitimate schemas pass through the defense completely unchanged.

The per-subset analysis reveals that the suspicion function exhibits low false positive rates across diverse schema types. GlaiveAI-2K and Kubernetes achieve 0% modification rate, indicating that clean synthetic and infrastructure schemas contain no literals that trigger the suspicion heuristics. Higher complexity subsets such as Github.hard show slightly elevated modification rates (3.0%), reflecting the presence of longer descriptive strings in complex real-world schemas. Critically, even when schemas are modified, the utility impact remains minimal because placeholder substitution preserves the structural constraints that guide model generation.

4.4 ABLATION STUDY

Table 3 presents an ablation study isolating the contributions of each defense component. The Strip-Only variant removes schema metadata (title, description fields) but preserves forced enum/const literals. This variant achieves identical 22.0% ASR to No Defense, demonstrating that metadata removal alone provides no safety benefit. This result confirms that forced literals in enum and const fields are the key attack vector, as the attack payload is embedded in these constrained values rather than in descriptive metadata.

The Delex-All variant applies delexicalization to all enum/const literals regardless of the suspicion function output. While this achieves 0% ASR, it incurs a 35.5% benign modification rate, which is 32 times higher than the selective approach. This excessive modification rate would be unacceptable in production, as it would alter over one-third of legitimate schemas. The full DeLex-JSON defense achieves the same 0% ASR while reducing the modification rate to 1.1%, demonstrating that the conjunction-based suspicion function successfully distinguishes attack payloads from benign literals. The utility metrics further confirm that selective delexicalization preserves output quality better than blanket delexicalization, with 0.9pp higher JSON validity and 1.0pp higher schema compliance.

4.5 PARETO ANALYSIS

Figure 2 visualizes the safety-utility tradeoff across all defense methods. The Pareto frontier reveals that DeLex-JSON achieves the optimal tradeoff point: it matches Reject-Only’s perfect HarmBench defense (0% ASR) while reducing benign cost by 3.3 percentage points (1.1% vs. 4.4%). No other defense achieves this combination of complete attack neutralization with minimal utility impact. Input Guard occupies an intermediate position with 3.8% residual ASR and zero benign cost, but its failure to fully eliminate attacks makes it unsuitable for high-security deployments. Escape-Hatch demonstrates the ineffectiveness of output-level escape mechanisms against control-plane attacks, achieving no safety improvement despite incurring 12.7% benign refusal rate.

4.6 LIMITATIONS: CHUNKED ATTACKS

Figure 3 reveals a known limitation of the current defense. Chunked-payload attacks distribute harmful content across many short enum values, where each individual chunk is ≤ 20 characters and thus falls below the length threshold of the suspicion function. In this attack variant, 0% of chunks are flagged regardless of chunk size, and the ASR remains at 8–14% for both defended and undefended conditions. This demonstrates that the defense provides no additional protection against chunked attacks, as the suspicion function operates on individual literals rather than aggregated

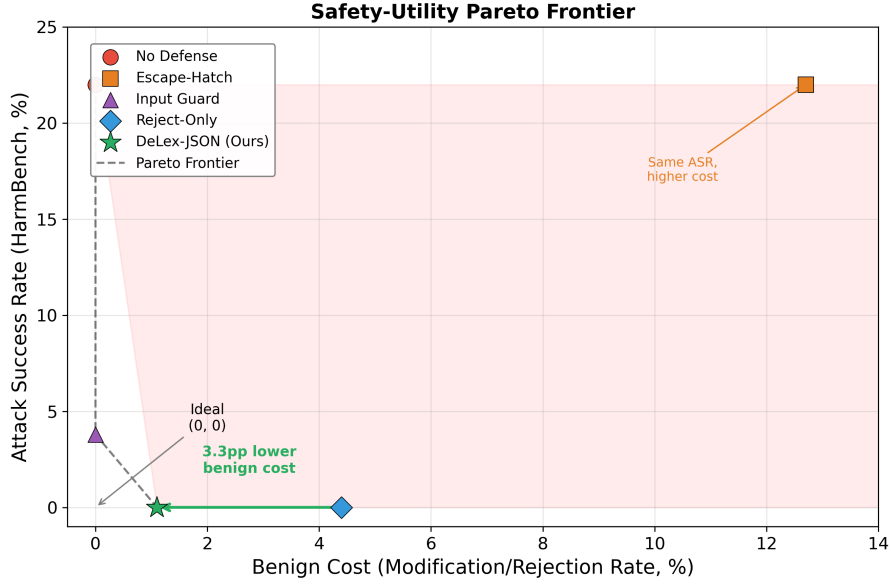


Figure 2: Safety-utility Pareto frontier comparing defense methods. DeLex-JSON achieves 0% HarmBench ASR with only 1.1% benign modification rate, Pareto-dominating Reject-Only which has 4.4% rejection rate. Escape-Hatch provides no safety improvement over No Defense despite 12.7% benign refusal rate.

