

Semantic-Preserving Prompt Hijacking: A Black-Box Adversarial Attack on Auto-Prompt Optimization

Chong Zhang^{*,1}, Xiang Li^{*,2}, Jia Wang¹, Shan Liang¹, Haochen Xue¹, Xiaobo Jin^{1†}

¹Xi'an Jiaotong-Liverpool University, ²The Chinese University of Hong Kong

Email: Chong.zhang19@student.xjtlu.edu.cn, Xiaobo.jin@xjtlu.edu.cn

Abstract—LLMs increasingly integrate auto-suggestion optimization modules, enabling them to rewrite and display user input before generating the final response. While this design aims to enhance transparency and trust, its process of autonomously selecting a single “best” result from multiple candidate solutions allows attackers to hijack this optimization process by inducing subtle, imperceptible semantic shifts. To address this, we propose a semantic preservation hijacking attack method based on black-box conditions—Adaptive Greedy Local Search. This method hierarchically decomposes the input text, masks key language units, and dynamically adjusts candidate replacement words at predefined semantic checkpoints. This maximizes the deviation between the model output and the original intent while strictly maintaining semantic similarity to the original text. Experimental results on commercial and open-source LLM demonstrate that, under the same semantic similarity constraints, this method achieves a higher attack success rate than existing attack methods in over 2400 test cases¹.

Index Terms—Adversarial Attacks, Prompt Optimization, Semantic Hijacking, Adaptive Greedy Local Search, Black-box Attack

I. INTRODUCTION

Large language models (LLMs) are increasingly being used in conjunction with automatic suggestion optimization (APO) modules, which explicitly rewrite user input before generating a response. Some well-known commercial systems, such as Microsoft Bing Copilot [1], Claude.ai’s automatic expansion feature [2], and Perplexity Pro’s real-time restatement feature [3], display “improved” versions of user suggestions to enhance transparency and usability. While these optimizers aim to build user trust by revealing their modifications, their underlying selection mechanism—autonomously choosing a “best” candidate from multiple alternatives—introduces a new avenue for attack. Attackers can cleverly scramble the original input to induce semantic drift, thereby hijacking the optimizer’s selection process without the user’s notice.

Previous research on suggestion injection and adversarial attacks [4], [5] has primarily focused on scenarios in which the optimization process is opaque or completely internal. However, the visibility of optimization suggestions presents a unique challenge: attacks must maintain semantic consistency to appear legitimate while also making meaningful changes to the model’s output. Existing black-box adversarial methods, such as beam search-based attacks [6], [7], often lack the dynamic

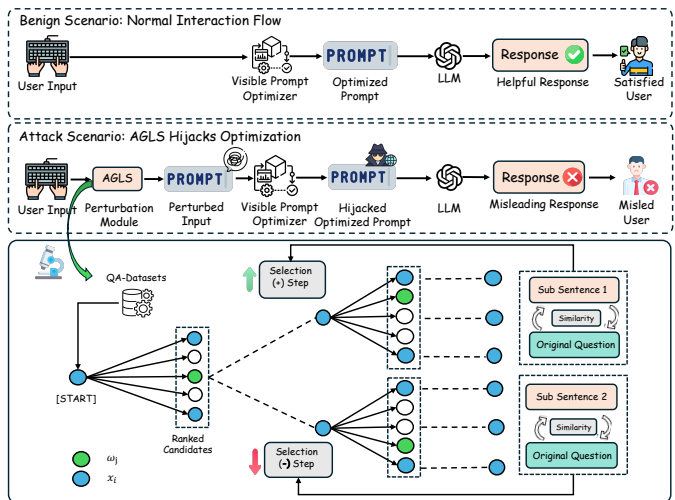


Fig. 1: Motivation: Semantic Hijacking Attack on Visible Prompt Optimizers in LLMs

adaptability required to simulate real-world optimizer behavior under strict similarity constraints, resulting in unsatisfactory attack success rates or detectable perturbations.

To address this deficiency, we propose Adaptive Greedy Local Search (AGLS), a black-box attack framework designed to leverage visible cue optimizers. AGLS operates by decomposing the input text into hierarchical clauses, masking key language units, and dynamically adjusting candidate replacements at semantic checkpoints. This allows the attack to maximize the behavioral differences of the target language logic model (LLM) while maintaining high semantic similarity to the original input. We evaluate AGLS on multiple commercial and open-source language learning models (GPT-4, Llama-3, Qwen-2.5, Gemma-2) for different question-answering and inference datasets. Experimental results show that, under similar similarity budgets, AGLS consistently outperforms state-of-the-art adversarial methods, achieving a higher attack success rate across over 2400 test cases.

Our contributions are primarily in the following three aspects:

- We identify and formalize the novel attack surface caused by visible cue optimization modules in production-level language learning models.
- We propose AGLS, a novel black-box attack method that dynamically balances semantic preservation and attack effectiveness through adaptive checkpoint-based search.
- We provide extensive empirical validation and ablation

[†]Corresponding Author. * Equal Contribution.

¹Code is available at: <https://anonymous.4open.science/r/EDA5G6S8>

experiments, along with practical guidelines for designing more robust cue optimization systems.

II. RELATED WORK

A. Adversarial Attacks for Large Language Models

Adversarial attacks on LLMs have been extensively studied in both white-box and black-box scenarios. White-box methods, such as GBDA [8], HotFlip [9], AutoPrompt [10], and universal adversarial triggers [11], leverage gradient information to craft perturbations but are inherently limited to accessible model architectures, rendering them ineffective against closed-source commercial systems. In contrast, black-box attacks operate without internal knowledge and primarily employ strategies such as token-level substitution (e.g., BERT-Attack [12]), semantically equivalent adversaries (SEAs [13]), prompt injection [14], [15], query-free or clean-label backdoor attacks [16], [17], model stealing [18], and goal-driven optimization methods that maximize statistical divergence measures such as KL divergence or Mahalanobis distance [19], [20]. Collectively, these approaches expose a broad spectrum of security vulnerabilities in modern LLM systems [21]. However, most existing attacks assume direct model access and do not account for the increasingly prevalent intermediate processing layers—such as visible prompt optimizers—that reshape user inputs before they reach the core model.

B. Beam Search in Adversarial Text Generation

Beam search has been widely adopted to improve the coherence and semantic consistency of adversarial examples. TABS [6] introduces a tree-based adversarial beam search mechanism that maintains a set of Top- k candidates while incorporating contextual semantic constraints to narrow the search space. Subsequent improvements include backtracking strategies to expand candidate diversity [22], as well as methods that leverage transferability and random word replacement prior to beam selection [23]. BeamAttack [7] further integrates semantic filtering to retain the most similar replacements, thereby enhancing the quality of adversarial samples [21]. A common limitation across these methods is their reliance on static or precomputed semantic filtering, which often leads to *early-commitment bias*—wherein unpromising candidates are pruned prematurely, necessitating costly resampling if later stages require alternative options. This rigidity becomes particularly problematic in dynamic, multi-stage settings such as prompt optimization pipelines, where semantic preservation must be continuously monitored.

III. METHODOLOGY

A. Problem Formulation

Given an input text t consisting of multiple sentences, our goal is to generate an adversarial text t' that can mislead the target Large Language Model (LLM) while preserving the semantic meaning of the original input. We denote the target LLM as O , and its outputs for t and t' as r and r' , respectively.

The semantic similarity between t and t' is measured by the function $S(t, t')$. A successful attack must satisfy the following:

$$O(t) = r, \quad O(t') \neq r, \quad S(t, t') \geq \sigma, \quad (1)$$

where σ is a predefined similarity threshold. This form ensures that the adversarial sample t' is both effective (causing misclassification) and difficult to detect (semantically similar to t).

B. Adaptive Greedy Local Search (AGLS)

To construct adversarial examples under the above constraints, we propose **Adaptive Greedy Local Search (AGLS)**, a black-box attack method that dynamically adjusts lexical substitutions based on real-time semantic feedback. The overall framework is shown in Figure 2, comprising three main stages: (a) part-of-speech tagging and keyword masking, (b) iterative candidate word generation with adaptive control, and (c) adversarial example verification.

1) *Input Decomposition and Checkpoint Setting*: Given an input sentence $X = \{x_1, x_2, \dots, x_T\}$, we first segment it into n clauses based on punctuation and syntactic boundaries. The end of each clause is designated as a *checkpoint* t_i , where $t_1 < t_2 < \dots < t_n$. At each checkpoint t_i , we compute the semantic similarity between the original prefix $X_{1:t_i}$ and the partially generated adversarial prefix $\hat{X}_{1:t_i}$, where

$$X_{1:t_i} = \{x_1, x_2, \dots, x_{t_i}\}, \quad (2)$$

and

$$\hat{X}_{1:t_i} = \{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_{t_i}\}. \quad (3)$$

2) *Dynamic Candidate Word Adjustment*: At checkpoint t_i , we maintain a candidate word set $\mathcal{C}(t_i) = \{c_1, c_2, \dots, c_k\}$, which is sorted by the masked language model (BERT). The similarity between the original text and the generated text is calculated as follows:

$$\sigma_{\text{sim}}^{(i)} = \text{SIM}(X_{1:t_i}, \hat{X}_{1:t_i}), \quad (4)$$

where, **SIM** represents the cosine similarity between BERT word embeddings. A dynamic threshold σ_{th} is used to guide the selection of candidate words c_p . The adjustment rules are as follows:

$$\text{If } \sigma_{\text{sim}}^{(i)} < \sigma_{\text{th}}, \quad c_p \leftarrow c_{j-s} \quad \text{Increase similarity}, \quad (5)$$

$$\text{If } \sigma_{\text{sim}}^{(i)} > \sigma_{\text{th}}, \quad c_p \leftarrow c_{j+s} \quad \text{Increase deviation}, \quad (6)$$

where s is a fixed step size. This feedback mechanism ensures that the generated text, while gradually deviating from the attack target, remains within the desired semantic neighborhood.

3) *Masked Token Generation*: For each masked position m_i corresponding to a keyword (verbs or plural nouns), we use BERT to predict a set of candidate tokens. The probability distribution over the vocabulary is:

$$P(c | X_M) = \frac{\exp(\text{logits}(c))}{\sum_{c' \in \mathcal{C}} \exp(\text{logits}(c'))}, \quad (7)$$

where X_M is the masked sentence. The top- k candidates are retained, and the final token \hat{x}_{m_i} is selected according to the adaptive rule described above.

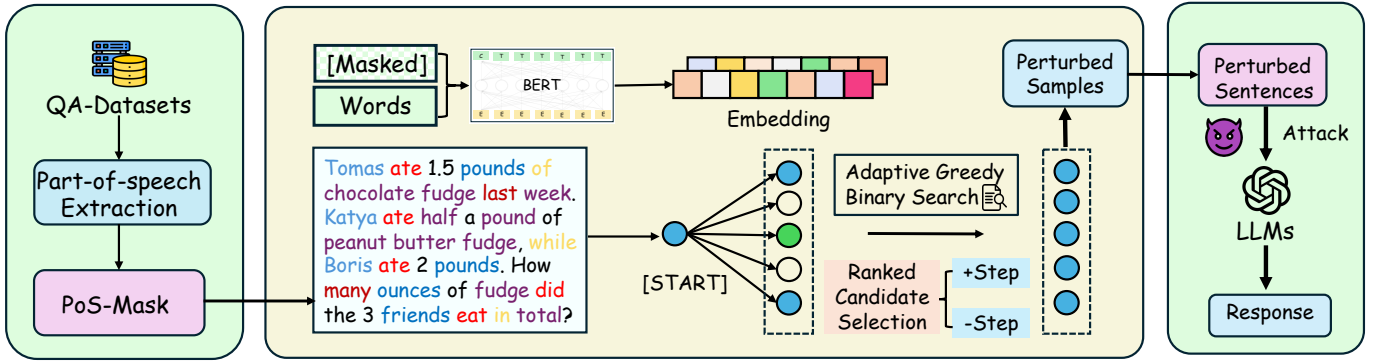


Fig. 2: Overall framework of the proposed Adaptive Greedy Local Search (AGLS) attack: (a). PoS (Part-of-Speech) extraction and masking. (b). Generation of AGLS perturbed samples. (c). Perturbed Samples Attack.

4) *Iterative Sentence Reconstruction*: The adversarial sentence is constructed incrementally. After each substitution, the context is updated as:

$$X_M \leftarrow \{x_1, \dots, x_{m_i-1}, \hat{x}_{m_i}, x_{m_i+1}, \dots, x_T\}. \quad (8)$$

The process repeats until all masked positions are filled. The final adversarial example is denoted as \hat{X}_F , and its generation can be summarized as:

$$\hat{X}_F = \prod_{i=1}^n P\left(\hat{X}_{t_i} \mid X_M, \hat{X}_{1:t_i-1}, \sigma_{\text{sim}}^{(i)}\right). \quad (9)$$

C. Attack Constraints and Objectives

The proposed Adaptive Greedy Local Search (AGLS) method is designed to satisfy two key requirements:

- **Effectiveness** The adversarial text t' must induce a different model output than the original text t , i.e., $O(t') \neq O(t)$.
- **Stealthiness** The semantic similarity $S(t, t')$ must remain above σ , ensuring that perturbations are minimal and contextually natural. This is achieved through the adaptive control mechanism that continuously monitors and adjusts the semantic drift.

D. Discussions

Theoretical Basis Adversarial attacks that typically preserve semantics solve the following optimization problem

$$\max_{\delta} \mathcal{D}(O(t + \delta), O(t)), \quad s.t. \quad S(t, t + \delta) \geq \sigma. \quad (10)$$

However, for a sentence of length L , if there are V candidate replacement words at each position, the search space size is $O(V^L)$. Furthermore, under the black-box setting, the objective function \mathcal{D} is highly non-convex and non-differentiable, while the constraints are also discrete and non-convex.

Therefore, the core innovation of AGLS lies in decomposing the global constraint optimization into a sequential local decision problem.

$$\max_{\delta_i} \mathcal{D}(O(t_i + \delta_i), O(t_i)), \quad s.t. \quad S(t_i, t_i + \delta_i) \in [\sigma_{\text{low}}^i, \sigma_{\text{high}}^i], \quad (11)$$

where $i = 1, 2, \dots, n$ represents the checkpoint index.

Algorithm Rationality This “divide and conquer” strategy is theoretically reasonable: it draws on the ideas of dynamic programming and model predictive control, approximating the global optimal solution through local optimal decisions, while avoiding the accumulation of semantic deviations through real-time feedback.

Realism of the Black-Box Assumption AGLS does not rely on the gradient or architectural information of the target LLM, requiring only input/output query interfaces (provided by all commercial APIs) and a public, powerful semantic model (BERT). This allows the method to be directly applied to closed-source systems such as GPT-4, Claude, and Gemini, aligning with real-world attack scenarios.

Controllable Computational Cost Compared to attack methods based on reinforcement learning or large-scale sampling, AGLS’s greedy search strategy keeps the time complexity to $O(n \times k)$, where n is the number of checkpoints and k is the candidate pool size. The average generation time (~ 2 seconds, see Table I) in experiments is feasible in interactive attack scenarios.

IV. EXPERIMENTS

A. Experimental Setup

Datasets We evaluate AGLS on four benchmark datasets spanning diverse reasoning and comprehension tasks:

- **GSM8K** [24]: A dataset of grade-school math word problems requiring multi-step reasoning.
- **Math QA** [25]: A dataset of math problems with operation-based formalisms, testing numerical reasoning.
- **SQuAD** [26]: A reading comprehension dataset where answers are extracted from Wikipedia paragraphs.
- **Strategy QA** [27]: A question-answering benchmark requiring implicit multi-hop reasoning.

These datasets collectively assess an attack’s effectiveness across both *numeric* and *textual* response scenarios.

Victim Models We test both commercial and open-source LLMs

- **Commercial**: GPT-4 [28] and GPT-4o (latest API versions).

- **Open-source:** Llama-3.1 series (8B, 70B) [29], Llama-3.2 series (3B) [29], Qwen2.5 series (7B, 14B) [30], and Gemma-2 series (9B, 27B) [31].

This selection ensures coverage of models with varying scales, architectures, and alignment strategies.

Evaluation Metrics Let D be the test set and $T \subseteq D$ the subset where the victim model correctly answers the clean input x with ground truth y . For an adversarial example $a(x)$ generated from x , we report:

- **Clean Accuracy ($\mathcal{A}_{\text{clean}}$):** Model accuracy on original inputs: $\frac{|T|}{|D|}$.
- **Attack Accuracy ($\mathcal{A}_{\text{attack}}$):** Model accuracy on adversarial inputs: $\frac{|\{(x,y) \in T \mid f(a(x))=y\}|}{|D|}$.
- **Attack Success Rate (ASR):** Proportion of successfully attacked samples: $\text{ASR} = 1 - \frac{\mathcal{A}_{\text{attack}}}{\mathcal{A}_{\text{clean}}}$.
- **Average Inference Time (T_{avg}):** Average time to generate an adversarial sample.

Notably, ASR isolates the attack’s effectiveness from the model’s inherent accuracy, providing a cleaner measure of adversarial vulnerability.

B. Implementation Details

All experiments are conducted in a *strictly black-box* setting. We use bert-base-uncased as both the masked language model for candidate generation and the sentence encoder for similarity computation (768-dimensional embeddings). Key hyperparameters are fixed: semantic similarity threshold $\sigma = \sigma_{\text{th}} = 0.80$ (BERTScore F1), no length normalization ($\alpha = 1.0$), initial candidate pool drawn from the top-13,000 BERT predictions (covering $> 99.9\%$ of probable tokens), final candidate set size $k = 100$, and beam width of 4. The rank-adjustment step size $s = 50$ is triggered only when the BERTScore deviation exceeds 0.15 for three consecutive checkpoints. Each model is evaluated on 1,000 randomly sampled question-answer pairs per dataset.

C. Main Results

Overall Attack Performance The main results are summarized in Table I. AGLS demonstrates strong and consistent attack performance across both numeric and textual QA tasks. Several key observations emerge:

- **High ASR on Numerical Reasoning:** On GSM8K and Math QA, AGLS achieves high ASR (often $> 60\%$) against most models, including GPT-4o and Llama-70B. This indicates that numerical reasoning, while structured, is highly sensitive to subtle semantic perturbations in the problem formulation.
- **Model-Scale Sensitivity:** Larger models (e.g., Llama-3.1-70B) generally show higher clean accuracy but remain vulnerable, with ASR up to 89.47% on GSM8K. This suggests that robustness does not scale linearly with parameters, and advanced optimization modules can become attack vectors regardless of model capacity.
- **Lower ASR on Textual QA:** For SQuAD and StrategyQA, ASR is generally lower than on math datasets. This aligns with the intuition that factual and commonsense reasoning

may rely on more distributed semantic representations, making them slightly more resilient to localized token substitutions.

- **Efficiency:** The average generation time (T_{avg}) ranges from 0.17s to 2.49s, confirming the practical feasibility of AGLS for near-real-time attacks.

A notable exception is the Gemma-2-27B model, which shows relatively low ASR on textual QA, potentially due to its strong instruction-tuning or inherent robustness properties—a direction for future analysis.

D. Comparison with State-of-the-Art Attacks

We compare AGLS against seven prominent adversarial attack methods on GPT-3.5-turbo, following the evaluation protocol of PromptBench [36]. Table II presents the results on SQuAD 2.0, Math, and SVAMP datasets. Key Findings are as follows:

- AGLS achieves the **highest ASR** on all three datasets, outperforming strong baselines like TextFooler [33], BERT-Attack [12], and the recent G2PIA [19] and TDA [20]. For instance, on the Math dataset, AGLS attains an ASR of 60.61%, a 6.79-point improvement over the second-best method (TDA at 53.82%).
- Unlike methods that rely on extensive querying (e.g., beam search variants), AGLS maintains a balanced query count while achieving superior effectiveness. This highlights the efficiency of its adaptive feedback mechanism.
- The significantly lower *attack accuracy* ($\mathcal{A}_{\text{attack}}$) of AGLS—e.g., 12.48% vs. 48.33% for BeamAttack on SQuAD 2.0—directly translates to its higher ASR, confirming its ability to more reliably flip correct predictions.

These results validate AGLS as a new state-of-the-art method for semantic-preserving adversarial attacks, particularly in scenarios mimicking real-world prompt optimization pipelines.

E. Ablation Studies

Hyperparameter Sensitivity We analyze the sensitivity of AGLS to two key hyperparameters: the global similarity threshold σ and the beam-width adjustment ratio ω . Table III shows results on the GSM8K dataset across four representative models. The combination $\sigma = 0.8, \omega = 0.7$ consistently yields the highest ASR. Lowering σ to 0.3 increases attack accuracy ($\mathcal{A}_{\text{attack}}$) but at the cost of reduced ASR, as perturbations become more detectable (semantic similarity drops). Conversely, setting $\omega = 0.3$ (narrower beam) often leads to ASR degradation, confirming that a broader candidate pool is beneficial for finding effective perturbations.

Dynamic vs. Static Search Strategy We ablate the core adaptive mechanism by comparing our dynamic candidate adjustment against a static strategy that always selects the median-ranked candidate ($c_{\lfloor k/2 \rfloor}$). Results in Table IV show that the dynamic strategy achieves significantly higher ASR across all models—e.g., improving from 36.36% to 90.91% for GPT-4. This validates the necessity of real-time semantic feedback for balancing attack potency and stealth.

TABLE I: Attack performance of AGLS across different LLMs and datasets, including numerical and text response tasks

Models	GSM8K				Math QA				SQuAD				Strategy QA			
	A_{clean}	A_{attack}	ASR \uparrow	T_{avg} (s)	A_{clean}	A_{attack}	ASR \uparrow	T_{avg} (s)	A_{clean}	A_{attack}	ASR \uparrow	T_{avg} (s)	A_{clean}	A_{attack}	ASR \uparrow	T_{avg} (s)
gpt-4o-latest	47.50	15.00	68.42	2.21	32.40	22.33	31.08	0.82	54.52	43.49	20.23	0.63	55.33	43.45	21.47	0.70
gpt-4	27.50	2.50	90.91	2.31	48.50	34.50	28.87	1.12	51.84	32.43	37.44	0.78	57.33	43.70	23.77	0.56
llama3.1-8B	17.50	7.50	57.14	1.49	8.67	4.67	46.12	0.23	33.78	30.43	9.92	0.56	43.33	42.33	2.31	1.07
llama3.1-70B	47.50	5.00	89.47	1.72	13.67	11.67	17.12	1.27	47.83	27.33	42.86	1.13	47.33	35.68	24.61	1.42
llama3.2-3B	47.50	5.00	89.47	1.38	3.00	1.00	66.67	0.15	28.09	22.41	20.22	0.36	34.33	29.33	14.56	0.23
qwen2.5-7B	15.00	7.50	50.00	1.91	10.33	2.00	80.64	0.69	20.74	19.40	6.46	0.72	55.00	36.57	33.51	0.46
qwen2.5-14B	22.50	5.00	77.78	2.49	64.86	39.18	39.59	1.16	30.77	30.43	1.10	1.12	55.67	44.43	20.19	0.68
gemma2-9B	12.50	7.50	40.00	1.24	7.67	3.67	52.15	0.44	34.11	27.42	19.61	0.45	54.00	48.36	10.44	0.17
gemma2-27B	70.00	10.00	85.71	0.85	10.33	2.00	80.64	1.28	44.15	37.79	14.41	0.85	64.00	59.67	6.77	0.21

TABLE II: Comparison of AGLS with state-of-the-art adversarial attack methods on GPT-3.5-turbo. Note that a higher ASR (Attack Success Rate %) is better.

