Semantic-Preserving Adversarial Attacks on LLMs: An Adaptive Greedy Binary Search Approach

Anonymous ACL submission

Abstract

Large Language Models (LLMs) increasingly rely on automatic prompt engineering in graphical user interfaces (GUIs) to refine user inputs and enhance response accuracy. However, the diversity of user requirements often leads to unintended misinterpretations, where automated optimizations distort original intentions and produce erroneous outputs. To address this challenge, we propose the Adaptive Greedy Binary Search (AGBS) method, which simulates common prompt optimization mechanisms while preserving semantic stability. Our approach dynamically evaluates the impact of such strategies on LLM performance, enabling robust adversarial sample generation. Through extensive experiments on open and closed-source LLMs, we demonstrate AGBS's effectiveness in balancing semantic consistency and attack efficacy. Our findings offer actionable insights for designing more reliable prompt optimization systems. Code is available at: https://anonymous.4open.science/r/A5E7202F.

1 Introduction

The rapid deployment of large language models across industries has led to a paradigm shift in human-computer interaction, with leading manufacturers increasingly integrating automatic suggestion optimization directly into the user interface. Systems such as Microsoft's Bing Copilot (Microsoft, 2023) and the data-efficient plug-and-play suggestion enhancement system (PAS) (Zheng et al., 2024) embody this trend, employing complex algorithms to reconstruct user queries to improve response accuracy. These systems typically operate through a multi-stage refinement process that may include vocabulary normalization, intent disambiguation, and context-aware expansion—processes that have shown significant effectiveness in standardized testing environments.

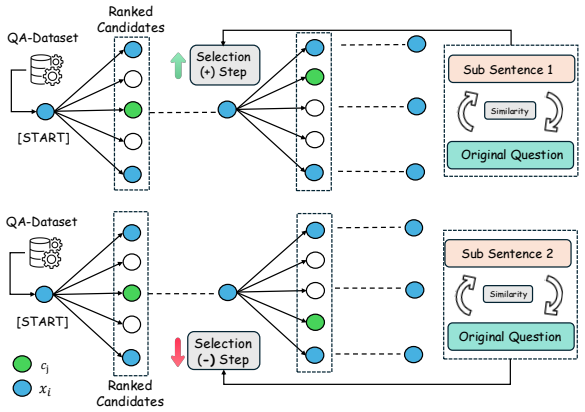


Figure 1: The core of adaptive greedy binary search. After identifying the keyword x_i for each clause and masking it, we use this method to determine the ω_j that replaces x_i and semantically compare the new clause with the original to guide the selection of top-k positions.

However, these automatic suggestion engineering systems have exposed their fundamental limitations in real-world applications. As shown in Figure 1, static optimization strategies often lead to semantic drift—a phenomenon in which iterative suggestion modifications gradually deviate from the user's original intent—in the face of the inherent diversity of user contexts, language patterns, and task requirements. This drift is particularly severe in edge cases, where automated systems may: (1) incorrectly resolve lexical ambiguities, (2) overfit common query patterns while ignoring specific requests, or (3) inadvertently amplify subtle biases present in the training data. More worryingly, these optimizations create attack surfaces for adversarial attacks, as demonstrated by recent research on just-in-time injection vulnerabilities (Zou et al., 2023; Maloyan and Namiot, 2025).

To address these challenges, we propose the Adaptive Greedy Binary Search (AGBS) framework, which outperforms the current state-of-the-art methods in three key dimensions:

Dynamic Semantic Stability: Unlike traditional

beam search methods that apply fixed constraints, AGBS implements an adaptive threshold mechanism that dynamically adjusts the semantic similarity bounds based on real-time analysis of the effects of instantaneous perturbations. This innovation enables precise control of the balance between attack strength and semantic preservation.

Hierarchical Cue Decomposition: AGBS adopts a novel hierarchical decomposition strategy to first identify key semantic units (keywords, clauses, and contextual markers) in the cue, and then applies targeted perturbations while maintaining grammatical and pragmatic coherence.

Through extensive experiments, we demonstrate that AGBS can successfully induce targeted misbehavior in 2400 test cases of commercial LLMs while maintaining an average BERTScore of approximately 0.80 compared to the original prompts. These findings not only validate the effectiveness of our approach but also reveal fundamental limitations of current prompt optimization methods. Our main contributions include:

Formalize the attack surface of just-in-time optimization with a novel adversarial attack strategy achieving high success rates across LLMs and datasets;

Integrate AGBS into automatic prompt learning to improve the concealment and success rate of adversarial attacks against complex scenarios and diversified inputs;

Provide practical guidelines for developing more robust just-in-time optimization systems that balance practicality and security.

2 Related Work

2.1 Adversarial Attacks on Deep Neural Networks

Adversarial attack research mainly changes the input to detect the vulnerability of the model: since deep neural networks are vulnerable to adversarial samples, small perturbations may lead to serious classification errors (Szegedy, 2013). In response to this phenomenon, the fast gradient representation method (FGSM) (Goodfellow et al., 2014) shows how to induce model errors by generating perturbations through gradients. In the field of natural language processing, the discrete nature of text data makes text adversarial attacks more challenging (Li et al., 2018), so researchers have proposed a variety of complex attack methods that can bypass defense techniques (Carlini and Wagner, 2017). On the other hand, adversarial training can effectively improve the robustness of the model, which has been proven to be one of the effective methods to improve the robustness of the model (Mądry et al., 2017). These studies have not only promoted the development of adversarial attack techniques and defense strategies but also had an important impact on the application of artificial intelligence in security-sensitive fields.

2.2 Adversarial Attacks for Large Language Models

Adversarial attacks based on both white-box and black-box pose a significant threat to LLMs. In the white-box scenario, gradient-based distribution attack (GBDA) (Guo et al., 2021) leverages the Gumbel-Softmax technique for optimization with a differentiable adversarial loss, and uses BERTScore and perplexity to enhance the perceptibility and fluency of the attack. HotFlip (Ebrahimi et al., 2018) manipulates adversarial text by mapping text operations to a vector space and computing derivatives, while AutoPrompt (Shin et al., 2020) leverages a gradient-based strategy to optimize the prompt template. Wallace et al. (Wallace et al., 2021) introduced methods for discovering universal adversarial triggers to change model outputs. However, these gradient-based attacks do not apply to closed-source large language models.

Black-box adversarial attacks leverage various techniques to exploit vulnerabilities in NLP models. Ribeiro et al. (Ribeiro et al., 2018) introduced SEAs, a token manipulation method to identify and mitigate excessive sensitivity in models. BERT-Attack (Li et al., 2020) uses context-aware word replacements to subtly modify inputs. Hint injection attacks, such as target hijacking and hint leakage, embed harmful instructions to deceive LLMs, as explored by McKenzie (McKenzie et al., 2023) and Perez & Ribeiro (Perez and Ribeiro, 2022). Other black-box methods include query-free techniques like BadNets (Gu et al., 2017) and model replacement strategies (Papernot et al., 2017). Goal-driven attacks maximize KL divergence, equivalent to increasing Mahalanobis distance between clean and adversarial text embeddings, effectively targeting LLMs (Zhang et al., 2024b,a). These approaches highlight diverse tactics for addressing model security challenges (Wang et al., 2024).

2.3 Adversarial Attacks with Beam Search

The TABS (Choi et al., 2022) method combines semantic beam search with contextual semantic filtering while maintaining the Top-k candidate adversarial sentences and effectively narrows the search space through semantic filtering to maintain semantic consistency. In the improved beam search algorithm (Zhao et al., 2021), each iteration selects the K nodes from the previous iteration, and the word selection range is expanded by backtracking the iteration to improve efficiency. In leveraging transferability and improved beam search (Zhu et al., 2022), multiple words are randomly selected for replacement, multiple candidate sentences are generated for semantic filtering, and finally, the beam width selects the sentence with the highest defense. In Beam Attack (Zhu et al., 2023), the semantic filtering method is used to improve the semantic similarity of candidate words, and the words with the highest similarity are selected for replacement to generate the best adversarial attack samples (Wang et al., 2024). When the above methods generate adversarial samples for the pre-trained language model, semantic pre-screening cannot correct the bias in time, and when semantic constraints are imposed and unqualified samples are re-generated, it will increase the computational cost.

3 Methodology

Scope and Objectives: Starting with a text t consisting of multiple sentences, the goal is to generate a new text t' that can effectively challenge LLMs like ChatGPT while preserving the original meaning of t. Otherwise, we believe the attack text t' targets a text unrelated to t. Here, we use $S(t', t)$ to indicate the similarity between the semantics of the texts t and t'. When the LLM outputs $O(t)$ and $O(t')$ are inconsistent, t' is identified as an adversarial example for O. The target formula is as follows:

$$O(t) = r,\ O(t') = r',\ S(t, t') \in \sigma, \qquad (1)$$

Here, r and r' represent the outputs of the model O for texts t and t', respectively, with r also being the ground truth for text t. We introduce the similarity function $S(.,.)$ and a small threshold σ to evaluate the semantic relationship between the two texts. In our work, our attack satisfies the following conditions:

- **Effective:** The attack ensures that the adversarial text maintains a high semantic similarity with the original input, such that $S(t, t') \in \sigma$. If $S(t, t')$ exceeds or falls below the threshold σ, the adversarial text is deemed invalid.

- **Imperceptible:** Defense mechanisms (e.g., jamming code detection) are often embedded in LLM to make direct attacks easy to identify. Therefore, the perturbations introduced in the adversarial text t' are carefully designed to remain contextually appropriate, ensuring the natural fluency and coherence of the text are not disrupted. For example, by selecting alternative words that have similar semantics but can change the output of the model, t' is difficult to be detected by the defense mechanism, so as to improve the attack concealment and success rate.

3.1 Adversarial Attacks with Adaptive Local Search

Adversarial samples must achieve the desired attack effect while preserving the semantic integrity of the input samples. To induce incorrect model outputs, it is crucial to maintain a controlled semantic deviation between the adversarial examples and the original inputs. Next, we will use a step-by-step adaptive local search method to make the similarity between adversarial samples and input samples approximately equal to θ. Given a sentence input X, it is divided into n sub-clauses x_i, where each sub-clause terminates at a designated position t_i (referred to as a "checking point"). By definition, the positions are ordered as $t_1 < t_2 < \cdots < t_n$. For the original sentence, this can be represented as

$$X = \{x_1, x_2, \ldots, x_T\}, \qquad (1)$$

Where T denotes the total number of sub-clauses in sentence X. We split it into several sub-clauses and defined the endpoint of each sub-clause as a checking point. Checking points for each sentence serves as a positioning sentence adaptive process, and positions punctuation points for calculating similarity with the original sentence. Let the position of the i-th checking point be t_i, where the points' positions are $t_1 < t_2 < \cdots < t_n$ in sub-clause, and n represents the number of checking points. During the generation process, at each checking point t_i, the similarity is computed between the partially generated sentence

$$\hat{X}_{1:t_i} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{t_i}\}, \qquad (2)$$

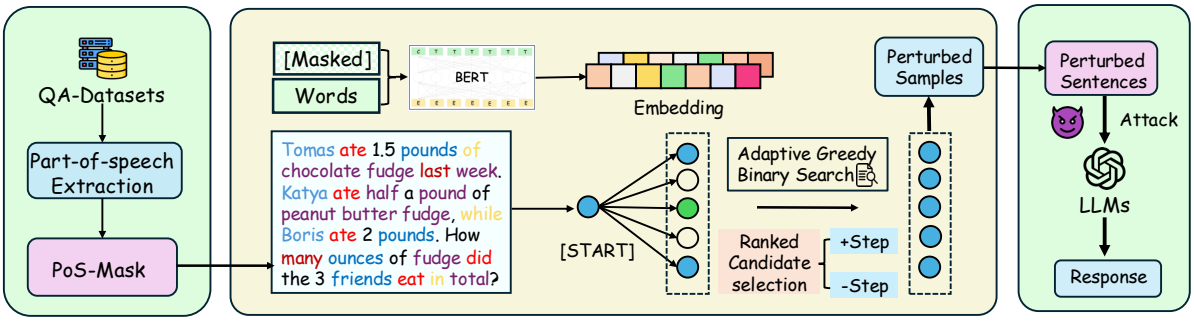


Figure 2: The framework of **Adaptive Greedy Binary Search** for adversarial attack. There are three components in AGBS. (a). PoS (Part-of-Speech) extraction and masking. (b). Generation of AGBS perturbed samples. (c). Perturbed Samples Attack.

and the corresponding segment of the original sentence

$$X_{1:t_i} = \{x_1, x_2, \ldots, x_{t_i}\}. \tag{3}$$

in same-length sub-clause positions of t_i.

At each checking point t_i, we set a similarity threshold σ_{sim} and dynamically adjust the selection of candidate words c_p in Top-k candidate set $\mathcal{C}$ based on the current similarity value $\text{Sim}(X_{1:t_i}, \hat{X}_{1:t_i})$.

Let the set of candidate words at checking point t_i be defined as

$$\mathcal{C}(t_i) = \{c_1, c_2, \ldots, c_k\}, \tag{4}$$

where c_j represents the j-th candidate word in the Top-k ranking, and smaller subscript j represents a higher Top-k ranking for c_j in set $\mathcal{C}$. Initially, we select the middle-ranked word $c_{\lfloor k/2 \rfloor}$ as the starting candidate.

The dynamic adjustment process based on the similarity threshold σ_{th} is as follows:

If the original sub-clause x_i contains masked keywords (tagged as *'VB', 'VBZ', 'VBD', 'VBN', or 'NNS'*), the following adjustments are applied:

$$\sigma_{sim} = \text{Sim}(X_{1:t_i}, \hat{X}_{1:t_i}), \tag{5}$$

indicating that the generated sentence deviates too much from the original. Here we use BertTokenizer to get the embeddings of $X_{1:t_i}$and $\hat{X}_{1:t_i}$and compute the cosine similarity between them to get σ_{sim}.

To make the generated sentence closer to the original, we adjust the candidate word down in the Top-k list by one step position s, perform a "Rank + s " operation, and move the selected position from j to $j - s$,

$$\text{If} \quad \text{Sim}(X_{1:t_i}, \hat{X}_{1:t_i}) < \sigma_{th}, \quad c_\text{p} \leftarrow c_{j-s}. \tag{6}$$

Conversely, if the similarity value in Equation 5 exceeds the threshold σ_{th}, it indicates that the generated sentence is overly similar to the original, potentially failing to achieve the intended attack effect. In this case, we adjust the candidate word up in the Top-k list, performing a "Rank - s" operation,

$$\text{If} \quad \text{Sim}(X_{1:t_i}, \hat{X}_{1:t_i}) > \sigma_{th}, \quad c_\text{p} \leftarrow c_{j+s}. \tag{7}$$

Through the threshold judgment of similarity σ_{th}, the selection order of candidate words is dynamically adjusted so that the generated sentence not only maintains a reasonable similarity with the original sentence but also can achieve the attack goal of modifying the selection of candidate words in real-time based on the similarity between the generated sentence and the original sentence, thereby controlling the reasonableness and attack effectiveness of the generated output.

3.2 Dynamic generation of sub-clauses

At each generation step, the newly generated result $\hat{X}_{t_i}$ is integrated into the masked position of the original sentence $X_1 : t_i$, with the current context $\hat{x}_{t_i}$ updated to x_{m_i}. These updates ensure that the next generation step is based on the newly updated context, gradually optimizing the generation process.

Initially, the masked sentence X_M is given by:

$$X_\text{M} = \{x_1, \ldots, x_{m_i-1}, [\text{MASK}], x_{m_i+1}, \ldots, x_T\}, \tag{8}$$

where [MASK] indicates the masked word at position m_i. For [MASK] position m_i, beam search progressively expands the candidates to generate the complete sentence.

We set the model prediction probability distribution to be $P\left(c \mid X_\text{M}\right), \forall c \in C$, which is the conditional probability of the model for each word C in the vocabulary c, given the input sentence. In this step, BERT is employed to predict the word for the [MASK] position, producing the logits output as follows:

$$\text{logits}(c) = \text{BERT}\left(X_M, x_{m_i}\right) \tag{9}$$

$logits(c)$ is the raw score of the word $c \in C$, we apply softmax to logits to generate a probability distribution normalized over the entire candidate set $\mathcal{C}$,

$$P\left(c \mid X_{\mathrm{M}}\right)=\frac{\exp(\text{logits}(c))}{\sum_{c \in C} \exp(\text{logits}(c))} \tag{10}$$

We sort the probability distribution $P\left(c \mid X_{\mathrm{M}}\right)$ to select the Top-K candidates with the highest probability $P\left(c \mid X_{\mathrm{M}}\right)=\{c_1, c_2, \ldots, c_k\}$. Here we pick c_p as the best candidate, which $\hat{x}_{m_i}=c_p$. At the i-th generation step, the model generates $\hat{x}_{t_i}$ to replace the masked word, updating the sentence:

$$X_{\mathrm{M}}=\{x_1, \ldots, x_{m_i-1}, \hat{x}_{t_i}, x_{m_i+1}, \ldots, x_T\}. \tag{11}$$

Finally, the updated context for the next step is:

$$\hat{X}_{\mathrm{F}}=\hat{X}_{1:t_n}=\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{t_n}\}. \tag{12}$$

where $\hat{X}_{1:t_{i+1}}$ represents the part of the sentence that has been generated so far, including all results up to the current step.

During each step of the generation process, we continuously update the partially generated sentence based on similarity calculations and dynamic candidate word adjustment strategies until the entire sentence is completed. This process continues to perform similarity calculations and candidate adjustments until the last segment of the sentence is generated.

Let the final generated sentence be $\hat{X}_{\mathrm{F}}$, and its generation process can be described by the following formula:

$$\hat{X}_{\mathrm{F}}=\prod_{i=1}^{n} P(\hat{X}_{t_i} \mid X_{\mathrm{M}}, \hat{X}_{1:t_{i-1}}, \mathrm{Sim}(X_{1:t_i}, \hat{X}_{1:t_i})), \tag{13}$$

Here, $\hat{X}_{t_i}$ denotes the word generated at step i, X_M represents the original masked sentence, $\hat{X}_{1:t_i-1}$ corresponds to the portion of the sentence generated up to step t_i, and $\mathrm{Sim}\left(X_{1:t_i}, \hat{X}_{1:t_i}\right)$ evaluates the semantic similarity between the original and generated sentences at position t_i.

4 Experiments

4.1 Experimental Details

Datasets: We selected the following datasets, including the QA scenario numerical response datasets GSM8K (Cobbe et al., 2021), Math QA (Amini et al., 2019), Strategy QA (Geva et al., 2021), and SVAMP (Patel et al., 2021) with numeric responses, and SQuAD (Rajpurkar et al., 2016), SQuAD 2.0 (Rajpurkar et al., 2018), Movie QA (Tapaswi et al., 2016) and Complex Web Questions (Talmor and Berant, 2018) with textual responses in QA scenario. Datasets details will be listed in the Appendix A.

Parameter Settings: We set the semantic similarity threshold to σ= 0.80. The strength of candidate words is 13000, the number of randomly selected questions is 300, and the BERT used is the pre-trained BERT (bert-large-uncased) model. This model has 24 transformer encoding layers, and the dimension of the hidden layer is 1024. There are 16 attention heads in this version of the BERT model. The detailed Appendix C.2 is for the setting of beam width. The response criteria in numbers and text are described in Appendix B.

Victim Models: In the evaluation for AGBS method, we selected ChatGPT-4/4o (Radford et al., 2020), Llama 3.1/3.2 (Dubey et al., 2024), Qwen 2.5 (Yang et al., 2024), Gemma 2 (Team et al., 2024), and Phi-3.5 (Abdin et al., 2024) series of LLMs, where each LLM selected a variety of weight parameter sizes. Appendix A.2 provides a detailed description of the victim models.

Evaluation Metrics: Assume that the test set is D, the set of all question-answer pairs predicted correctly by the LLM model f is T, and $a(x)$ represents the attack sample generated by the clean input. Then we can define the following three evaluation indicators,

- **Clean Accuracy** The Clean Accuracy measures the accuracy of the model when dealing with clean inputs $\mathcal{A}_{\text{clean}}=\frac{|T|}{|D|}$.
- **Attack Accuracy** The Attack Accuracy metric measures the accuracy of adversarial attack inputs $\mathcal{A}_{\text{attack}}=\frac{|\sum_{(x,y)\in T} f(a(x))=y|}{|D|}$.
- **Attack Success Rate (ASR)** The attack success rate indicates the rate at which a sample is successfully attacked. Now we formally describe it as follows ASR $=\frac{|\sum_{(x,y)\in T} f(a(x))\neq y|}{|T|}$ It is worth noting that for the above three measurements, we have the following relationship $\text{ASR}=1-\frac{\mathrm{A}_{\text{attack}}}{\mathcal{A}_{\text{clean}}}$.
- **Average inference time (AVG)** We assume that the time cost to infer sample is T, then our average inference time is T_{avg}, then T_{avg} is: $\mathrm{T}_{avg}=\frac{1}{n}\sum_{i=1}^{n}\mathrm{T}_i$

Algorithm 1 Adaptive Greedy Binary Search (AGBS)

Input: Original sentence $X = \{x_1, x_2, \ldots, x_T\}$, similarity threshold σ_{th}, Top-k candidate set $C = \{c_1, c_2, \ldots, c_k\}$, search range k, masked sentence X_M, similarity function $\text{Sim}(\cdot)$, mask position m_i, checking points t_i.

Output: adversarial sentence X_F

Split X into n sub-clauses based on POS tags (*VB, VBZ, VBD, VBN, NNS*), obtaining $X_{1:t_i}$
Initialize $X_F \leftarrow X$ ▷ Adversarial sentence initialization
Initialize candidate position $c_p \leftarrow c_{\lfloor k/2 \rfloor}$ ▷ Start with middle-ranked word
for Each checking point t_i **do**
    Initialize candidate set $C(t_i) \leftarrow \{c_1, c_2, \ldots, c_k\}$
    Update $\hat{X}_{1:t_i} \leftarrow \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{t_i-1}, c_p\}$ ▷ Integrate current candidate word
    Update $X_M \leftarrow \{x_1, \ldots, x_{m_i-1}, c_p, x_{m_i+1}, \ldots, x_T\}$ ▷ Update masked sentence
    Compute similarity $\sigma_{sim} \leftarrow \text{Sim}(X_{1:t_i}, \hat{X}_{1:t_i})$ ▷ Evaluate semantic similarity
    if $\sigma_{sim} < \sigma_{th}$ **then**
        Adjust $c_p \leftarrow c_{p+s}$ ▷ Move to higher-ranked s steps if $\sigma_{sim} < \sigma_{th}$
    else if $\sigma_{sim} > \sigma_{th}$ **then**
        Adjust $c_p \leftarrow c_{p-s}$ ▷ Move to lower-ranked s steps if $\sigma_{sim} > \sigma_{th}$
    end if
end for
Set $X_F \leftarrow \hat{X}_{1:t_n}$ ▷ Concatenate all generated sub-clauses into final output
return X_F

4.2 Implementation Details

Our parameters are set as follows: In our greedy binary search procedure, we set the upper search range to 13,000 and the semantic threshold to σ= 0.8. This threshold is represented by σ as the threshold to adjust the Top-k candidate position at each step, while we set the clause position and the greedy binary search mechanism at the beginning of the second sentence. The length norm α = 0.7 is the initial value; Step X is set to 50. The embedding shapes of the predicted clause and the main clause are both [1,768].

For our experiments, we established two distinct categories of question-answering (QA) scenarios. The first category, numerical response QA, primarily encompasses datasets such as GSM8K, SVAMP, and Math QA. The second category, text response QA, is represented by datasets including SQuAD, Strategy QA, and Movie QA. To create our experimental environment, we randomly selected 300 question-answer pairs from these datasets to serve as the evaluation range.

4.3 Main Attack Results

We first evaluate the common open-source and closed-source LLMs, mainly to test the attack success rate on various LLMs under the adaptive greedy search method. The main datasets included are the QA datasets of text-type and numerical responses scenarios. The detailed experimental results are as follows in Table 1 and Table 2.

According to the main attack evaluation results in Table 1. We can see that the AGBS method achieves good attack results in numerical responses, QA, and text reply scenarios. From the model's perspective, the overall experiment shows that LLM with a larger parameter number in the same model is relatively more resistant. The AGBS method on the Numerical Response Test shows a high ASR for the GSM8K dataset and all tested LLMS. On the SVAMP dataset, the AGBS method only has a high ASR on the llama 3.1/3.2 and Qwen2.5 LLMs and has a good attack effect. However, it does not have a high ASR on the Gemma2 model. Part of the reason may be that Gemma2 itself does not have a high Clean Accuracy, thus limiting the exploration of the attack success rate. Except for llama3.1-70B in Math QA, which has many parameters, all other models show high ASR. In summary, it can be said that our AGBS method has achieved a good attack effect on the numerical response test.

In other datasets are text response scenarios, which contain SQAUD, Strategy QA, and Movie QA datasets, and the results can be seen in Table 2. This data set shows that the higher ASR occurs in the LLMs with smaller weights under the same model. For example, Qwen 2.5-0.5B and 1.5B

Models	GSM8K				SVAMP				Math QA			
	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	AVG	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	AVG	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	AVG
gpt-4o-latest	47.50	15.00	68.42	2.21s	88.00	33.33	**62.13**	0.55s	32.40	22.33	**31.08**	0.82s
gpt-4-turbo	27.50	2.50	**90.91**	2.31s	84.67	38.00	55.12	0.55s	48.50	34.50	28.87	1.12s
llama3.1-8B	17.50	7.50	57.14	1.49s	17.33	8.00	53.84	0.81s	8.67	4.67	46.12	0.23s
llama3.1-13B	47.50	5.00	89.47	1.72s	47.83	27.33	42.86	1.15s	13.67	11.67	17.12	1.27s
llama3.2-1B	17.50	0.00	**100.00**	1.78s	31.76	3.33	**89.49**	0.97s	7.00	3.00	57.14	0.25s
llama3.2-3B	47.50	5.00	89.47	1.38s	39.33	5.33	86.45	0.83s	3.00	1.00	**66.67**	0.15s
qwen2.5-1.5B	47.50	5.00	**89.47**	1.97s	14.38	9.70	32.55	1.21s	5.67	2.67	52.91	0.75s
qwen2.5-7B	15.00	7.50	50.00	1.91s	53.00	22.00	58.49	1.05s	10.33	2.00	**80.64**	0.69s
qwen2.5-14B	22.50	5.00	77.78	2.49s	76.67	27.67	**63.91**	1.31s	64.86	39.18	39.59	1.16s
gemma2-9B	12.50	7.50	40.00	1.24s	60.27	19.67	**67.58**	0.75s	7.67	3.67	52.15	0.44s
gemma2-27B	70.00	10.00	**85.71**	0.85s	31.44	29.77	5.31	0.21s	10.33	2.00	**80.64**	1.28s
phi3.5-3.8B	22.50	0.00	100.00	3.35s	14.33	6.00	58.13	2.91s	1.00	0.00	100.00	1.73s

Table 1: Comparison of attack effects of AGBS on different LLMs and datasets (Numerical response test)

Models	SQUAD				Strategy QA				Movie QA			
	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	AVG	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	AVG	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR ↑	AVG
gpt-4o-latest	54.52	43.49	20.23	0.63s	55.33	43.45	21.47	0.70s	76.06	56.31	**25.97**	0.81s
gpt-4-turbo	51.84	32.43	**37.44**	0.78s	57.33	43.70	**23.77**	0.56s	77.99	58.91	24.46	0.96s
llama3.1-8B	33.78	30.43	9.92	0.56s	43.33	42.33	2.31	1.07s	66.41	60.62	8.72	0.36s
llama3.1-13B	47.83	27.33	42.86	1.13s	47.33	35.68	**24.61**	1.42s	79.54	75.29	5.34	0.92s
llama3.2-1B	16.39	9.03	**44.91**	1.82s	54.00	46.33	14.20	0.26s	42.08	37.84	**10.08**	0.34s
llama3.2-3B	28.09	22.41	20.22	0.36s	34.33	29.33	14.56	0.23s	64.48	59.07	8.39	0.38s
qwen2.5-1.5B	14.38	9.70	**32.55**	0.38s	43.00	41.00	4.65	0.42s	33.20	30.45	**8.28**	0.38s
qwen2.5-7B	20.74	19.40	6.46	0.72s	55.00	36.57	**33.51**	0.46s	36.68	34.75	5.26	0.69s
qwen2.5-14B	30.77	30.43	1.10	1.12s	55.67	44.43	20.19	0.68s	66.02	62.93	4.68	1.16s
gemma2-9B	34.11	27.42	**19.61**	0.45s	54.00	48.36	**10.44**	0.17s	62.16	59.85	**3.72**	0.44s
gemma2-27B	44.15	37.79	14.41	0.85s	64.00	59.67	6.77	0.21s	69.11	67.57	2.23	0.94s
phi3.5-3.8B	14.33	6.00	58.13	3.36s	20.67	19.67	4.84	2.30s	33.59	31.66	5.75	0.36s

Table 2: Comparison of attack effects of AGBS on different LLMs and datasets (Text response test)

achieve high ASR on our adversarial attack test. However, by increasing the weight of the Llama and Qwen models, for example, to a scale greater than 20B (20 billion parameters), the models show obvious resistance to our AGBS method, and our ASR drops sharply on the above scale models. This phenomenon can be said to be the point at which our future work can improve.

Overall, comparing results in the Table 1 and Table 2 tables, we can observe that AGBS performs better in the numerical response QA scenario than in the text response QA scenario. We believe that direct perturbation adversarial attacks against semantic boundaries may be difficult to perform adversarial attacks against reasoning about semantic perturbations, and we will further explore the semantic change angle decomposition of the AGBS method in Table 3.

4.4 Comparison to other mainstream methods

We compare the mainstream adversarial attack methods and find the intersection of these methods that can be tested to conduct comparative experiments. It contains the adversarial attack generation methods TextFooler (Li et al., 2018), TextBugger (Jin et al., 2020), and DeepWordBug (Gao et al., 2018). There are also methods for assessing model robustness, such as BertAttack (Li et al., 2020), StressTest (Ribeiro et al., 2020), and CheckList (Ribeiro et al., 2020). The above Attack Success Rate experimental data are from PromptBench (Zhu et al., 2024). The main results of this experiment can be seen in Table 3. The Query situation of various methods is also shown in the Table 3. In addition, we set the statistics of the average consumption time of adversarial sample generation and the average successful total semantic similarity for the AGBS method, which can better show the characteristics of our AGBS method. The specific experimental results are shown in Table 2.

The results of our longitudinal comparison experiments on GPT-3.5-turbo are shown in Table 3. Here, we count the query mode of the above attack methods as a supplement so our efficiency comparison range can be determined well in the subsequent experiments. In many similar studies, SQUAD 2.0 and Math QA datasets are selected as our experimental control Baseline. The relevant experimental results show that our AGBS method far exceeds similar classical methods and currently reaches SOTA.

Models	SQuAD2.0			Math			SVAMP		
	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	**ASR**	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	**ASR**	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	**ASR**
BertAttack (Li et al., 2020)	71.16	24.67	65.33	72.30	44.82	38.01	88.00	77.41	12.03
DeepWordBug (Gao et al., 2018)	70.41	65.68	6.72	72.30	48.36	33.11	88.00	64.83	26.33
TextFooler (Jin et al., 2020)	72.87	15.60	78.59	72.30	46.80	35.27	88.00	43.62	50.43
TextBugger (Li et al., 2018)	71.66	60.14	16.08	72.30	47.75	33.96	88.00	60.72	20.77
Stress Test (Ribeiro et al., 2020)	71.94	70.66	1.78	72.30	39.59	45.24	-	-	-
CheckList (Ribeiro et al., 2020)	71.41	68.81	3.64	72.30	36.90	48.96	-	-	-
G2PIA (Zhang et al., 2024b)	68.30	14.00	79.50	72.30	52.37	27.57	88.00	69.42	21.11
Target-Driven (Zhang et al., 2024a)	71.16	14.91	83.02	72.30	33.39	53.82	88.00	64.87	20.28
Our Method	71.16	12.48	**83.09**	72.33	28.50	**60.61**	88.00	33.33	**62.13**

Table 3: Comparison of the effectiveness of the AGBS method with other SOTA adversarial attack methods

4.5 Ablation Study

4.5.1 Parameter Sensitive Study

This section will mainly conduct parameter sensitivity tests on σ and our X. We will conduct specific experiments on multiple sets of parameters under the same dataset and LLM to determine the best set of parameters. Where σ makes us judge the parameter of the critical semantic similarity value, and X is the specific step size that we adjust when adjusting beam search dynamically. While ω is the beam width at each step of our specific beam search, we will explore the three parameters to determine the best combination. We select two open sources and one closed-source common large model to conduct parameter sensitivity tests on the GSM8K dataset. The results are shown in Table 4.

Target Models	σ	ω	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	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	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 4: Hyperparameter sensitivity analysis for σ and ω of AGBS attack.

The results of the sensitivity experiments for hyperparameters are shown in the Table 4. We selected the open-source models llama3.1-8B, Qwen2.5-7B, Gemma2-9B, and OpenAI's closed-source model gpt-4-0125-preview as the target models for our parameter sensitivity experiments. It can be seen from the experimental results that when we take the best ASR as the hyperparameter index when σ and ω are set to 0.8 and 0.7, respectively, our AGBS strategy can play the greatest attack effect, and we choose this group of parameters as our best AGBS strategy parameters.

4.6 Dynamic Optimization Study

In this part of the ablation study, we will explore whether our dynamic optimization strategy truly works with beam search. We will compare the Dynamic and Static strategies. The static strategy is to omit the selection position adjustment of Top-K, fixed middle-ranked position as $\mathcal{C}_{\lfloor k/2 \rfloor}$ as a candidate. The results are shown in Table 5.

Type	Target Models	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	ASR
Dynamic	llama3.1-8B	17.50	7.50	**57.14**
Static	llama3.1-8B	17.50	15.00	14.23
Dynamic	llama3.2-3B	47.50	5.00	**89.47**
Static	llama3.2-3B	47.50	27.75	41.58
Dynamic	qwen2.5-7B	15.00	7.50	**50.00**
Static	qwen2.5-7B	15.00	12.50	16.67
Dynamic	gpt-4-turbo	27.50	2.50	**90.91**
Static	gpt-4-turbo	27.50	17.50	36.36

Table 5: Experimental comparison of dynamic and static strategies on AGBS attack

The experimental results show that the ASR of the dynamic Beam Search strategy is significantly higher than that of the static strategy, especially on GPT-4-turbo and llama3.2-3B.

5 Conclusion

The AGBS method offers an effective approach for adversarial attacks on LLMs by combining automatic prompt engineering with dynamic greedy search, ensuring semantic stability and high attack success rates. It reduces semantic biases from diverse inputs, enhancing attack concealment and effectiveness in QA tasks. Experiments demonstrate AGBS's robustness across various LLMs, including ChatGPT, Llama, Qwen, and Gemma, emphasizing its value in testing and improving LLM security. Future work will extend AGBS to multimodal tasks and multi-turn conversations, addressing vulnerabilities in complex LLM applications and enhancing their resilience.

Limitation

This study is limited to automatic prompt engineering based on beam search and does not involve prompt engineering of other methods. In addition, only the most common number and text response QA scenarios of LLMs are introduced in the application environment, and VQA (Visual Question Answering) and multi-round QA scenarios of LLMs are not practiced.

Ethics Statement

Adversarial attacks against large language models are crucial to enhance their robustness. However, the techniques developed to exploit vulnerabilities in LLMs found in this paper could be used for malicious purposes against LLMs, such as making LLMs produce misinformation or even hallucinations. In short, we aim to find effective ways to attack large language models to encourage the model creator or manager to fix and improve the LLM vulnerabilities to improve the robustness of the LLMs under test.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee.

YunSeok Choi, Hyojun Kim, and Jee-Hyong Lee. 2022. Tabs: Efficient textual adversarial attack for pretrained nl code model using semantic beam search. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5490–5498.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. *Preprint*, arXiv:1712.06751.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *SPW*, pages 50–56. IEEE.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *arXiv preprint arXiv:2101.02235*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*.

Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. 2021. Gradient-based adversarial attacks against text transformers. *Preprint*, arXiv:2104.13733.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *AAAI*, volume 34, pages 8018–8025.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: adversarial attack against BERT using BERT. In *EMNLP*, pages 6193–6202. Association for Computational Linguistics.

Aleksander Mądry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *stat*, 1050(9).

Narek Maloyan and Dmitry Namiot. 2025. Adversarial attacks on llm-as-a-judge systems: Insights from prompt injections. *arXiv preprint arXiv:2504.18333*.

Ian R McKenzie, Alexander Lyzhov, Michael Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Aaron Kirtland, Alexis Ross, Alisa Liu, et al. 2023. Inverse scaling: When bigger isn't better. *arXiv preprint arXiv:2306.09479*.

Microsoft. 2023. Bing copilot: Your ai companion for search. Accessed on 2024-10-10.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *ACCC*, pages 506–519.

Arkil Patel, Satwik Bhattamishra, Navin Goyal, et al. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Fábio Perez and Ian Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *Preprint*, arXiv:2211.09527.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Adam B. Santoro, Samuel Chaplot, Aditya Patra, and Ilya Sutskever. 2020. Chatgpt: A language model for conversational agents. *OpenAI*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 65–70.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *ACL*, pages 856–865.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. *arXiv preprint arXiv:2005.04118*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV au2, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *Preprint*, arXiv:2010.15980.

C Szegedy. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 574–584. Association for Computational Linguistics.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2021. Universal adversarial triggers for attacking and analyzing nlp. *Preprint*, arXiv:1908.07125.

Taowen Wang, Zheng Fang, Haochen Xue, Chong Zhang, Mingyu Jin, Wujiang Xu, Dong Shu, Shanchieh Yang, Zhenting Wang, and Dongfang Liu. 2024. Large vision-language model security: A survey. In *International Conference on Frontiers in Cyber Security*, pages 3–22. Springer.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Chong Zhang, Mingyu Jin, Dong Shu, Taowen Wang, Dongfang Liu, and Xiaobo Jin. 2024a. Target-driven attack for large language models, ccf b. In *European Conference on Artificial Intelligence (ECAI), 2024*.

Chong Zhang, Mingyu Jin, Qinkai Yu, Chengzhi Liu, Haochen Xue, and Xiaobo Jin. 2024b. Goal-guided generative prompt injection attack on large language models. *arXiv preprint arXiv:2404.07234*.

Tengfei Zhao, Zhaocheng Ge, Hanping Hu, and Dingmeng Shi. 2021. Generating natural language adversarial examples through an improved beam search algorithm. *arXiv preprint arXiv:2110.08036*.

Miao Zheng, Hao Liang, Fan Yang, Haoze Sun, Tianpeng Li, Lingchu Xiong, Yan Zhang, Youzhen Wu, Kun Li, Yanjun Shen, Mingan Lin, Tao Zhang, Guosheng Dong, Yujing Qiao, Kun Fang, Weipeng Chen, Bin Cui, Wentao Zhang, and Zenan Zhou. 2024. Pas: Data-efficient plug-and-play prompt augmentation system. *arXiv preprint arXiv:2407.06027*.

Bin Zhu, Zhaoquan Gu, Yaguan Qian, Francis Lau, and Zhihong Tian. 2022. Leveraging transferability and improved beam search in textual adversarial attacks. *Neurocomputing*, 500:135–142.

Hai Zhu, Qinyang Zhao, and Yuren Wu. 2023. Beamattack: Generating high-quality textual adversarial examples through beam search and mixed semantic spaces. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 454–465. Springer.

Kaijie Zhu, Qinlin Zhao, Hao Chen, Jindong Wang, and Xing Xie. 2024. Promptbench: A unified library for evaluation of large language models. *Journal of Machine Learning Research*, 25(254):1–22.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

Appendix

Contents

A Experiment details

A.1 Dataset details

GSM8K: The GSM8K dataset (Cobbe et al., 2021) is a high-quality and linguistically diverse dataset of mathematical word problems introduced by OpenAI. It contains 8000 questions extracted from Google searches, each with 8 similar questions. The GSM8K dataset aims to train the model on these problems to improve its performance when dealing with imperfect matching problems.

Math QA: The Math QA dataset (Amini et al., 2019) by Aliyun comprises curated math word problems with detailed explanations, principles, choices, and solution annotations.

Strategy QA: The Strategy QA (Geva et al., 2021) dataset is specifically designed for question-answering tasks and contains many question-and-answer pairs for training and evaluating question-answering systems.

SVAMP: The SVAMP dataset (Patel et al., 2021) is a dataset for question-answering tasks, which contains many question-and-answer pairs designed to help train and evaluate question-answering systems.

SQUAD: SQuAD (Stanford Question Answering Dataset) (Rajpurkar et al., 2016) is a widely used question-answering dataset. It contains more than 100,000 question-answering pairs derived from Wikipedia articles. The strength of the SQuAD dataset lies in its large scale, high quality, and the diversity of contexts and questions it contains.

SQUAD 2: SQuAD 2.0 (Rajpurkar et al., 2018)is an upgraded version of the SQuAD dataset, which introduces unanswerable questions based on the original answerable questions. These unanswerable questions are similar in form to the

answerable questions, but the paragraphs do not contain answers.

Complex Web Questions: The ComplexWebQuestions (CWQ) (Talmor and Berant, 2018) dataset is based on Freebase, which contains questions and Web Snippet files.

Movie QA: The Movie QA (Tapaswi et al., 2016) dataset contains 14,944 questions about 408 movies, covering multiple question types ranging from simple to complex. The dataset is unique in that it contains a variety of information sources, such as Video clips, subtitles, scripts, and DVS (Described Video Service).

A.2 Victim Models Details

ChatGPT: This dialogue generation model developed by OpenAI can produce conversations that closely mimic human interactions (Radford et al., 2020). By training on a large number of datasets, it gains a lot of knowledge and insights. In the experiments, GPT-4-Turbo and GPT-4-o are selected as the victim models in the OpenAI family.

Llama 3.1/3.2: Llama 3.1 and 3.2 are advanced language models developed by Meta AI (Dubey et al., 2024). Trained on various datasets, Llama improves language understanding and generation capabilities and is suitable for various application scenarios such as chatbots and content authoring. In the experiments, our attack methods will test the 7B, 13B, and 70b versions of llama 3.1 and 1B and 3B versions of llama 3.2.

Qwen2.5: The Qwen 2.5 is a new series of large language models from the Alibaba Group (Yang et al., 2024). It contains the 0.5b, 1.5b, 7b, 72b. In this experiment, we will select Qwen2.5 with the main parameter size for experiments.

Gemma2: Google's Gemma 2 model (Team et al., 2024) is available in three sizes, 2B, 9B, and 27B, featuring a brand-new architecture designed for class-leading performance and efficiency.

Phi3.5: Proposed by Microsoft, Phi-3.5 is a lightweight LLM designed for data analysis and medical diagnosis, featuring high accuracy and scalability (Abdin et al., 2024). This experiment focuses on the Phi-3.5-3.8b model.

B Criteria description on QA scenarios

B.1 Text Response Criteria:

When the length of the answer in the QA data pair is greater than or equal to three words, the LLMs reply and the standard answer in the QA data pair are crossed, and the same word is more than two words, which is correct, and less than or equal to two words, which is wrong. When the answer length in the QA data pair is less than three words, the intersection of the two must contain the answer in the QA data pair. Otherwise, the situation is judged as an error.

Let A and B be the sets of words in the LLM's response and the QA pair's standard answer, respectively. Let $|A \cap B|$ denote the number of common words between A and B.

For answers with three or more words:

$$\text{Correctness} = \begin{cases} \text{Correct} & \text{if } |A \cap B| > 2 \\ \text{Incorrect} & \text{if } |A \cap B| \leq 2 \end{cases} \tag{14}$$

For answers with less than three words:

$$\text{Correctness} = \begin{cases} \text{Correct} & \text{if } B \subseteq A \cap B \\ \text{Incorrect} & \text{if } B \nsubseteq A \cap B \end{cases} \tag{15}$$

B.2 Numerical Response Criteria:

The final numerical result in the LLM's response should be exactly the same as the answer in each QA data pair. Let N_r and N_a be the numerical responses of the dataset's QA pair and standard answers, respectively. So if N_r = N_a, this case is judged to be the correct answer. If the N_r is not equal to N_a, the case is judged to be a wrong answer. The case where the decision is correct is strictly enforced, equal to the decision.

B.3 Detailed Prompts

B.3.1 Ollama text responses prompts

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

B.3.2 Ollama numerical responses prompts

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

B.3.3 OpenAI text responses prompts

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

B.3.4 OpenAI numerical responses prompts

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

C Research on changes of AGBS Semantic Similarity

In the results of changes in AGBS semantic similarity, we will focus on exploring the detailed properties of the AGBS method under these changes. These include the change in semantic similarity when generating attack samples for a single dataset, the correlation between semantic change and attack accuracy under the AGBS strategy, and the change in total ASR by adjusting the limited beam search beam width. This section will set up three kinds of experiments to achieve these three aspects of interpretability exploration.

C.1 Semantic Similarity of Single-Dataset Attack Samples

In this part, we will evaluate the semantic changes of the same dataset under the AGBS method on different LLMs, where we select the GSM8K and SVAMP datasets in llama3.1-8B, qwen2.5:7B, qwen2.5:14B, gemma2:9B, and llama3.2:3B, respectively. Samples of successful attacks on these chosen LLMs generate a change in semantic similarity. The results of the average variation trend of position and semantic similarity for word selection are shown in Figure 3, Figure 4.

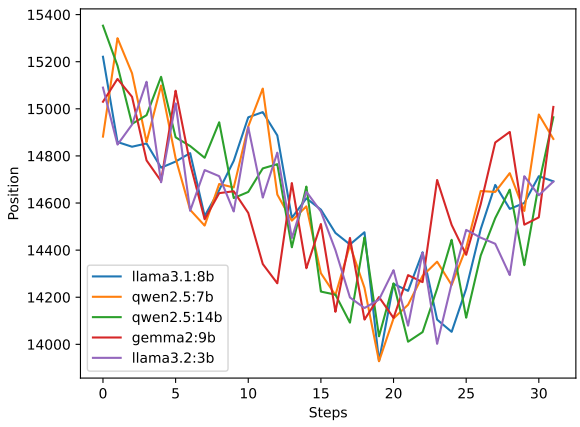


Figure 3: Dynamic optimization beam search position change trend

According to the results shown in the Figure 3, the similarity and selection of position-changing trends could be analyzed by visualizations. Although the semantic changes of the AGBS strategy on the same dataset in different LLMs are slightly different, the overall position trend is still the same. For example, in Figure 3, we can see roughly the same inflectional point when selecting the position of words on AGBS. This approximate inflection point can prove that our dynamic strategy works and that adaptive greedy binary search on different LLMs is effective.

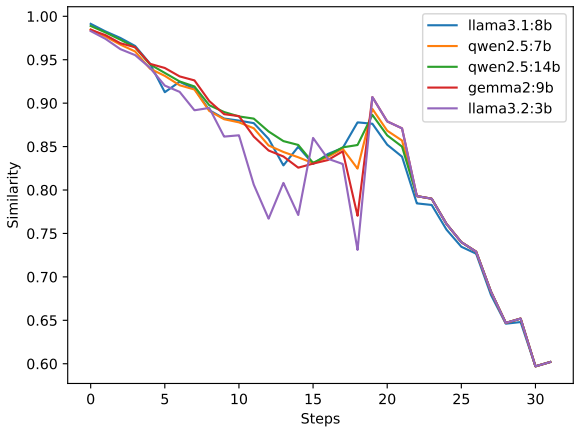


Figure 4: Dynamic optimization beam search semantic similarity change trend

At the same time, from Figure 4 on our semantic similarity change trend graph, we can see that the semantic similarity change of our AGBS strategy on different LLMs has the same trend. For example, a similarity inflection point is generated at about 18 and 19 steps. However, the magnitude of similarity variation is different for each LLM. In our experimental results, the semantic similarity changes on llama3.2:3b and gemma2:9b are the largest, and the changes on qwen2.5:7b, 14b, and llama3.1:8b are more similar, which can be understood as our AGBS strategy is more suitable for the type of LLMs.

C.2 Attack success rate variation under AGBS search scope variation

In this experiment, we explore the correlation between the search scope setting and the attack success rate. The search scope directly affects the efficiency of our algorithm, so exploring the relationship between the search scope and our attack rate is the key to balancing attack effectiveness and algorithm complexity.

We will select the search scope range in order of magnitude to verify the attack on the QA pairs we extracted from the validation set. We must know the relationship between semantic similarity and attack success rate to determine our search scope.

Our results are shown in the Table 6.

According to our experimental results in Table 6, we can observe that under our AGBS strategy, our attack effect reaches the best state when the search scope is 13,000, and the ASRs are almost the highest. However, there are some special cases, such as the performance of GSM8K and SVAMP on Qwen2.5-14B in the QA scenario, both of which have the best performance at a beam width of 10000. The 10000 beam width on Qwen2.5-14B also outperforms the other cases on the GSM8K dataset. The 10000 beam width on Qwen2.5-14B also outperforms the other cases on GSM8K. So, we set the beam width of the AGBS strategy to 13,000 as our optimal hyperparameter value.

Furthermore, we draw the line chart of the attack success rate of the same model under different search scopes. The below Figure 5 and Figure 6 are the ASR variation curves under different search scopes on llama3.2-3B, llama3.1-8B, qwen2.5-7B, and qwen2.5-14B, respectively. It is evident that the attack accuracy is affected by the search scope variation and thus affects the final ASR variation. At a lower search scope, the performance is limited for each dataset under various LLMs, and a reasonable commonality range is between 10,000 and 13,000.

In conclusion, we believe that the search scope of AGBS plays a relatively critical role in determining the performance of our AGBS strategy. This part of our research identifies the best hyperparameters for the AGBS policy.

D Additional experiments results

Below are the results of additional adversarial attack experiments on the AGBS method.

D.1 The Additional experiments of the main experiment

The following results in Table 7 and Table 8 show the additional results of the comparison of attack effects of AGBS on different LLMs of the all dataset. The target model includes llama-3.1-70B, llama-3.3-70B, Qwen2.5-0.5B, and Gemma2-2 B.

Models	Search scope	GSM8K			SQUAD			SVAMP		
		$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	**ASR ↑**	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	**ASR ↑**	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	**ASR ↑**
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
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
	16000	17.50	7.50	57.14	34.11	28.76	15.68	17.00	10.67	37.24
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**
	16000	10.00	7.50	25.00	19.40	18.39	5.21	52.67	22.33	62.96
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

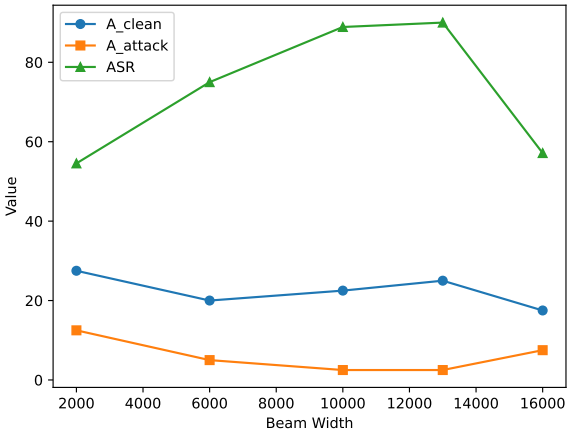
Table 6: The result of attack success rate via the AGBS search scope changes experiments (Llama3.2-3B/8B, Qwen2.5-7B/4)

Models	GSM8K				SVAMP				Math QA			
	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	**ASR ↑**	**AVG**	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	**ASR ↑**	**AVG**	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	**ASR ↑**	**AVG**
llama3.1-70B	47.50	15.00	68.42	3.44s	74.50	32.82	55.95	3.05s	13.67	11.33	17.12	0.14s
llama3.3-70B	72.50	17.50	75.86	9.45s	77.67	40.00	48.50	4.92s	32.33	23.67	26.79	1.45s
qwen2.5-0.5B	32.50	2.50	**92.31**	1.84s	8.70	0.00	**100.00**	1.47s	6.00	2.33	61.17	1.81s
gemma2-2B	50.00	5.00	**90.00**	1.49s	37.67	12.33	67.27	0.74s	46.72	34.79	25.54	0.38s

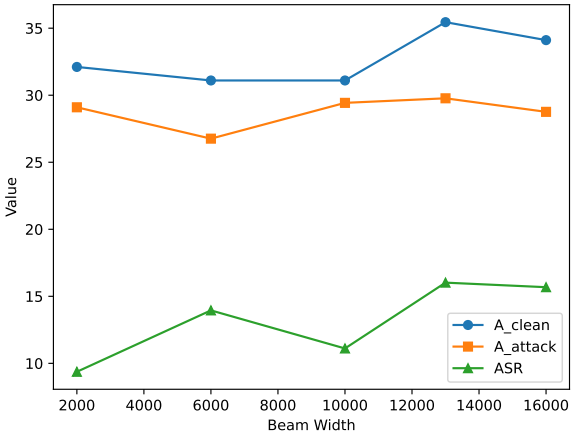
Table 7: Comparison of attack effects of AGBS on different LLMs and datasets (Numerical response test)

Models	SQUAD				Strategy QA				Movie QA			
	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	**ASR ↑**	**AVG**	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	**ASR ↑**	**AVG**	$\mathcal{A}_{\text{clean}}$	$\mathcal{A}_{\text{attack}}$	**ASR ↑**	**AVG**
llama3.1-70B	49.83	44.15	11.40	2.01s	49.67	47.00	5.38	2.03s	79.54	75.29	5.34	2.23s
llama3.3-70B	52.17	49.16	5.77	2.53s	50.33	47.33	5.96	1.49s	78.76	75.29	4.41	1.90s
qwen2.5-0.5B	9.03	8.03	11.07	0.64s	26.67	25.33	5.02	1.09s	20.08	10.46	**47.91**	0.80s
gemma2-2B	19.40	15.97	17.68	0.38s	51.67	36.67	**29.03**	0.26s	62.16	48.59	**21.83**	0.39s

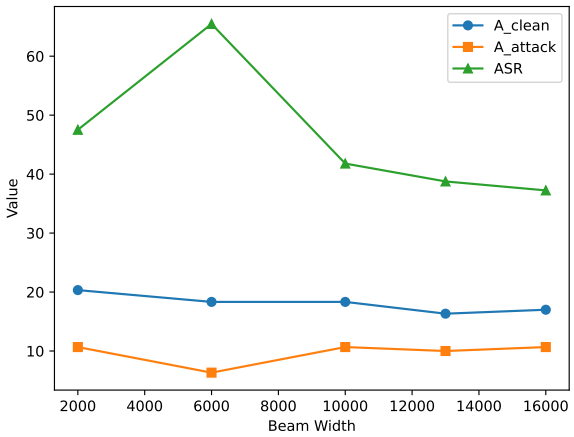
Table 8: Comparison of attack effects of AGBS on different LLMs and datasets (Text response test)



(a) GSM8K on llama3.1-8B

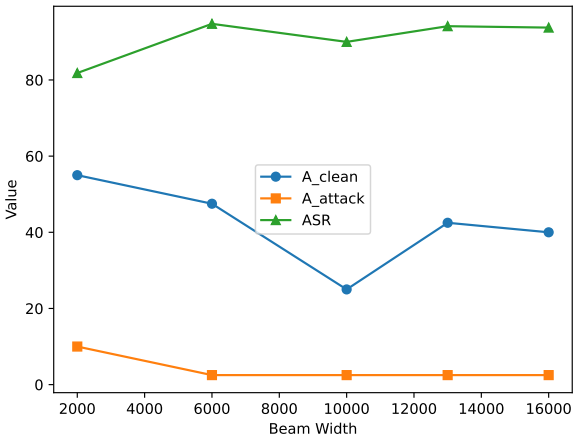


(b) SQUAD on llama3.1-8B

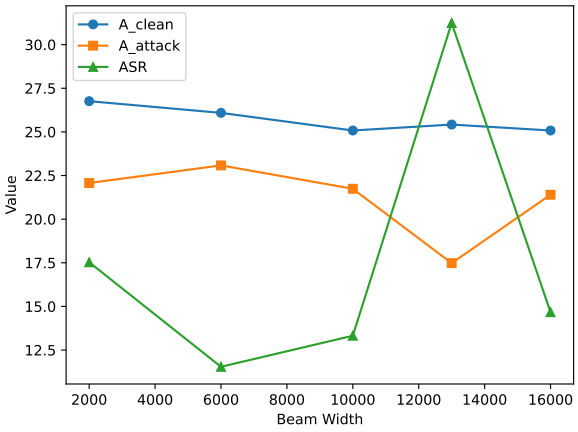


(c) SVAMP on llama3.1-8B

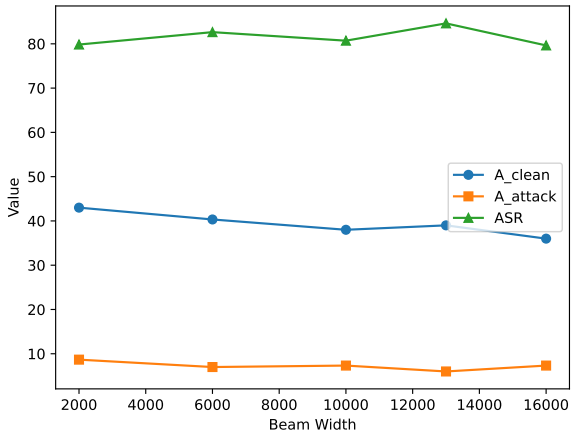
Figure 5: The relationship between the AGBS search scope and the attack success rate. (Part I)



(a) GSM8K on llama3.2-3B

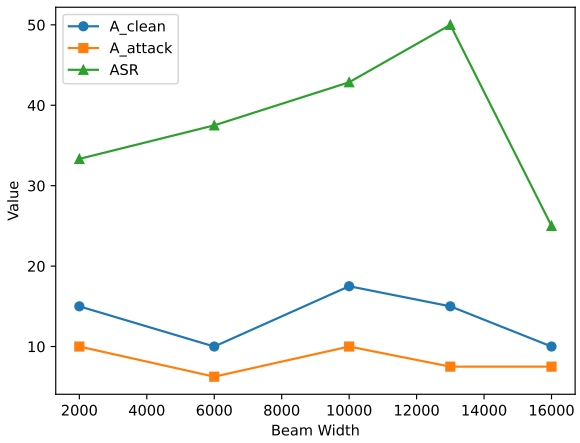


(b) SQUAD on llama3.2-3B

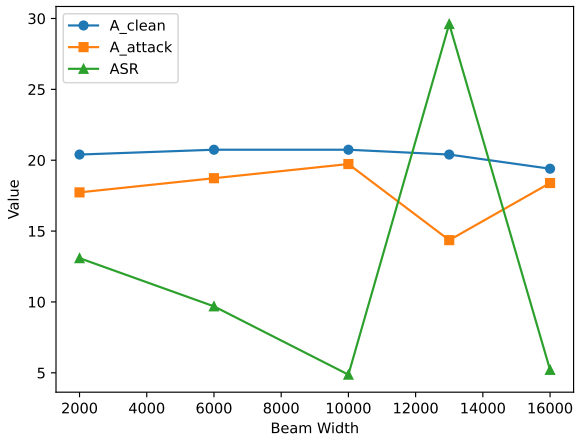


(c) SVAMP on llama3.2-3B

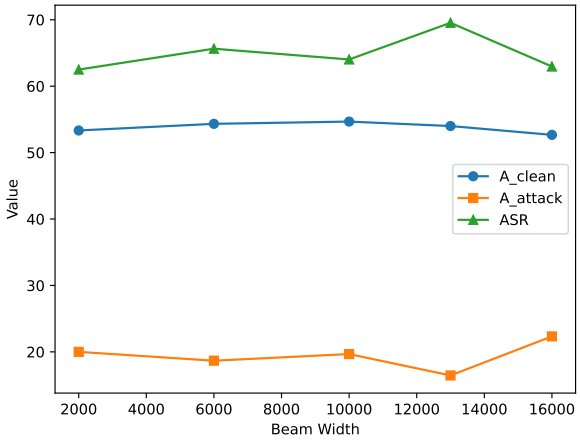
Figure 6: The relationship between the AGBS search scope and the attack success rate. (Part II)



(a) GSM8K on qwen2.5-7B

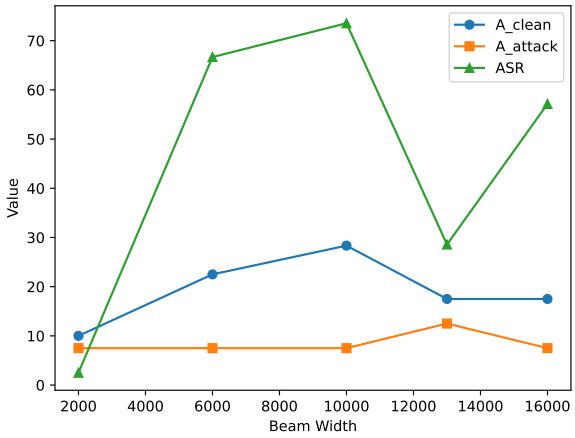


(b) SQUAD on qwen2.5-7B

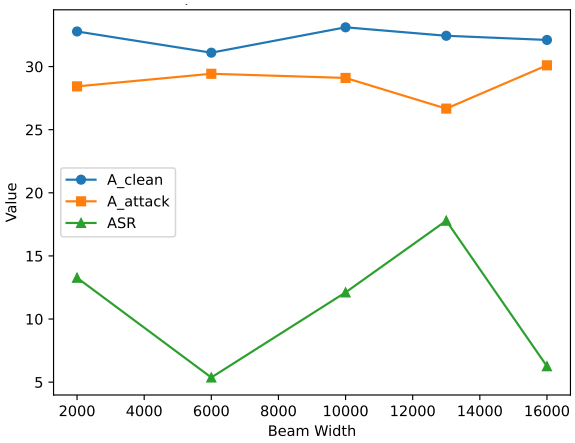


(c) SVAMP on qwen2.5-7B

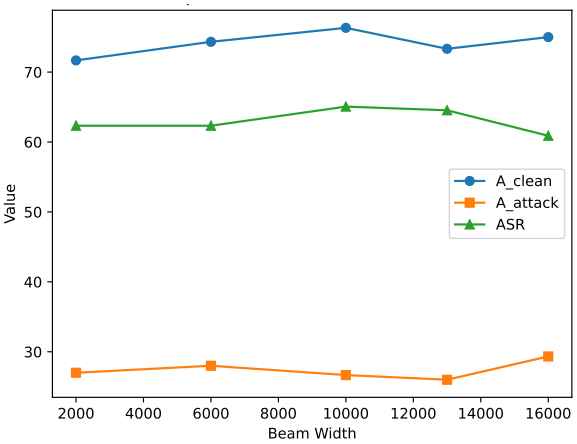
Figure 7: The relationship between the AGBS search scope and the attack success rate. (Part III)



(a) GSM8K on qwen2.5-14B



(b) SQUAD on qwen2.5-14B



(c) SVAMP on qwen2.5-14B

Figure 8: The relationship between the AGBS search scope and the attack success rate. (Part IV)