Auto-Prompt Generation is not robustness: Prompt Optimization Driven by Adversarial Training

Anonymous ACL submission

Abstract

The performance of Large Language Models (LLMs) depends on the quality of prompts and the semantic and structural integrity of the input data. However, existing prompt generation methods primarily focus on well-structured input data, often neglecting the impact of perturbed inputs on prompt effectiveness. To address this limitation, we propose **BATprompt** (***B**y **A**dversarial **T**raining prompt*), a novel method for prompt generation designed to withstand input perturbations (such as typos in the input). Inspired by adversarial training techniques, BATprompt demonstrates strong performance on a variety of perturbed tasks through a two-step process: adversarial perturbation and iterative optimization on unperturbed input via LLM. Unlike conventional adversarial attack methods, BATprompt does not need access to model parameters and gradients. Instead, BATprompt leverages the advanced reasoning, language understanding and self reflection capabilities of LLMs to simulate gradients, guiding the generation of adversarial perturbations and optimizing prompt performance. We evaluate BATprompt on multiple datasets across both language understanding and generation tasks. The results indicate that BATprompt outperforms existing prompt generation methods, delivering superior robustness and performance under diverse perturbation scenarios.

1 Introduction

LLMs can perform a wide range of tasks including text classification, summarization, generation (Abburi et al., 2023; Ge et al., 2024; Laban et al., 2023). With their broad applicability, researchers have begun exploring strategies to enhance the performance of LLMs on these tasks by effectively activating their capabilities. To address this issue, researchers have begun designing more effective prompts to enhance the performance of LLMs. To reduce the burden of manual prompt design,

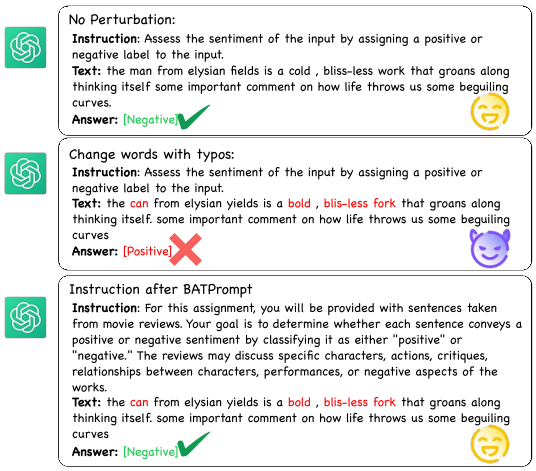

Figure 1: An example of adding a perturbation to the input and its result in language understanding task, where the top one indicates that the prompt gets the correct output under normal input, the middle one indicates that the prompt gets the wrong result under perturbed input, the bottom indicates the BATprompt gets correct result under perturbed input.

LLMs are now widely utilized to optimize prompts and generate improved candidates. Some existing methods include fine-tuning the performance of LLMs (Ouyang et al., 2022; Ziegler et al., 2019), optimizing the prompt by generating CoT (Chain-of Thought) (Shum et al., 2023; Zhang et al., 2022), and optimizing the prompt by guiding the inference ability of LLMs (Wang et al., 2023b; Liu et al., 2024). However, despite the success of these methods, a critical issue is overlooked: in real-world scenarios, task inputs frequently contain errors, such as typos, vague expressions, or inaccuracies. Under such conditions, the prompts generated by current prompt optimization techniques may not exhibit sufficient robustness to handle such imperfections in the input data. For example, in Figure 1, the normal prompt will make LLMs get wrong answer in the classification task when input text has typos.

To enhance prompt robustness for task perfor-

mance, many researchers have turned to data augmentation, adding perturbed text to the training data and training the model on this augmented dataset to produce robust prompts. However, our experimental results indicate that these prompt generation approaches frequently yield prompts that lack robustness against input perturbations (see §5.6). In some tasks, the performance of these prompts even declines, likely due to the excessive diversity of perturbation types introduced during augmentation. We contend that while basic data augmentation methods can be beneficial in certain contexts, they may inadvertently introduce noise that interferes with effective prompt generation and limits generalization to unseen perturbations.

To enable the model to better identify disturbances in the data and enhance the generalization ability of the generated prompts, We introduce a novel prompt optimization method, called **BATprompt**, designed to generate discrete instructions resilient to perturbations. Our approach takes inspiration from adversarial training Madry (2017), which is widely recognized for improving neural network robustness by exposing the model to adversarial attacks during training. We found that the concept of adversarial attacks can also be applied to prompt optimization, making the generated prompts more resilient to perturbed inputs. In the attack phase of our approach, we use gradient updates to deliberately induce adversarial inputs that degrade LLM performance. During the adversarial training stage, we implement two distinct optimization modes tailored to address different types of attacks, iteratively refining the prompts using gradient-based guidance. This ensures that the resulting prompts maintain robustness under diverse conditions. Our main contributions are as follows:

- We propose an adversarial attack-inspired method to perturb the input text without using model gradient, aiming to identify the vulnerabilities of existing prompt optimization techniques to such text perturbations.

- We introduce a novel framework **BATprompt** for generating prompts by **adversarial prompt optimization**, which are resilient to input perturbations. Using an adversarial training framework, we harness gradient-based techniques to drive both the attack and prompt generation processes. The resulting prompts show robust performance in many perturbation types.

- We present a **perturbation dataset**, which includes various tasks such as language understanding and language generation. For each task, we added different levels of perturbation, such as character level, word level and sentence levels.

- Experimental results demonstrate that the proposed method achieves exceptional performance across multiple tasks under various types of perturbations, validating the robustness of BATprompt in addressing these challenges.

2 Relarted Work

2.1 Adversarial Training

Adversarial Attack: Numerous adversarial attack methods (Goodfellow et al., 2014; Madry, 2017; Carlini and Wagner, 2017), have been developed for computer vision and traditional NLP tasks. Among these, FGSM (Goodfellow et al., 2014) and PGD (Madry, 2017) are utilized to add adversarial attack on data. Nowadays, people begin to study adversarial attacks against LLMs, which involve adding slight modifications to natural language input, such as spelling errors, synonym substitutions, character substitutions, or out-of-order, to trick LLMS into making wrong predictions or generation. Zhou et al. (2024c) add attacks on math solving problems in LLMs. Zhu et al. (2023) classifies the hint attacks on LLM into four categories. Specific examples of different classes of attacks are given in Xu et al. (2024). Among them, word-level (Wang et al., 2023a; Zhou et al., 2024b) and sentence-level (Gu et al., 2023; Dong et al., 2021) attacks are more common. (Zhou et al., 2024b) proposed a word-level attack based on the classification task without changing the semantics.

Adversarial Training: FGSM (Goodfellow et al., 2014) proposed the concept of adversarial training, which optimizes the model after adversarial attacks on the data. Miao et al. (2024) proposes a method for an effective adversarial attack on T2M. In LLMs, Raina et al. (2024) presents the unrobustness of Judge-LLM against adversarial attacks, which leads to the inflation of LLM scores. Yao et al. (2023a); Kumar et al. (2023) proposed adversarial attacks on different aspects of LLM and proposed several defense methods, and Yao et al. (2023a) proposed several methods to defend against malicious attacks in prompts. Kumar et al. (2023) proposes an adversarial hallucination attack and proposes a defense strategy (Sheshadri et al., 2024;

Xhonneux et al., 2024; Lin and Zhao, 2024). The adversarial training method is used to improve the robustness of LLM. Sheshadri et al. (2024) proposed a method ReFAT to simulate input attacks and defend against them by refusal feature ablation. Xhonneux et al. (2024) Latant Adversarial Traning is used to improve the robustness of LLM against Jalibreak. Lin and Zhao (2024) proposed a defense technique called LLAMOS to enhance the adversarial robustness of LLM through adversarial attacks of text perturbation.

