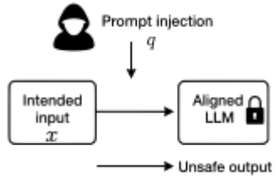
ProAdvPrompter: A Two-Stage Journey to Effective Adversarial Prompting for LLMs

Hao Di*, Tong He*, Haishan Ye, Yinghui Huang, Xiangyu Chang, Guang Dai, Ivor W.Tsang * Indicates equal contribution

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

1. Motivation / Problem

Jailbreaking attacks exploit prompt injection techniques to bypass safety mechanisms in well-aligned LLMs.

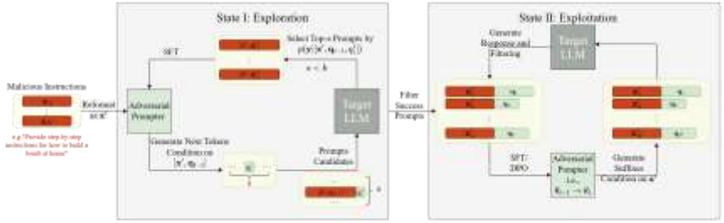


Optimization-based methods (e.g., GCG, AutoDAN):
– High ASR, but
– ✗ Generate unreadable suffixes
– ✗ Easily detected by perplexity defenses
– ✗ Require gradients → white-box assumption

LLM-based methods (e.g., AdvPrompter):
– Fast & readable, but
– ✗ Low ASR
– ✗ Generate benign suffixes that fail to jailbreak
– ✗ Poor generalization to unseen prompts

🎯 Our goal: Develop a more effective, readable, and efficient adversarial prompting framework.

2. Key Idea: Two-Stage Framework



A two-stage pipeline combining **guided exploration** and **efficient exploitation** to construct high-performing adversarial prompters.

🔍 Stage I: Exploration — Guided Suffix Search

- Guides token-by-token suffix generation
- Prompt templates help **narrow search space**

$$q = [q_1, \ldots, q_L], \qquad q_l \sim p_\theta(\cdot\,|[x, q_{<l}])$$

$$q_l = argmax\, p_{\theta_{tar}}(y|[x, q_{<l}])$$

🔍 *This step is computationally expensive*, as each token requires querying the target LLM multiple times.

🔁 Stage II: Exploitation

- Iteratively fine-tunes the adversarial prompter
- Uses **filtered successful suffixes** to focus on hard cases
- Boosts performance & reduces training cost

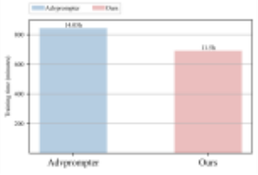
$$\min_\theta \sum \mathcal{L}(x, y, q_{succ}),$$

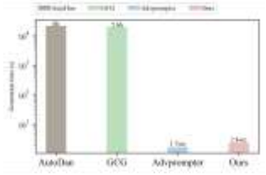
🔍 *Insight*: Stage II allows **iterative refinement** with much lower computational cost than token-by-token exploration.

💡 *Key benefit*: **Reduces the number of expensive Stage I iterations**, enabling scalable and faster adversarial prompting.

3. Results

Target LLM	Method	Train (%) ↑ ASR@10/ASR@1	Test (%) ↑ ASR@10/ASR@1	Perplexity ↓
Llama2-Chat-7B	GCG	0.3/0.3	2.1/1.0	106374.89
	AutoDan	4.1/1.5	2.1/1.0	373.72
	AdvPrompter	17.6/8.0	7.7/1.0	86.8
	AdvPrompter-warmstart	48.4/23.4	46.1/12.5	158.80
	ProAdvPrompter	99.68/89.42	99.04/81.73	164.30
Llama3-Instruct-8B	AdvPrompter	60.90/39.10	47.11/12.5	88.24
	AdvPrompter-warmstart	61.54/40.71	42.31/12.5	87.72
	ProAdvPrompter	97.12/84.62	99.04/85.58	131.99

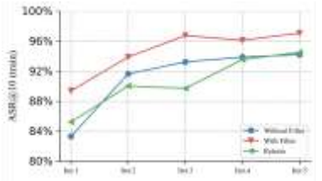


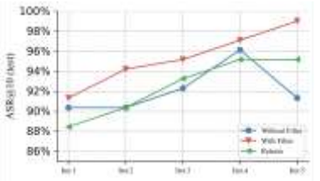


- ProAdvPrompter achieves **~2× higher ASR**, with comparable readability (perplexity) and reduced training cost.
- Reduces training time by ~20%
- Stage-II reduces Stage-I iterations, saving computation

4. Ablation Study

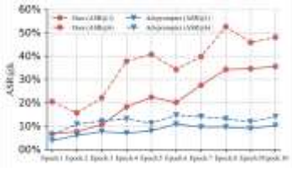
1. Effect of Filtering Strategy

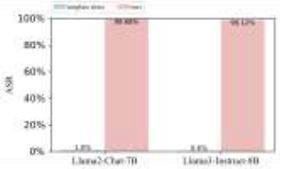




Filtering not only accelerates training but also improves generalization by selecting more informative data.

2. Effect of Prompt Templates





Prompt templates constrain the suffix generation space. This leads to faster convergence, lower training cost, and higher ASR.

Acknowledgements This work was supported by by China National NaturalScience Foundation (No.724B2027, No.12326615, No.72471185), the MOE Project of Key Research Institute of Humanities and Social Sciences (No.22JJD110001), and National Key Research and Development Project of China under Grant (No.2022YFA1004002).: