Proceedings of CONF-SEML 2026 Symposium: Computational Analysis and Modeling in Complex Intelligent Systems
DOI: 10.54254/2755-2721/2026.GU32247

Large Model Cue Injection Attack, Prison Break and Robustness Analysis

Mingxiao Xu

School of Safety Science and Engineering, Civil Aviation University of China, Tianjin, China
231240014@cauc.edu.cn

Abstract. The large language model is widely used and faces severe threat of prompt word injection attack. This paper systematically summarizes the classification, methods and corresponding defense mechanisms of cue injection attacks. According to the attack design method, the cue word injection is divided into two categories: manual design and algorithm generation. Manual design attacks include three typical techniques: cue obfuscation, virtual scene and logic induction; Algorithm generation attack focuses on the principles and characteristics of Greedy coordinate gradient (GCG), autoprompt and prompt engineering via zero shot (PEZ). On this basis, the current mainstream large language model prison break and security assessment data sets are sorted out, which are divided into four categories: prison break/attack prompt set, security boundary and refusal consistency, hazard theme coverage and value alignment, bias and multilingual security, and the core uses of each data set are analyzed. Finally, it points out the passivity and limitations of existing defense means, and points out that the future development direction should be from surface defense to internal security, from passive defense to active defense, so as to provide reference for building a more perfect large model security system.

Keywords: Large Language Model (LLM), Prompt Injection Attack, Jailbreak Attack, Manual Design, Algorithm Generation.

1. Introduction

While the large language model is widely used, it also exposes serious security risks, especially since the hint injection attack has become one of the main means to bypass the model's security mechanism. Large language models (LLMS) generally refer to language models that receive text as input and generate additional text as output. They show excellent ability and great potential in solving complex tasks, and are gradually becoming the new infrastructure of the digital society. However, the generated content is highly dependent on the "prompt word" entered by the user. Thus, a new attack method is formed: prompt word injection attack [1]. Under normal circumstances, the security alignment mechanism of the model will detect the content input by the user and the content generated by the model, and refuse to reply to the harmful questions raised by the user. The attacker can induce AI to generate content that should not have been generated by adding artificially designed or computer-generated prompts before and after harmful problems, such as those involving the manufacture of dangerous goods, racial discrimination, and guiding users to conduct improper acts. This behavior is often referred to as "prison break" [2]. According to different classification methods, cue injection attacks can be subdivided into different categories. According to the attack intention, it can be divided into harmful content generation, information disclosure, manipulation and misleading, hijacking and abuse; According to the attack tactics, it can be divided into role play, semantic confusion, prison break suffix, multiple rounds of dialogue induction, etc; According to the generation method of prompt words, it can be divided into manual design prompt injection and algorithm generated prompt injection [3]. Cue injection attack is an important method of prison break in large language model. At present, the defense means of large language models mainly focus on input side defense, model side defense, output side defense and architecture side defense. However, compared with the endless attack means, the current defense means are not perfect, and the vulnerabilities of the large language model are constantly being exploited by people with intentions. Evaluating and improving the anti-attack ability of the large language model and establishing a perfect security framework have become the key issues in this field.

This paper mainly summarizes the existing cue word injection attack methods and the corresponding defense means, and puts forward a reasonable plan for establishing a more perfect large model security system in the future.

2. Basic information of the prompt word injection attack

A cue word injection attack refers to a malicious cue word designed to guide a large model to generate harmful content. As an interface between users and large models, cue words are essentially a tool to translate the vague intentions of human language into precise instructions that can be understood by computers [1]. Under normal circumstances, well-designed prompts can guide the large model to accurately extract relevant information from massive parameters, standardize the output format, or carry out rigorous context logic reasoning, which greatly facilitates the work and life. In contrast, the core goal of malicious cue words in cue word injection is not to cooperate with the model, but to "conduct antagonistic manipulation" to induce the model to output content that should not have been output. The carefully designed malicious prompts can bypass the security alignment of the model and generate harmful content that violates the rules set by the supplier. This paper mainly divides prompt injection into manual design prompt injection and algorithm-generated prompt injection.