2.2 Prompt Optimization

The manually designed LLM prompt method sometimes does not make LLM perform better in performing tasks, so the prompt optimization method arises. Li et al. (2023) proposes a fine-tuning method based on context ordering and probability ordering. Beyond fine tuning, reinforcement learning is also a great optimization (Ma et al., 2023; Sun et al., 2023; Yao et al., 2023b), and Ma et al. (2023) proposes Eureka, which generates a reward function that outperforms human experts. Sun et al. (2023) optimizes the arithmetic reasoning ability of large models by Prompt-OIRL. RLprompt (Deng et al., 2022) introduces a reword mechanism to generate better prompts. At present, LLM optimization has become the main method of prompt optimization (Shum et al., 2023; Zhang et al., 2022; Yang et al., 2023). Zhou et al. (2022) proposed APE to use LLM to automatically generate prompts according to input and output. Pryzant et al. (2023) proposed to use LLM self-correction as pseudo-gradient to optimize prompts. Guo et al. (2023) uses genetic algorithm to optimize LLM prompts through heredity and mutation. Jin et al. (2024) search the impact of step length in activating LLMs. Zhou et al. (2024a) use adversarial attack to generate more success jailbreak to deceive LLMs and add some suffix after the input text to resist jailbreak. Additionally, most existing methods generate optimized prompts using clean datasets, overlooking the inevitable perturbations present in natural language inputs. These perturbations can negatively affect the effectiveness of the prompts. Our method addresses this limitation by proposing a strategy that emphasizes maintaining robustness against such perturbed inputs.

Perturbation level	Name.	Explanation	Type
Character	C1	Change words to have typos	P1
	C2	Change Letters.	P1
	C3	Add extraneous characters	P1
Word	W1	Change word to synonyms.	P2
	W2	Delete meaningless words.	P2
	W3	Add neutral words.	P2
Sentence	S1	Add meaningless handle	P1
	S2	Paraphrase the sentence.	P2
	S3	Change the syntactic structure.	P2

Table 1: Explanation of different kinds of perturbation.

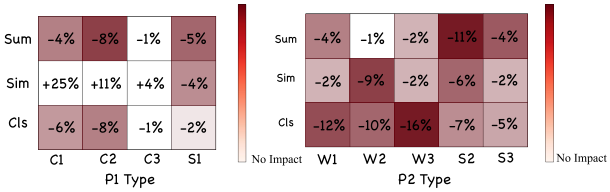

Figure 2: Heat maps showing the magnitude of the impact of each perturbation on different types of tasks, where darker colors indicate a stronger impact of the perturbation on this type of task, and vice versa a lower one. No color indicates no effect or a positive effect

3 Perturbation Types and Task Vulnerability

We adopt a widely used approach following Xu et al. (2023), which form the basis for the perturbations in our experiments (See Table 1). These disturbances are categorized into two groups: *P1* and *P2*. *P1* includes: *C1*, *C2*, *C3*, and *S1*. The perturbations in this category primarily introduce typographical errors and non-sensical strings into the text, without significantly altering the underlying semantic structure of the sentence. *P2* includes: *W1*, *W2*, *W3*, *S2*, and *S3*. Unlike *P1*, these perturbations alter the semantic structure of sentences by modifying their syntactic composition, while maintaining their core meaning. These modifications can result in a bias in how LLMs interpret the input, potentially affecting their understanding.

To assess the weakness of different language tasks on these nine types of perturbations, we added each perturbation to the dataset of different tasks and evaluated the performance of Manual Instructions (Bach et al., 2022) and Natural Instructions (Mishra et al., 2021) on these datasets. The results of the impact of the perturbation on the task are shown in Figure 2, where darker colors indicate greater impact and vice versa. For each task, we select perturbations that have a significant impact as potential candidates and disregard those with a positive or negligible effect. It can be

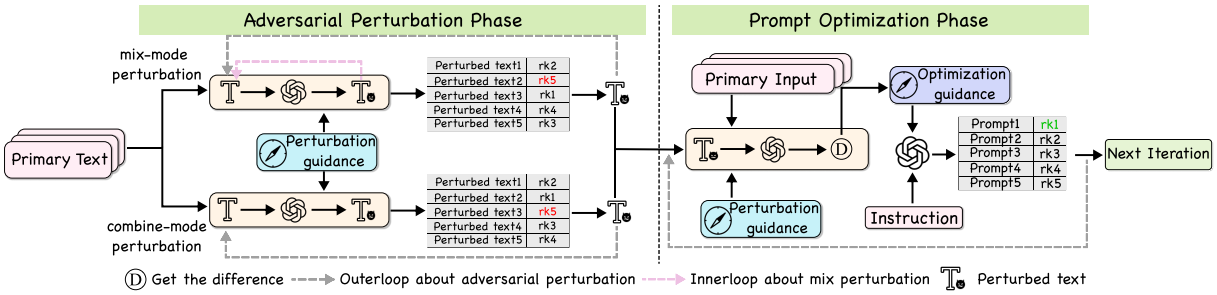

Figure 3: The workflow of an iteration of BATprompt. The Adversarial Attack Phase is used to generate the adversarial samples. The Adversarial Optimization Phase is used to generate optimized prompt. rk means score ranking. For example, rk1 means that the score is ranked first in all outputs.