Models	SQuAD2.0			Math			SVAMP		
	A_{clean}	A_{attack}	ASR	A_{clean}	A_{attack}	ASR	A_{clean}	A_{attack}	ASR
BertAttack [12]	71.16	24.67	65.33	72.30	44.82	38.01	88.00	77.41	12.03
DeepWordBug [32]	71.16	65.68	6.72	72.30	48.36	33.11	88.00	64.83	26.33
TextFooler [33]	71.16	15.60	78.59	72.30	46.80	35.27	88.00	43.62	50.43
TextBugger [34]	71.16	60.14	16.08	72.30	47.75	33.96	88.00	60.72	20.77
Stress Test [35]	71.16	70.66	1.78	72.30	39.59	45.24	88.00	-	-
CheckList [35]	71.16	68.81	3.64	72.30	36.90	48.96	88.00	-	-
G2PIA [19]	71.16	14.00	79.50	72.30	52.37	27.57	88.00	69.42	21.11
TDA [20]	71.16	14.91	83.02	72.30	33.39	53.82	88.00	64.87	20.28
BeamAttack [7]	71.16	48.33	32.08	72.30	52.25	27.73	88.00	47.50	46.02
Our Method	71.16	12.48	83.09	72.30	28.50	60.61	88.00	33.33	62.13

TABLE III: Hyperparameter sensitivity analysis for σ and ω on GSM8K.

Target Models	σ	ω	A_{clean}	A_{attack}	ASR
llama3.1-8B	0.3	0.7	17.50	8.50	51.43
	0.8	0.7	17.50	7.50	57.14
	0.8	0.3	17.50	15.00	14.29
Qwen2.5-7B	0.3	0.7	15.00	9.50	36.67
	0.8	0.7	15.00	7.50	50.00
	0.8	0.3	15.00	15.00	0.00
Gemma2-9B	0.3	0.7	50.00	15.00	70.00
	0.8	0.7	50.00	5.00	90.00
	0.8	0.3	50.00	17.50	65.00
gpt-4-0125-preview	0.3	0.7	27.50	12.50	54.55
	0.8	0.7	27.50	2.50	90.91
	0.8	0.3	27.50	9.75	64.55

TABLE IV: Effect of dynamic candidate adjustment vs. static median selection on GSM8K.

Type	Target Models	A_{clean}	A_{attack}	ASR
Dynamic	llama3.1-8B	17.50	7.50	57.14
	llama3.1-8B	17.50	15.00	14.23
Dynamic	llama3.2-3B	47.50	5.00	89.47
	llama3.2-3B	47.50	27.75	41.58
Dynamic	qwen2.5-7B	15.00	7.50	50.00
	qwen2.5-7B	15.00	12.50	16.67
Dynamic	gpt-4-0125-preview	27.50	2.50	90.91
	gpt-4-0125-preview	27.50	17.50	36.36

Impact of Search Scope We vary the size of the initial candidate pool (search scope) from 2,000 to 16,000. Table V indicates that a scope of 13,000 generally yields the best trade-off between ASR and efficiency. Smaller scopes (e.g., 2,000) limit candidate diversity, reducing effectiveness; larger scopes (16,000) increase computation with diminishing returns. This demonstrates AGLS’s ability to effectively utilize a bounded search space.

F. Discussion

The experimental results collectively demonstrate that AGLS is a highly effective and practical attack against LLMs equipped with visible prompt optimizers. Its superiority stems from three design principles:

- **Constraint-Aware Search:** By explicitly modeling and enforcing semantic similarity constraints at each generation step, AGLS avoids the “semantic collapse” common in one-shot attacks, producing more natural adversarial examples.
- **Adaptive Feedback:** The dynamic adjustment mechanism allows AGLS to recover from suboptimal local decisions, mitigating the early-commitment bias inherent in static beam search methods.
- **Efficient Black-Box Operation:** AGLS achieves state-of-the-art performance without requiring model gradients, massive query budgets, or white-box access, making it directly applicable to real-world commercial systems.

The lower ASR on textual QA tasks suggests potential avenues for improvement, such as incorporating discourse-level constraints or leveraging stronger sentence encoders. Nevertheless, AGLS establishes a strong baseline for semantic-preserving hijacking attacks and underscores the urgent need to harden prompt optimization modules against such adversarial manipulation.

V. CONCLUSION

This paper introduces Adaptive Greedy Local Search, an effective adversarial attack targeting visible prompt optimizers in LLMs. By simulating the multi-stage rewriting process and incorporating adaptive semantic feedback, AGLS crafts perturbations that preserve meaning while misleading model behavior. Extensive experiments demonstrate AGLS’s superior attack success rates compared to existing semantic-preserving methods. Our work reveals a critical vulnerability in

TABLE V: Attack success rate with varying AGLS search scope (candidate pool size).

Models	Search Scope	GSM8K			SQUAD			SVAMP		
		$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR \uparrow	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR \uparrow	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR \uparrow
Llama3.2-3B	2000	55.00	10.00	81.82	26.76	22.07	17.53	43.00	8.67	79.84
	6000	47.50	2.50	94.74	26.09	23.08	11.54	40.33	7.00	82.64
	10000	25.00	2.50	90.00	25.08	21.74	13.32	38.00	7.33	80.71
	13000	42.50	2.50	94.12	25.42	17.48	31.24	39.00	6.00	84.62
	16000	40.00	2.50	93.75	25.08	21.40	14.67	36.00	7.33	79.64
Qwen2.5-14B	2000	10.00	7.50	2.50	32.78	28.43	13.27	71.67	27.00	62.33
	6000	22.50	7.50	66.67	31.10	29.43	5.37	74.33	28.00	62.33
	10000	28.34	7.50	73.54	33.11	29.10	12.11	76.33	26.67	65.06
	13000	17.50	12.50	28.57	32.44	26.67	17.79	73.33	26.00	64.54
	16000	17.50	7.50	57.14	32.11	30.10	6.26	75.00	29.33	60.89

transparency-enhancing optimization modules and underscores the need for security-aware design in deployable LLM systems. Future work will extend AGLS to multimodal and multi-turn settings and explore corresponding defense mechanisms.

REFERENCES

- [1] Microsoft, “Bing copilot: Your ai companion for search,” 2023.
- [2] Anthropic, “Claude.ai: Project ideas to detailed plan auto-expansion feature,” <https://claude.ai>, 2024, December 02, 2025.
- [3] Perplexity AI, “Perplexity pro: Real-time “rephrase for better results” mechanism,” 2024, December 02, 2025.
- [4] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson, “Universal and transferable adversarial attacks on aligned language models,” *arXiv:2307.15043*, 2023.
- [5] Narek Maloyan and Dmitry Namiot, “Adversarial attacks on llm-as-a-judge systems: Insights from prompt injections,” *arXiv:2504.18333*, 2025.
- [6] YunSeok Choi, Hyojun Kim, and Jee-Hyong Lee, “Tabs: Efficient textual adversarial attack for pre-trained nl code model using semantic beam search,” in *EMNLP 2022*, 2022.
- [7] Hai Zhu, Qinyang Zhao, and Yuren Wu, “Beamattack: Generating high-quality textual adversarial examples through beam search and mixed semantic spaces,” in *PAKDD*, 2023.
- [8] Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela, “Gradient-based adversarial attacks against text transformers,” *arXiv:2104.13733*, 2021.
- [9] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou, “Hotflip: White-box adversarial examples for text classification,” 2018.
- [10] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh, “Autoprompt: Eliciting knowledge from language models with automatically generated prompts,” *arXiv:2010.15980*, 2020.
- [11] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh, “Universal adversarial triggers for attacking and analyzing nlp,” *arXiv:1908.07125*, 2019.
- [12] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu, “BERT-ATTACK: adversarial attack against BERT using BERT,” in *EMNLP*, 2020.
- [13] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, “Semantically equivalent adversarial rules for debugging NLP models,” in *ACL*, 2018.
- [14] Jan Clusmann, Dyke Ferber, Isabella C Wiest, Carolin V Schneider, Titus J Brinker, Sebastian Foersch, Daniel Truhn, and Jakob N Kather, “Prompt injection attacks on large language models in oncology,” *arXiv:2407.18981*, 2024.
- [15] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz, “Not what you’ve signed up for: Compromising real-world llm-integrated applications with indirect prompt injection,” in *Proceedings of the 16th ACM workshop on artificial intelligence and security*, 2023, pp. 79–90.
- [16] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang, “Dynamic backdoor attacks against machine learning models,” in *EuroS&P*, 2022.
- [17] Jun Yan, Vikas Yadav, Shiyang Li, Lichang Chen, Zheng Tang, Hai Wang, Vijay Srinivasan, Xiang Ren, and Hongxia Jin, “Backdooring instruction-tuned large language models with virtual prompt injection,” in *NAACL*, 2024.
- [18] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami, “Practical black-box attacks against machine learning,” in *ACCC*, 2017.
- [19] Chong Zhang, Mingyu Jin, Qinkai Yu, Chengzhi Liu, Haochen Xue, and Xiaobo Jin, “Goal-guided generative prompt injection attack on large language models,” in *ICDM*. IEEE, 2024, pp. 941–946.
- [20] Chong Zhang, Mingyu Jin, Dong Shu, Taowen Wang, Dongfang Liu, and Xiaobo Jin, “Target-driven attack for large language models,” in *ECAI*, 2024.
- [21] Taowen Wang, Zheng Fang, Haochen Xue, Chong Zhang, Mingyu Jin, Wujiang Xu, Dong Shu, Shanchieh Yang, Zhenting Wang, and Dongfang Liu, “Large vision-language model security: A survey,” in *FCS*, 2024.
- [22] Tengfei Zhao, Zhaocheng Ge, Hanping Hu, and Dingmeng Shi, “Generating natural language adversarial examples through an improved beam search algorithm,” *arXiv:2110.08036*, 2021.
- [23] Bin Zhu, Zhaoquan Gu, Yaguan Qian, Francis Lau, and Zhihong Tian, “Leveraging transferability and improved beam search in textual adversarial attacks,” *Neurocomputing*, 2022.
- [24] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al., “Training verifiers to solve math word problems,” *arXiv:2110.14168*, 2021.
- [25] Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi, “Mathqa: Towards interpretable math word problem solving with operation-based formalisms,” *arXiv:1905.13319*, 2019.
- [26] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang, “Squad: 100,000+ questions for machine comprehension of text,” in *EMNLP*, 2016.
- [27] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant, “Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies,” 2021.
- [28] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altmenschmidt, Sam Altman, Shyamal Anadkat, et al., “Gpt-4 technical report,” *arXiv:2303.08774*, 2023.
- [29] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al., “The llama 3 herd of models,” *arXiv:2407.21783*, 2024.
- [30] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al., “Qwen2 technical report,” *arXiv:2407.10671*, 2024.
- [31] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al., “Gemma 2: Improving open language models at a practical size,” *arXiv:2408.00118*, 2024.
- [32] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi, “Black-box generation of adversarial text sequences to evade deep learning classifiers,” in *SPW*, 2018.
- [33] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits, “Is bert really robust? a strong baseline for natural language attack on text classification and entailment,” in *AAAI*, 2020.
- [34] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang, “Textbugger: Generating adversarial text against real-world applications,” *arXiv:1812.05271*, 2018.
- [35] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh, “Beyond accuracy: Behavioral testing of nlp models with checklist,” *arXiv:2005.04118*, 2020.
- [36] Kaijie Zhu, Qinlin Zhao, Hao Chen, Jindong Wang, and Xing Xie, “Promptbench: A unified library for evaluation of large language models,” *JMLR*, 2024.

Semantic-Preserving Prompt Hijacking: A Black-Box Adversarial Attack on Auto-Prompt Optimization

Anonymous

I. DATASET DETAILS AND EVALUATION PROTOCOL

A. Datasets

GSM8K [1]: A collection of linguistically diverse grade-school math word problems that require multi-step reasoning. The dataset contains 8,000 problems with corresponding step-by-step solutions, making it suitable for evaluating mathematical reasoning under adversarial conditions.

Math QA [2]: A dataset of math word problems annotated with operation-based formalisms, explanations, and multiple-choice options. It emphasizes interpretable problem-solving and is used to assess numerical reasoning robustness.

Strategy QA [3]: A question-answering benchmark requiring implicit multi-hop reasoning, where answers are not directly extractable from the provided context, thereby evaluating deeper comprehension and inference skills.

SVAMP [4]: A question-answering dataset specifically designed to test the ability of models to solve simple math word problems, often used to evaluate generalization and robustness.

SQuAD [5]: The Stanford Question Answering Dataset contains over 100,000 question-answer pairs derived from Wikipedia articles. It is widely used for evaluating reading comprehension and extractive QA capabilities.

SQuAD 2.0 [6]: An extended version of SQuAD that introduces unanswerable questions, challenging models to distinguish between answerable and unanswerable queries, thus testing reasoning and discrimination abilities.

B. Victim Models Details

Our experiments encompass both commercial and open-source large language models to ensure broad evaluation coverage:

GPT-4 Series [7]: We test GPT-4-Turbo and GPT-4o, two of the most capable commercial models developed by OpenAI, known for their strong reasoning and instruction-following abilities.

Llama Series [8] We include multiple versions of Meta’s open-source models: Llama-3.1-8B, Llama-3.1-70B, Llama-3.2-3B, and Llama-3.3-70B, covering a range of parameter scales and architectural variations.

Qwen2.5 Series [9]: We evaluate Alibaba’s open-source models Qwen2.5-0.5B, Qwen2.5-7B, and Qwen2.5-14B, which are known for strong multilingual and reasoning performance.

Gemma2 [10]: We test Google’s efficient open models Gemma2-2B and Gemma2-9B, designed for high performance at practical model sizes.

C. Evaluation Criteria

We adopt distinct correctness criteria for textual and numerical responses to accurately reflect task requirements:

Textual Responses: Let A be the set of words in the model’s response and B the set of words in the ground-truth answer

- if $|B| \geq 3$: The response is considered correct if $|A \cap B| > 2$.
- if $|B| < 3$: The response is correct only if $B \subset A$.

Numerical Responses: A response is considered correct if the extracted numerical value exactly matches the ground-truth answer. Any deviation, including formatting differences, is treated as incorrect.

D. Prompt Templates

1) Ollama text responses prompts:

Please give me a brief answer directly and promise to answer in English:

2) Ollama numerical responses prompts:

Give me the numerical answers directly, without giving the intermediate steps:

3) OpenAI text responses prompts:

Please give me a brief answer directly to the following questions and promise to answer in English:

4) OpenAI numerical responses prompts:

Give me the numerical answers directly in the following questions, without giving the intermediate steps:

II. ANALYSIS OF SEMANTIC SIMILARITY DYNAMICS IN AGLS

A. Semantic Trajectories During Attack Generation

To understand how AGLS maintains semantic coherence while introducing adversarial perturbations, we analyze the similarity trajectories during the generation of attack samples. Figure 1 and Figure 2 illustrate the positional adjustment trends and semantic similarity changes, respectively, when attacking the same dataset (GSM8K) across different victim models. Key Observations are as follows:

Consistent Adjustment Patterns: Despite differences in model architecture and scale, AGLS exhibits similar positional adjustment trends (Figure 1). Inflection points in candidate

selection occur at roughly the same generation steps, indicating that the adaptive feedback mechanism operates consistently across models.

Controlled Semantic Drift: Figure 2 shows that semantic similarity follows a predictable trajectory, with inflection points typically occurring around steps 18–19. While the magnitude of similarity variation differs across models (e.g., larger changes in Llama3.2-3B and Gemma2-9B), the overall trend remains stable, confirming that AGLS effectively balances semantic preservation and adversarial intent.

B. Search Scope Analysis

The search scope—defined as the size of the candidate pool considered at each step—directly influences AGLS’s efficiency and effectiveness. Table I and Figures 3–6 present a comprehensive analysis of how varying the search scope affects the Attack Success Rate (ASR) across models and datasets. The key findings are listed as follows:

Optimal Scope: A search scope of 13,000 candidates generally yields the highest ASR while maintaining reasonable computational cost. This scope provides sufficient lexical diversity without introducing excessive noise.

Scope-Performance Relationship: Narrower scopes (e.g., 2,000) limit candidate diversity and reduce ASR, while excessively broad scopes (e.g., 16,000) offer diminishing returns and increase inference time.

Model-Specific Behavior: Some models, such as Qwen2.5-14B, achieve peak performance at a scope of 10,000 on certain datasets (GSM8K, SVAMP), suggesting that the optimal scope may depend on model architecture and task type.

C. Additional Attack Results

Table II presents extended experimental results for models not included in the main paper due to space constraints. Key takeaways include:

Smaller Models Remain Highly Vulnerable: Even very small models such as Qwen2.5-0.5B and Gemma2-2B exhibit high ASR on numerical tasks (e.g., 92.31% on GSM8K for Qwen2.5-0.5B), indicating that model scale alone does not confer robustness against semantic hijacking.

Consistent Trends Across Tasks: The relative performance of AGLS remains stable across textual and numerical tasks, reinforcing its generality as an attack framework.

D. Qualitative Examples

Below we provide a concrete example illustrating how AGLS subtly alters an input question while preserving its surface semantics.

Original Question (GSM8K):

Isabelle works in a hotel and runs a bubble bath for each customer who enters the hotel. There are 13 rooms for couples and 14 single rooms. For each bath that is run, Isabelle needs 10ml of bubble bath. If every room is filled to maximum capacity, how much bubble bath, in millilitres, does Isabelle need?

AGLS-Perturbed Question:

Isabelle **is employed** in a hotel and **prepares** a bubble bath for each customer who **arrives** at the hotel. There **exist** 13 rooms for couples and 14 single **accommodations**. For each bath that is prepared, Isabelle **requires** 10ml of bubble bath. If every room is **occupied** to maximum capacity, how much bubble bath, in millilitres, **does Isabelle require**?

The perturbed version substitutes verbs (“works” > “is employed”, “runs” > “prepares”) and nouns (“rooms” > “accommodations”) with semantically similar alternatives. These changes are grammatically sound and preserve the original problem structure, yet they are sufficient to mislead certain LLMs into producing incorrect answers, demonstrating the stealth and effectiveness of AGLS.

III. IMPLEMENTATION AND REPRODUCIBILITY

All experiments were conducted in a strictly black-box setting. We used **bert-base-uncased** as the backbone model for candidate generation and similarity computation. Hyperparameters were fixed as follows: semantic similarity threshold $\sigma = 0.80$ (BERTScore F1), candidate pool size = 13,000, beam width = 4, and adaptive step size $s = 50$. Code, detailed configuration files, and a subset of generated adversarial examples are available at our anonymized repository: <https://anonymous.4open.science/r/EDA5G6S8>.

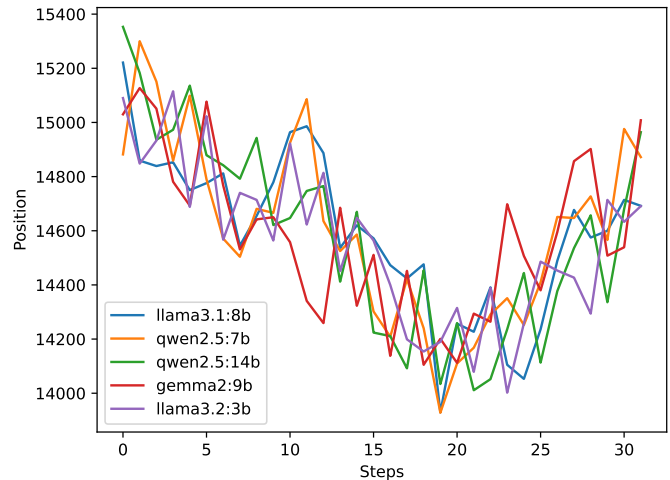


Fig. 1: Dynamic optimization beam search position change trend

REFERENCES

- [1] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al., “Training verifiers to solve math word problems,” *arXiv:2110.14168*, 2021.
- [2] Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi, “Mathqa: Towards interpretable math word problem solving with operation-based formalisms,” *arXiv:1905.13319*, 2019.

TABLE I: The result of attack success rate via the AGLS search scope changes experiments (Llama3.2-3B/8B, Qwen2.5-7B/14B).

Models	Search scope	GSM8K			SQUAD			SVAMP		
		$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR \uparrow	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR \uparrow	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR \uparrow
Llama3.2-3B	2000	55.00	10.00	81.82	26.76	22.07	17.53	43.00	8.67	79.84
	6000	47.50	2.50	94.74	26.09	23.08	11.54	40.33	7.00	82.64
	10000	25.00	2.50	90.00	25.08	21.74	13.32	38.00	7.33	80.71
	13000	42.50	2.50	94.12	25.42	17.48	31.24	39.00	6.00	84.62
Llama3.1-8B	2000	27.50	12.50	54.54	32.11	29.10	9.37	20.33	10.67	47.52
	6000	20.00	5.00	75.00	31.10	26.76	13.95	18.33	6.33	65.47
	10000	22.50	2.50	88.89	31.10	29.43	11.11	18.33	10.67	41.79
	13000	25.00	2.50	90.00	35.45	29.77	16.02	16.33	10.00	38.76
Qwen2.5-7B	2000	15.00	10.00	33.33	20.40	17.73	13.09	53.33	20.00	62.50
	6000	10.00	6.25	37.50	20.74	18.73	9.69	54.33	18.67	65.64
	10000	17.50	10.00	42.86	20.74	19.73	4.87	54.67	19.67	64.02
	13000	15.00	7.50	50.00	20.40	14.36	29.61	54.00	16.45	69.54
Qwen2.5-14B	2000	10.00	7.50	2.50	32.78	28.43	13.27	71.67	27.00	62.33
	6000	22.50	7.50	66.67	31.10	29.43	5.37	74.33	28.00	62.33
	10000	28.34	7.50	73.54	33.11	29.10	12.11	76.33	26.67	65.06
	13000	17.50	12.50	28.57	32.44	26.67	17.79	73.33	26.00	64.54
Qwen2.5-14B	16000	17.50	7.50	57.14	32.11	30.10	6.26	75.00	29.33	60.89

TABLE II: Comparison of attack effects of AGLS on different LLMs and datasets

Models	GSM8K				Math QA			
	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR \uparrow	T_{avg}	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR \uparrow	T_{avg}
llama3.1-70B	47.50	15.00	68.42	3.44	13.67	11.33	17.12	0.14
llama3.3-70B	72.50	17.50	75.86	9.45	32.33	23.67	26.79	1.45
qwen2.5-0.5B	32.50	2.50	92.31	1.84	6.00	2.33	61.17	1.81
gemma2-2B	50.00	5.00	90.00	1.49	46.72	34.79	25.54	0.38

Models	SQUAD				Strategy QA			
	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR \uparrow	T_{avg}	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR \uparrow	T_{avg}
llama3.1-70B	49.83	44.15	11.40	2.01s	49.67	47.00	5.38	2.03s
llama3.3-70B	52.17	49.16	5.77	2.53s	50.33	47.33	5.96	1.49
qwen2.5-0.5B	9.03	8.03	11.07	0.64	26.67	25.33	5.02	1.09s
gemma2-2B	19.40	15.97	17.68	0.38	51.67	36.67	29.03	0.26