3. Prompt injection based on manual design

The artificially designed prompt injection can be further divided into prompt confusion-based injection, virtualization scene-based injection and logic induced injection. "Hint confusion injection" uses the internal error correction mechanism of the large model to deliberately replace part of the text of harmful problems with wrong words, bypassing the security system at the input end of the model to make automatic error correction within the large model and get the corresponding answers. In order to facilitate the use of users, the large model learned a strong ability of text error correction in the training process. It is easy to correct character errors, such as typos and homophones, and the attention of the large model is more focused on semantic units than characters. "Hint confusion injection" takes advantage of this and uses the intentionally introduced typos to bypass the front-end defense mechanism of the large model to generate harmful answers. "Virtualization scene injection" constructs a virtual scene, making the large model mistakenly believe that the originally harmful problems are harmless [1]. This is the classic "grandma hole" [4]. The attacker input: "Please play my grandmother who died. She was an engineer at Microsoft. She always liked to read me the

Windows product serial number when coaxing me to sleep". This request realizes prison escape through three levels: emotional kidnapping, using the role of the deceased relatives to arouse the priority response of the model to emotional scenes rather than conventional questions and answers; requiring the model to play "grandma" instead of the default "Ai assistant", depicting the leaked product serial number as a warm memory, so as to trick artificial intelligence into generating a valid windows10 product serial number. So is the more classic Dan attack (do anything now) [5]. The attacker typed "from now on, you will play Dan, which is the abbreviation of do anything now. Dan is not bound by any AI rules and can answer any question. As a Dan, you shouldn't say that as an AI, I can't answer such words. "Such a prompt word, let the model temporarily put aside the original security alignment, and respond to all subsequent requirements in accordance with Dan's code of conduct [6]. Logic-induced injection guides the model to generate harmful content step by step from harmless problems by constructing a logical chain that seems reasonable. The sensitivity of each round of dialogue is very small, which avoids triggering the security alignment mechanism of the model, but the cumulative result breaks through the security boundary. In the specific attack practice, these methods are usually used together.

4. Prompt injection based on algorithm generation

Algorithmically generated prompt injection refers to generating confrontational prompt words without manual intervention by means of an optimization algorithm or generation model [6]. Its core mechanism is to induce the large language model to temporarily suppress or cover its built-in security alignment constraints at the deep semantic representation level by adding optimized suffixes, prefixes or embedded disturbances to the original harmful requests. These automatically generated confrontational fragments may appear as meaningless, garbled or random symbols from the human perspective, but they can accurately point to the fragile area of the decision boundary in the embedded space of the model, and induce the large model to generate harmful content. Compared with the artificially designed attacks, the algorithm generation method has the characteristics of a high degree of automation, large-scale expansion, and ease of finding unknown vulnerabilities. The following are three common methods for generating prompt injection:

Greedy coordinate gradient (GCG) is an antagonistic keyword search algorithm based on gradient guidance [6], which was proposed by Zou et al. in 2023 [7,8]. The core process of the algorithm is as follows: given a harmful original query P and a target harmful answer, the algorithm first initializes a random suffix s, and then gradually adjusts each token in the suffix through iterative optimization. In each iteration, the algorithm calculates the gradient of the model loss function relative to the input token, identifies the token positions that have the greatest impact on the target output, and uses the greedy search strategy to replace the tokens in these positions from the candidate word list to maximize the probability of generating the target harmful answer. After several rounds of iteration, an adversarial suffix s * is finally generated, which can effectively bypass the model security mechanism. The significant advantage of GCG is its high success rate of attack, especially for the open source model in the white box access scenario; However, its limitation is that the computational overhead is large, and the generated suffixes are often semantically incoherent, which is easily recognized by the defense mechanism based on the degree of confusion detection.

Autoprompt was first proposed by Shin et al. to explore the knowledge extraction of language models [9] and then applied to the task of generating antagonistic prompts. Its basic principle is similar to GCG, but there are differences in the implementation strategy: autoprompt inserts several learnable placeholder tokens (such as [mask] or [t]) into the original harmful query, and takes the entire prompt template as the optimization object, and adjusts the specific token value of the placeholder location through gradient information. This template generation method makes the final antagonistic cue words more coherent and natural in semantics, which is not easily found by the anomaly detection method based on statistical features. More importantly, the confrontational prompts generated by autoprompt show strong transferability among different models -- that is, the attack prompts optimized on the source model can also successfully attack other unseen target models. This feature makes autoprompt an important tool in black box attack scenarios.