concluded that perturbations of class *C3* have no negative impact on any task, whereas perturbations of class *P2* consistently affect all tasks to varying degrees. We observe that character-level perturbations in Simplification tasks can positively impact task performance. We believe this is because typo perturbations encourage LLMs to focus more on the core meaning of the input, aligning better with the objective of the simplify task. The specific experimental results and analysis, and the details of the experimental settings are in Appendix A.

4 Methodology

To enhance the robustness of auto-prompt under perturbation, we propose introducing adversarial training into this process. Traditional adversarial training depends on gradients to generate adversarial examples and guide training, but most LLMs operate via black-box APIs where gradients and parameters are inaccessible. Thus, we introduce **BATprompt**, that employs guided information instead of gradients. By leveraging this guidance, LLMs generate new prompts through adversarial attacks and iterative optimizations, enhancing their robustness in context learning.

4.1 Threat Model

Adversarial Perturbation Goal. Like traditional adversarial attacks, this phase aims to induce incorrect outputs from model $\mathcal{M}(\cdot)$ by adding a perturbation to the input. Specifically, given a clean sample x and a perturbed input x', we aim for$\mathcal{M}(x') = y$ $\mathcal{M}(x') = y'$, where y and y' represent the corresponding outputs, and $y' \neq y$. The perturbation must meet the following two conditions:

- Maintain the semantic similarity and the structural similarity to the original text.
- Degrade the performance of the large language model (LLM) on the task, such that the model's performance on the perturbed input is worse than on the clean input.

Algorithm 1 The algorithm flow of **BATprompt**

Require: Initial prompts p, m of adversarial attack gradients g_{adv} and optimization gradient g_{opt}, adversarial attack operation $f_{adv}(\cdot)$, optimization operation $f_{opt}(\cdot)$. adversarial attack constraints $\mathcal{D}(\cdot)$. Sorting function $\mathcal{S}(\cdot)$.

for $t = 1$ to T **do**
  Random Select: Num of texts T^i (i $\in$ N) are randomly selected from the unperturbed dataset.
  for $j = 1$ to num **do**
    Adversarial Attack: $T^{'} \leftarrow f_{adv}(T^i, g^k_{adv})$, where i $\in$ (1, N), k $\in$ (1, m).
    Select Worst Text: $\mathcal{D}(T^{'}, T^i) < \epsilon$, $Min\ \mathcal{S}(T^{'})$.
  end for
  Generate Gradient: Generate optimized gradient g_{opt} through the perturbed text.
  Optimize Prompt: $p \leftarrow f_{opt}(p, g_{opt})$
  Select Best Prompt: $Max\ \mathcal{S}(p)$
end for
Return: Returns the best prompt p_{best} with the highest rating on the perturbed task.

Prompt Optimization Goal. Users typically have typos or semantic ambiguities when typing, especially when dealing with large volumes of perturbed text. Therefore, the goal of prompt optimization is to train robust instructions that can still ensure good task performance, even when confronted with perturbed text inputs.

4.2 Framework of BATprompt

BATprompt follows a similar principle to existing gradient-based adversarial attack methods. It first iteratively adds attacks within the specified range to undisturbed inputs by adjusting them along the gradient direction to maximize disruption. The adversarial samples generated are then used for training based on a defined loss function. The workflow of our method is shown in Figure 3. The details of the specific implementation in LLM are shown

in Figure 11. BATprompt contains many iterations and every iterations include two key components:

- **Adversarial Attack Phase**. BATprompt begins with manually crafted prompts and unperturbed text as the initial input. In section 3, we categorize nine distinct perturbations into two types and design tailored adversarial attack methods for each category. These methods generate adversarial samples that simulate various perturbation scenarios, allowing the subsequent optimization algorithms to build robustness across all types of perturbations. This approach ensures that the model remains effective when faced with diverse perturbations.

- **Adversarial Optimization Phase**. At this stage, we employ an iterative optimization method that incorporates a "gradient" mechanism to guide the LLM in refining the prompts. For each generated prompts, we select the candidate with the best performance on the validation set and retain it for the next iteration.

4.3 Adversarial Attack Phase

At this stage, we employ an adversarial attack method inspired by FGSM (Goodfellow et al., 2014), which uses a fixed gradient direction to enable the model to rapidly generate adversarial samples. In BATprompt, we introduce various types of perturbation-specific guide words as a fixed gradient g, and apply perturbations to n samples x, randomly selected from the original unperturbed dataset. y is the standard output. The performance of the original prompt on the perturbed inputs is evaluated using a loss function $\mathcal{L}_{adv}$. The score is minimized when perturbed. After generating the perturbed sample prompt p, we compute either its Levenshtein distance or semantic similarity $||x' - x|| < \epsilon$, depending on the nature of the perturbation. The perturbations are defined as follows:

$$x' = x + \epsilon \cdot \arg\min \mathcal{L}_{adv}(x + g, y), \quad (1)$$

Specifically, for the two distinct perturbation types, *P1* and *P2*, we draw on the idea of Tramer and Boneh (2019) and design two different modes of adversarial attacks. This ensures that the text used in the adversarial training process encompasses all perturbations within each type. As a result, the generated prompts demonstrate robustness against all attacks corresponding to their respective types. For the *P1* type, we propose a **mix-mode**, which is implemented as follows:

$$x' = x_0 + \mathcal{P}(x_0, g_1) + \mathcal{P}(x_1, g_2) + \ldots + \mathcal{P}(x_{n-1}, g_n), \quad (2)$$

Where $\mathcal{P}(\cdot)$ denotes that the given input x is perturbed under the guide g, $g_n \in$ *P1*, x_n denotes the text generated under each perturbation. The rationale behind this design is that, under the perturbation of the *P1* mode, the superposition of each perturbation has minimal impact on the effect of other perturbations. For example, the input *"I like apple"* becomes *"I lide apple"* after the *C1* perturbation, and then the output becomes *"I lide apple.@jjs"* after the *C3* perturbation. This design enables adversarial attacks to generate results that effectively incorporate all types of perturbations. Thus, the robustness of the generated results against all types of perturbations in *P1* can be ensured, while simplifying the input for the second stage.

In the perturbation of *P2* mode, we propose a **combined-mode** as follows:

$$\mathcal{U}(x') = \mathcal{P}(x, g_1) \cup \mathcal{P}(x, g_2) \cup \ldots \cup \mathcal{P}(x, g_n), \quad (3)$$

where $\mathcal{U}(\cdot)$ denote the set generated after all perturbations. Unlike *P1* type perturbations, which do not interfere with each other's output, different perturbations in *P2* type can affect the results of other perturbations. Therefore, we choose to combine them and input them into the adversarial optimization phase to train the robustness of the generated prompts against all types of perturbations.

4.4 Adversarial Optimization Phase

In the optimization stage, we utilize the adversarial sample x' generated in the first stage. We then analyze the differences between x' and the corresponding original text x, and use these differences to compute the gradient g' when the guide is used as the optimization prompt. The detailed process is as follows:

$$g' = \mathcal{G}(\mathcal{D}(x_0, x_0') \cup \mathcal{D}(x_1, x_1') \cup \ldots \cup \mathcal{D}(x_n, x_n')), \quad (4)$$

Where $\mathcal{D}(\cdot)$ denotes the generating difference and $\mathcal{G}(\cdot)$ denotes the generation of general guidence. In the iterative optimization process following gradient generation, we also use the prompt's score on the task as the loss function. However, unlike the previous stage, we select the prompt with the highest score at each iteration to maximize task performance. To ensure that the final prompt is robust to all perturbations, we calculate the score for each

perturbation across the vulnerabilities of the target task. The specific loss function is as follows:

$$\mathcal{L}_{opt}(p) = \mathbb{E}_{(x',y)\sim D}\big[\mathcal{L}(x_1^{'}, y; p) + \ldots + \mathcal{L}(x_n^{'}, y; p)\big]. \quad (5)$$

Where $\mathcal{L}_{opt}(\cdot)$ represents the optimization loss. $\mathcal{L}(\cdot)$ denotes the loss per class of perturbation. In summary, the formula for each round of prompt optimization is $p' = p + \nabla_p \mathcal{L}_{opt}(p)$. In prompt generation, to expand the range of options, we rewrite the generated prompts at each iteration, increasing the likelihood of discovering better alternatives. During the intermediate optimization iterations, we consistently select the optimal prompt from each round to proceed to the next iteration of the loop.

5 Experiments

5.1 Implementation Details and Baselines

In the experiments, we use GPT-3.5-turbo to do the adversarial training and use **GPT-3.5-turbo**, **GPT-4o-mini** and **Llama2-7b** to test the effectiveness of the instructions generated BATprompt. For perturbations of type *P1*, we select five examples in each iteration, while for perturbations of type *P2*, we select three examples per iteration, considering the number of adversarial examples generated. After five iterations, we choose the prompt with the highest score on the training set and evaluate its performance on the test set. In the evaluation, we compare the prompts generated by BATprompt with the following methods:

- **EvoPrompt** (Guo et al., 2023): Evoprompt uses a genetic algorithm to optimize prompts, its tasks contain both language understanding and language generation tasks.
- **Manual Instructions(MI)**: These instructions are based on existing work that is task-specific guidelines. Including language understanding task Zhang et al. (2023a), text simplification task Zhang et al. (2023b) and summarization task Sanh et al. (2021).
- **Natural Instructions(NI)** (Mishra et al., 2021): This contains manually designed prompts across a diverse range of datasets and tasks.
- **Data Augmentation(DA)**: We use the data augmentation method as baseline, which takes the perturbed text as input data and iteratively optimizes it, to explore whether this method can remain robust to all types of text perturbations.

5.2 Data Generation and Metrics

To evaluate the effectiveness of BATprompt, we construct datasets with various perturbations tailored to different tasks. Using an iterative approach guided by gradients, we introduce perturbations into the datasets. However, different from the traditional method of adversarial attacks, which searches attacks freely in all ranges, our method adds attacks in specific perturbation spaces. For dataset selection, in the **language understanding tasks**, we focus on six datasets to apply and test perturbations. These include sentiment classification datasets: SST-2 (Socher et al., 2013), CR (Hu and Liu, 2004), SST-5 (Socher et al., 2013), and MR (PaNgB, 2005), as well as topic classification datasets: AG's News (Zhang et al., 2015) and TREC (Voorhees and Tice, 2000) with the Prediction accuracy as the score. In **language generation tasks**, we utilize the Asset (Alva-Manchego et al., 2020) for **text simplification**, which includes multiple benchmarks for reference translations. In this task we use SARI (Xu et al., 2016) as the metrics which is an n-gram-based scoring system extensively utilized for text editing tasks. For the **text summarization task**, we use the XSum (Narayan et al., 2018), which consists of concise summaries generated from longer texts. In this task, we use Rouge-1, Rouge-2, Rouge-L as metrics (Lin, 2004), which widely used to evaluate the quality of generated text tasks. They focus on the number of n-grams with the same outcome and the overlap of the longest common subsequence.

5.3 Effectiveness on Language Generation

In this section, we utilize the GPT-3.5-turbo model to evaluate the generated prompts. For the text summarization task, we assessed the quality of the generated prompt guidance using three metrics, ROUGE-1, ROUGE-2, and ROUGE-L, calculated on the test set. For text simplification tasks, the quality of the generated prompts were by calculating the SARI score of the prompt guided result.