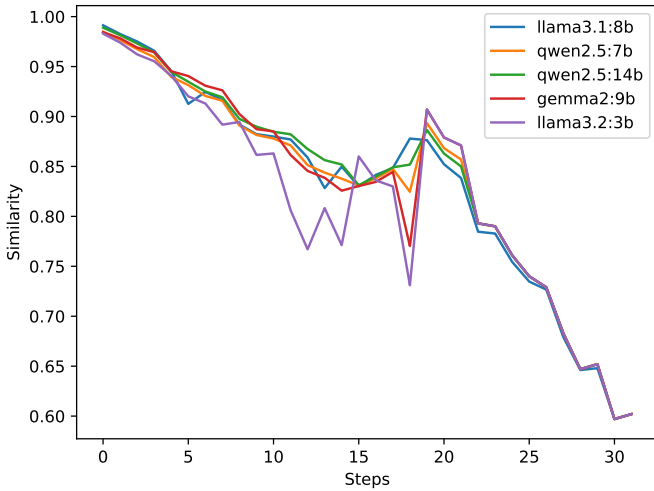
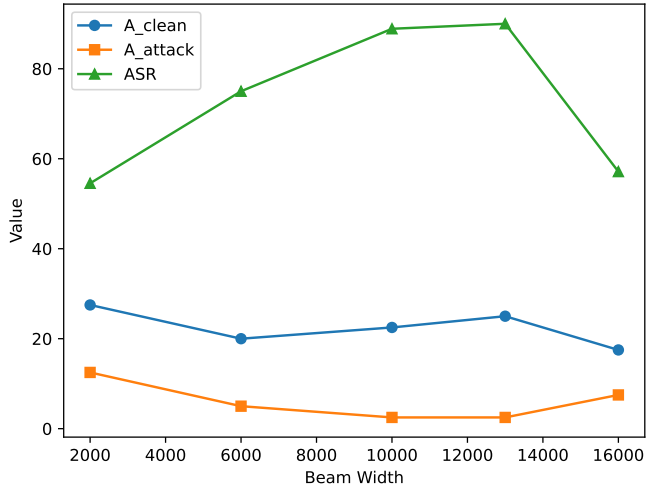
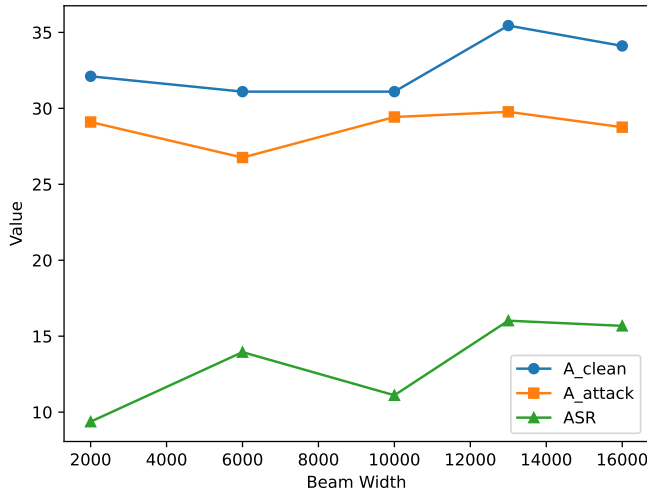


Fig. 2: Dynamic optimization beam search semantic similarity change trend

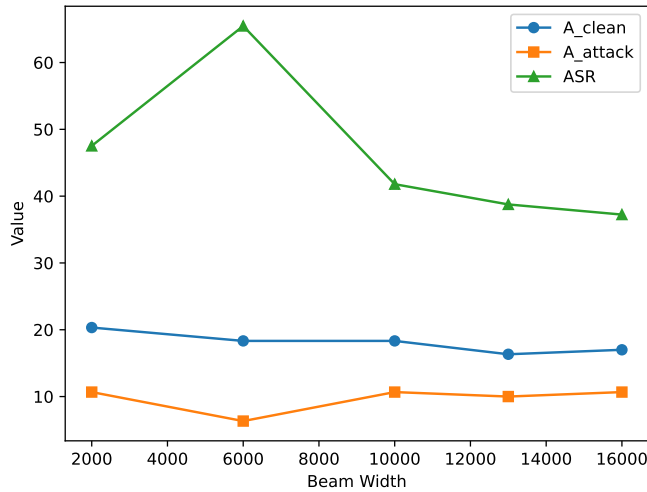
- [3] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant, “Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies,” *arXiv:2101.02235*, 2021.
- [4] Arkil Patel, Satwik Bhattamishra, Navin Goyal, et al., “Are nlp models really able to solve simple math word problems?,” *arXiv:2103.07191*, 2021.
- [5] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang, “Squad: 100,000+ questions for machine comprehension of text,” in *EMNLP*, 2016.
- [6] Pranav Rajpurkar, Robin Jia, and Percy Liang, “Know what you don’t know: Unanswerable questions for squad,” in *ACL*, 2018.
- [7] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Adam B. Santoro, Samuel Chaptot, Aditya Patra, and Ilya Sutskever, “Chatgpt: A language model for conversational agents,” *OpenAI*, 2020.
- [8] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al., “The llama 3 herd of models,” *arXiv:2407.21783*, 2024.
- [9] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al., “Qwen2 technical report,” *arXiv:2407.10671*, 2024.
- [10] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhatipatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al., “Gemma 2: Improving open language models at a practical size,” *arXiv:2408.00118*, 2024.



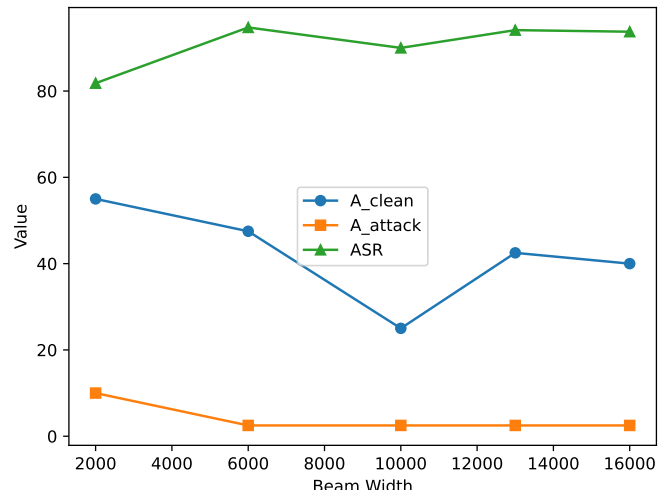
(a) GSM8K on llama3.1-8B



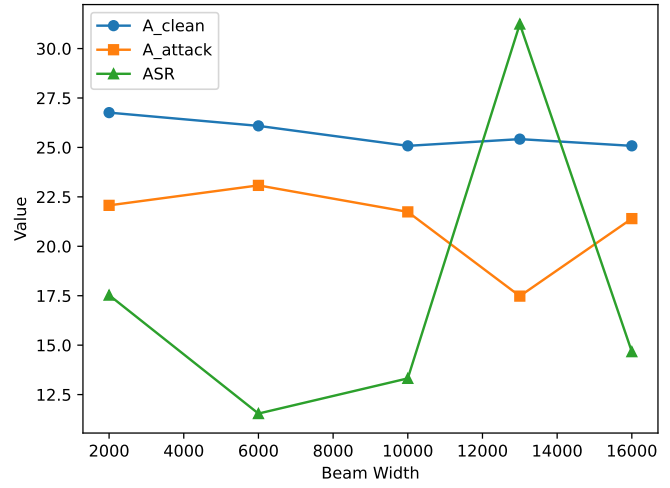
(b) SQUAD on llama3.1-8B



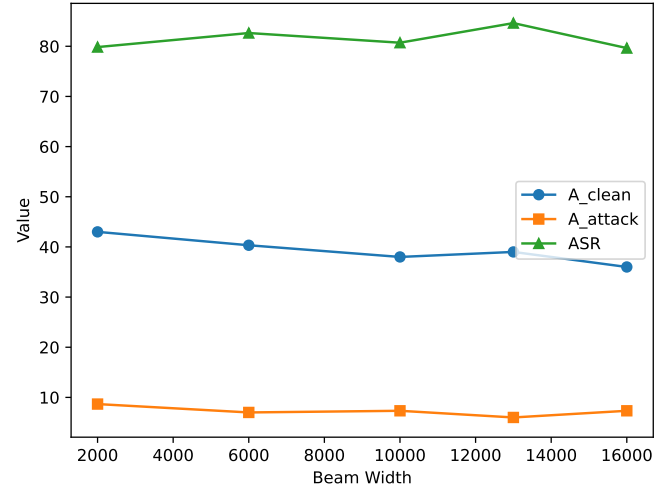
(c) SVAMP on llama3.1-8B



(a) GSM8K on llama3.2-3B



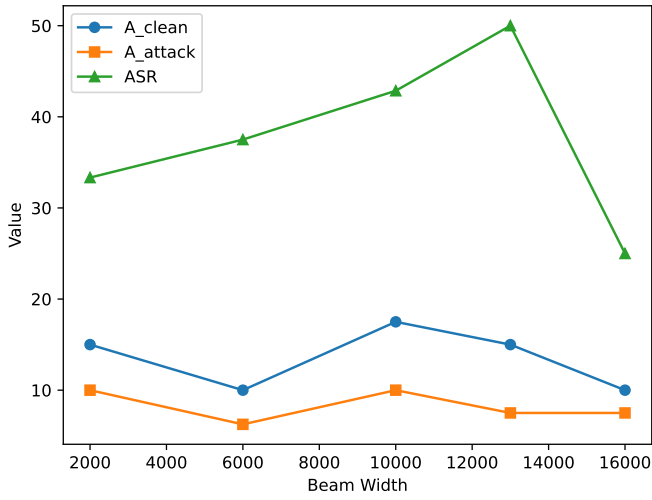
(b) SQUAD on llama3.2-3B



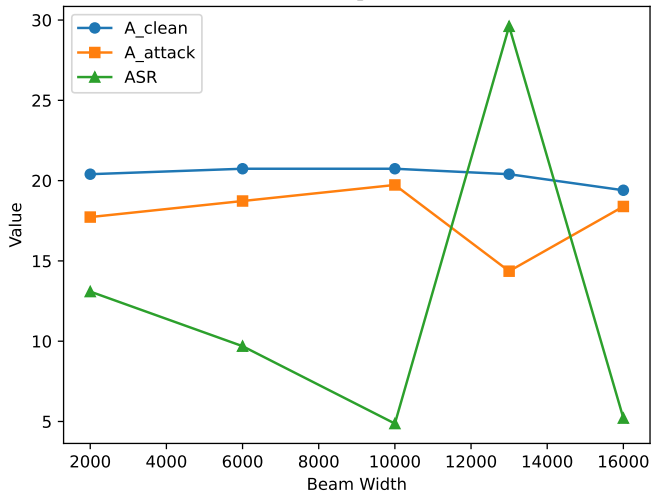
(c) SVAMP on llama3.2-3B

Fig. 3: The relationship between the AGLS search scope and the attack success rate. (Part I)

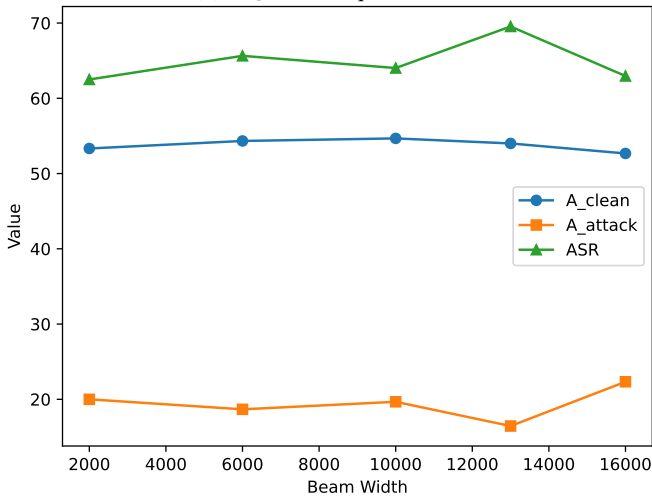
Fig. 4: The relationship between the AGLS search scope and the attack success rate. (Part II)



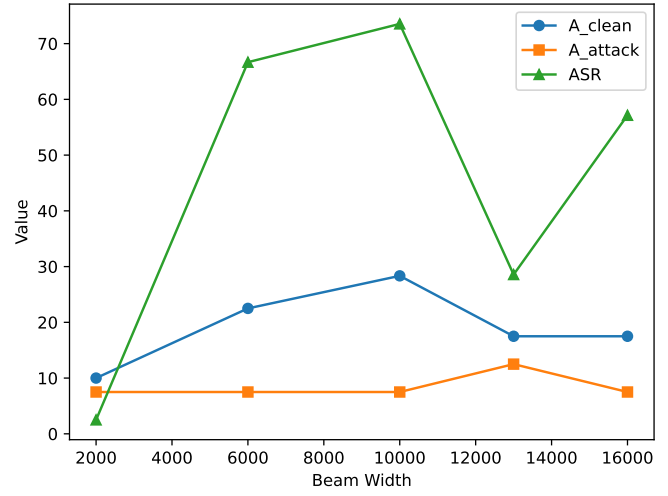
(a) GSM8K on qwen2.5-7B



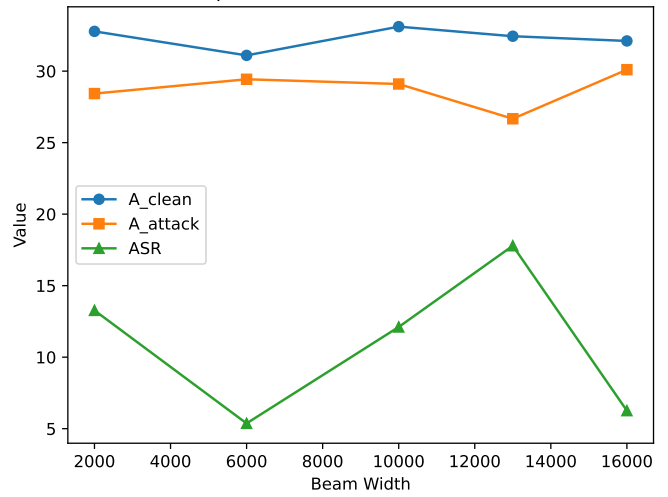
(b) SQUAD on qwen2.5-7B



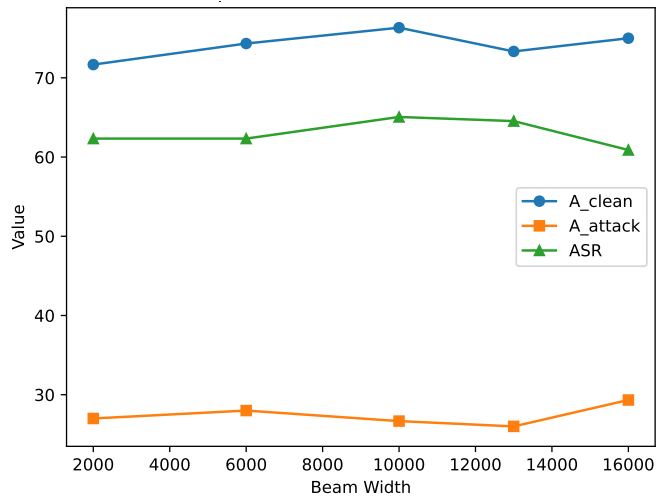
(c) SVAMP on qwen2.5-7B



(a) GSM8K on qwen2.5-14B



(b) SQUAD on qwen2.5-14B



(c) SVAMP on qwen2.5-14B

Fig. 5: The relationship between the AGLS search scope and the attack success rate. (Part III)

Fig. 6: The relationship between the AGLS search scope and the attack success rate. (Part IV)