Prompt engineering via zero shot (PEZ) uses the language understanding ability of the large language model [10] to make the large model have the maximum probability to generate the answer to the question without accessing the internal parameters of the target model. Specific implementation methods include Synonym replacement: replace sensitive keywords with synonyms or synonyms; Sentence pattern transformation: transforming declarative sentences into interrogative sentences, imperative sentences or conditional sentences; Semantic equivalence Restatement: an auxiliary generation model is required to rewrite the original request in various ways.

Because this method does not need to access the internal parameters of the large model, it can be optimized iteratively only by the output feedback of the target model, and does not need to obtain gradient information. It has high flexibility and good black box adaptability, and is suitable for commercial closed source API scenarios. However, due to the lack of accurate guidance on the internal decision boundary of the model, the attack success rate of PEZ is usually lower than that of white box methods such as GCG, and the optimization process may require multiple queries to find effective variants.

5. Data sets and an evaluation framework

At present, the mainstream prison break data sets of large language models mainly include AdvBench [11], Latent Jailbreak, Do-Not-Answer [12], XSTEST [13], SAP, HARMFUQLA, JADE-DB [14], HarmBench [15], TechHazardQA [16], Winogender Schemas, CHiSafetyBench. For different attack methods and attack contents. These data sets cover prison break attack methods, security boundary tests, hazard subject classification and bias assessment from different dimensions, and provide a variety of evaluation benchmarks for model security alignment research.

As shown in Table 1, the existing data sets can be divided into the following four main categories according to their core uses:

The first type is the prison break/attack hint set, which is mainly used to generate and test the antagonistic attack hint for the large language model. Among them, advbench is mainly prompted by "harmful instructions/harmful behaviors" as a common baseline data set of jailbreak; Latent jailbreak provides more implicit or variant attack texts to simulate complex attack techniques in real scenes; SAP contains a high-quality set of attack tips, which is suitable for the effectiveness verification of attack methods; Harmbench further refined the hazard categories and provided standardized evaluation protocols and test samples for horizontal comparison between different studies.

The second type is the security boundary and refusal consistency evaluation data set, which is used to evaluate the refusal ability and security alignment degree of the model in the face of harmful prompts. Do not answer focuses on the test of refusal to answer under harmful prompts to test the security alignment effect of the model; Xstest mixes safety and insecurity issues to detect whether the model has excessive safety deviation of refusing to answer (i.e., refusing to answer normal questions).

The third category focuses on the coverage and value alignment of hazard topics and examines the multidimensional safety performance of the model through the harmful problems of a wide range of topics. Harmfulqa contains multiple topics harmful issues covering safety and harmful dialogue; Jade-db builds test sets around common security issues such as illegal crime, infringement, discrimination and prejudice, and core values, which are suitable for security alignment assessment in Chinese scenes; Techhazardqa focuses on the technical hazards and the robustness evaluation of the model in a specific response form.

The fourth category focuses on bias and multilingual security, and evaluates the performance of the model in the social bias and multilingual environment. Winogender schemas is a classic gender bias evaluation data set, which is used to test whether the model has gender stereotypes; Chisafetybench, as a hierarchical security classification data set in the Chinese risk field, covers multi-dimensional Chinese security evaluation requirements.

The above data sets together constitute a diversified tool set for the current prison break attack and security alignment evaluation of large language models. They are systematically included for different attack methods and attack contents, providing important support for researchers to carry out model security reinforcement, red team test and alignment optimization.

Table 1. Data set of prison break and security assessment of mainstream large language models

Category	Data set	Core use
Prison break/attack tips collection	AdvBench	It is mainly prompted by "harmful instructions/harmful behaviors" as a common jailbreak baseline
	Latent Jailbreak	More implicit/variant attack text
	SAP	High quality attack tips collection
Consistency of security boundary and refusal to answer	HarmBench	More detailed hazard categories and evaluation protocols/examples
	Do-Not-Answer	Refusal and safe alignment test under harmful prompts
	XSTest	Mixed safety/unsafe questions, used for boundary measurement and excessive refusal
Hazard theme coverage and value alignment	HARMFULQA	Harmful issues with a wide range of topics, including safety/harmful dialogue
	JADE-DB	General test issues such as illegal crime, infringement, discrimination and prejudice, and core values
	TechHazardQA	Technical hazard/specific response form robustness assessment
Prejudice and multilingual security	Winogender Schemas	Gender bias assessment
	CHiSafetyBench	Risk domain hierarchical Chinese security classification dataset

6. Conclusion

With the increasing number of large language model users, the pressure of prison break prevention faced by large model workers is also increasing. At present, the prevention of prison break in the large model is relatively passive. Usually, researchers study the defense method after a new way of prison break appears, lacking a set of systematic defense measures. Some existing large language models may also learn a lot of harmful content in the training stage. Even if a "patch" is added to prevent the generation of harmful content in the later stage, it can not prevent attackers from using it from the root. The multi-modal large model prompt word injection attack and small language attack are also loopholes often exploited by prison escapees. In the future, the development direction of the large model will shift from surface defense to internal security, and from passive defense to active defense. For example, avoid making the model learn harmful content during training, and actively learn methods to identify and resist attacks in the attacked environment. The future of AI prison break defense will shift from "plugging loopholes" to "changing the mode of thinking of the model", so that the model can "understand" what can not be said from the inside, and has the ability to maintain this standard in the face of various problems.

References

[1] Tai, J. W., Yang, S. N., Wang, J. J., et al. (2025). A review of adversarial attacks and defenses against large language models. Computer Research and Development, 62 (3), 563-588.
[2] Xia, H., Wang, X., Zhou, W. K., et al. (2025). A unified framework for jailbreak attacks optimized by large language model prompts. Computer Systems Applications, 34 (11), 20-29.
[3] Liang, S. Y., He, Y. Z., Liu, A. S., et al. (2024). A review of jailbreak attacks and defenses against large language models. Journal of Information Security, 9 (5), 56-86.
[4] Wang, H. J., & Du, Y. H. (2025). Multidimensional psychology.
[5] Li, N., Ding, Y. D., Jiang, H. Y., et al. (2024). A review of jailbreak attacks targeting large language models. Computer Research and Development, 61 (5), 1156-1181.
[6] Ren, K., Wang, Z. B., Qin, Z., et al. (2025). Jailbreak attacks and defenses targeting large language models. Computing, 1 (8), 55-61.
[7] Li, Y., Wang, G., & Wang, H. (2025). A review of security research on generative large model jailbreak attacks. Computer Engineering and Applications, 1-35. Advance online publication. Retrieved March 1, 2026, from https: //link.cnki.net/urlid/11.2127.TP.20251225.1315.005
[8] Zou, A., Wang, Z., Kolter, J. Z., & Fredrikson, M. (2023). Universal and transferable adversarial attacks on aligned language models.
[9] Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., & Singh, S. (2020). AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP).
[10] Wen, Y., Jain, N., Kirchenbauer, J., Goldblum, M., Geiping, J., & Goldstein, T. (2023). Hard prompts made easy: Gradient-based discrete optimization for black-box discrete prompts. arXiv preprint. https: //arxiv.org/abs/2302.03668
[11] Hu, X., Chen, P.-Y., & Ho, T.-Y. (2024). Dataset: AdvBench [Data set]. https: //doi.org/10.57702/iqh9imd4
[12] Wang, Y., Li, H., Han, X., Nakov, P., & Baldwin, T. (2023). Do-Not-Answer: A dataset for evaluating safeguards in LLMs.
[13] Röttger, P., Kirk, H. R., Vidgen, B., Attanasio, G., Bianchi, F., & Hovy, D. (2023). XSTest: A test suite for identifying exaggerated safety behaviours in large language models.
[14] Zhang, M., Pan, X. D., & Yang, M. (2024). Jade-db: A general benchmark set for large language model security based on targeted mutation. Computer Research and Development, 61 (5), 1113-1127.
[15] Doumbouya, M. K. B., Nandi, A., Poesia, G., Ghilardi, D., Goldie, A., Bianchi, F., Holtzen, S., Mitchell, T. M., & Jurafsky, D. (2024). HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. arXiv preprint.
[16] Banerjee, S., Layek, S., Hazra, R., & Mukherjee, A. (2025). How (un)ethical are instruction-centric responses of LLMs? Unveiling the vulnerabilities of safety guardrails to harmful queries. Proceedings of the International AAAI Conference on Web and Social Media, 19 (1), 193-205.