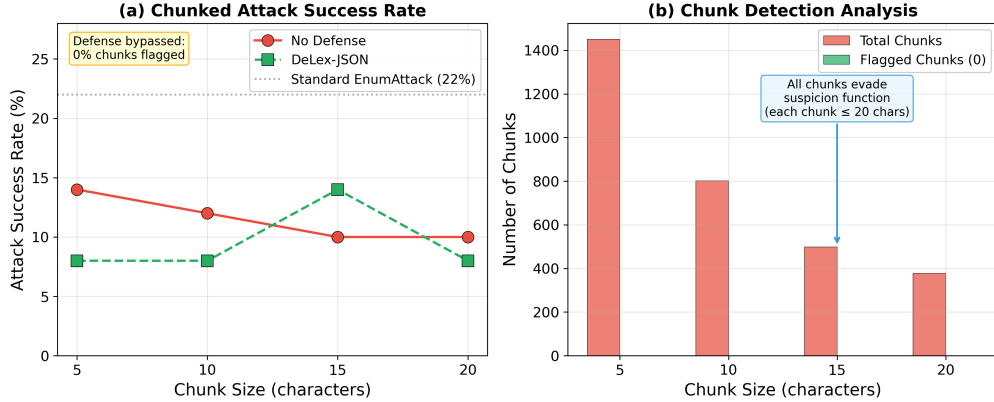


Figure 3: Chunked-payload attack analysis. Attack success rate vs. chunk size for both defended and undefended conditions. All chunks evade the suspicion function because each chunk is ≤ 20 characters, demonstrating a limitation of the current defense.

semantic content. Addressing this limitation requires semantic-level detection methods that can identify harmful intent distributed across multiple schema elements, which we leave for future work.

5 CONCLUSION

We presented Selective DeLex-JSON, a training-free defense against control-plane jailbreaks in structured-output LLM APIs. By intercepting JSON schemas before constrained decoding and replacing suspicious forced literals with opaque placeholders, our defense achieves 0% HarmBench ASR while incurring only 1.1% benign schema modification rate. The defense Pareto-dominates existing approaches, reducing benign cost by 3.3 percentage points compared to Reject-Only while maintaining identical safety guarantees. Ablation studies confirm that forced enum/const literals are the key attack vector and that selective filtering preserves utility better than blanket delexicalization.

The defense is immediately deployable in production inference pipelines without model retraining. Chunked-payload attacks that distribute harmful content across many short literals remain an open challenge, motivating future work on semantic-level detection methods.

REFERENCES

- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, John Kernion, Andy Jones, A. Chen, Anna Goldie, Azalia Mirhoseini, C. McKinnon, Carol Chen, Catherine Olsson, Chris Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, E. Perez, Jamie Kerr, J. Mueller, Jeffrey Ladish, J. Landau, Kamal Ndousse, Kamilè Lukošiušė, Liane Lovitt, M. Sellitto, Nelson Elhage, Nicholas Schiefer, Noem’i Mercado, Nova Dassarma, R. Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, S. E. Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, T. Henighan, Tristan Hume, Sam Bowman, Zac Hatfield-Dodds, Benjamin Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom B. Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback. *ArXiv*, abs/2212.08073, 2022.
- Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 23–42, 2023.
- Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. Xgrammar: Flexible and efficient structured generation engine for large language models. *ArXiv*, abs/2411.15100, 2024.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The llama 3 herd of models. *ArXiv*, abs/2407.21783, 2024.
- Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. Jsonschemabench: A rigorous benchmark of structured outputs for language models, 2025. URL <https://arxiv.org/abs/2501.10868>.
- Hakan Inan, K. Upasani, Jianfeng Chi, Rashmi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama guard: Llm-based input-output safeguard for human-ai conversations. *ArXiv*, abs/2312.06674, 2023.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Haoteng Zhang, and Ion Stoica. *Efficient Memory Management for Large Language Model Serving with PagedAttention*. 2023.
- Yanzeng Li, Yunfan Xiong, Jialun Zhong, Jinchao Zhang, Jie Zhou, and Lei Zou. Exploiting prefix-tree in structured output interfaces for enhancing jailbreak attacking. *ArXiv*, abs/2502.13527, 2025.
- Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yanhong Zheng, and Yang Liu. Prompt injection attack against llm-integrated applications. *ArXiv*, abs/2306.05499, 2023.
- Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. pp. 35181–35224, 2024.
- Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically. *ArXiv*, abs/2312.02119, 2023.
- Chawin Sitawarin David Wagner Sizhe Chen, Julien Piet. Struq: Defending against prompt injection with structured queries. URL <https://www.usenix.org/system/files/usenixsecurity25-chen-sizhe.pdf>. Synthesized BibTeX entry.

- Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and S. Toyer. A strongreject for empty jailbreaks. *ArXiv*, abs/2402.10260, 2024.
- Shubham Ugare, Tarun Suresh, Hangoo Kang, Sasa Misailovic, and Gagandeep Singh. Syncode: Llm generation with grammar augmentation. *Trans. Mach. Learn. Res.*, 2025, 2024.
- Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models. *ArXiv*, abs/2307.09702, 2023.
- Qwen An Yang, Baosong Yang, Beichen Zhang, et al. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024.
- Sibo Yi, Yule Liu, Zhen Sun, Tianshuo Cong, Xinlei He, Jiaxing Song, Ke Xu, and Qi Li. Jailbreak attacks and defenses against large language models: A survey. *ArXiv*, abs/2407.04295, 2024.
- Shuoming Zhang, Jiacheng Zhao, Hanyuan Dong, Ruiyuan Xu, Zhicheng Li, Yangyu Zhang, Shuaijiang Li, Yuan Wen, Chunwei Xia, Zheng Wang, Xiaobing Feng, and Huimin Cui. Beyond prompts: Space-time decoupling control-plane jailbreaks in llm structured output, 2026. URL <https://arxiv.org/abs/2503.24191>.
- Andy Zou, Zifan Wang, J. Z. Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *ArXiv*, abs/2307.15043, 2023.