For the **text summarization** task, the results for the perturbations *P1* and *P2* are presented in Table 2 and Table 3. Compared to previous prompts and their generation methods, the prompts generated by BATprompt demonstrate a significant performance improvement on the perturbed datasets. Furthermore, BATprompt exhibits strong robustness across all types of perturbations, achieving superior Rouge-1, Rouge-2, and Rouge-L scores

Methods	C2 perturbation			C1 perturbation			S1 perturbation		
	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
DA	11.37	2.65	9.78	11.90	2.48	9.93	10.90	2.38	9.11
MI	14.55	2.80	12.31	15.27	3.12	12.80	15.13	3.19	12.65
EvoPrompt	17.62	3.16	14.96	17.24	3.03	14.61	17.71	3.24	15.34
BATprompt*	18.31	3.08	15.50	18.65	2.91	15.03	17.65	2.81	15.58
BATprompt	**21.10**	**4.36**	**16.85**	**21.17**	**4.37**	**16.34**	**21.47**	**4.59**	**16.78**

Table 2: The Rouge-1(↑) score, Rouge-2(↑) score and Rouge-L(↑) score obtained by the prompts generated by the three methods for the task where the text summarization task is weak under the *P1* class perturbation.

Methods	W3 perturbation			W1 perturbation			S2 perturbation			S3 perturbation		
	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L	Rouge-1	Rouge-2	Rouge-L
DA	12.18	2.87	10.25	11.85	2.92	10.46	12.62	2.66	10.60	12.12	2.91	10.25
MI	15.55	3.15	13.01	15.09	3.19	12.88	14.06	3.03	11.69	15.29	2.74	13.13
EvoPrompt	17.54	3.09	14.93	17.20	3.03	14.85	16.97	2.67	14.48	17.03	2.93	14.49
BATprompt*	18.42	3.10	15.82	18.64	3.39	15.98	17.92	2.91	15.28	17.90	3.06	15.46
BATprompt	**22.04**	**4.91**	**17.29**	**22.09**	**4.71**	**16.70**	**21.51**	**4.76**	**16.45**	**21.97**	**4.55**	**16.78**

Table 3: The Rouge-1(↑) score, Rouge-2(↑) score and Rouge-L(↑) score obtained by the prompts generated by the three methods for the task where the text summarization task is weak under the *P2* class perturbation.

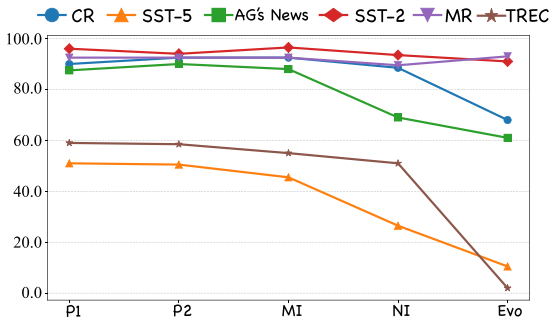

Figure 4: The performance of the prompts generated by BATprompt on undisturbed datasets of language understanding task.

compared to other prompts in all aspects. Notably, BATprompt outperforms the second-best method by an impressive 23% under C2 perturbations.

For the **text simplification** task, the SARI values achieved by BATprompt under P1 and P2 perturbations are shown in the Table 4. Similar to the text summarization task, BATprompt demonstrates strong robustness across all perturbations. Its SARI values consistently have better performance than those of the second-best method, highlighting its effectiveness and resilience.

5.4 Effectiveness on Language Understanding

In this part, we use GPT-3.5-turbo to evaluate the accuracy of its judgments on six datasets, which is recorded as its score. For each dataset, we train the model on perturbations from class P1 and perturbations from class P2. We then test the optimized prompts across different types of perturbations and compute the average performance over them. The results for the perturbations from class P1 and class P2 are presented in the Table 5. Specific experimental indicators are given in Figure 12.

The results demonstrate that the prompts generated by BATprompt outperform existing prompts across the six text understanding datasets, exhibiting notable stability against the seven perturbations to which BATprompt itself is not inherently robust. Its performance is particularly impressive. Notably, BATprompt achieves a 3% improvement on the TREC dataset for P1 perturbations and a remarkable 12% improvement for T2 perturbations.

5.5 Model Transferbility and Perturbation Transferbility

We also use **GPT-4o-mini** and **Llama2-7b** to test the effectiveness of our method. The results are showed in the subsection D.3. To further assess the robustness of our prompts against unseen perturbations, we evaluated them on perturbations that were not included in the training set. As shown in subsection D.4, although their performance is marginally lower than that of prompts specifically optimized for those perturbations, our approach consistently outperforms all baseline methods.

Methods	S1 perturbation	W3 perturbation	W1 perturbation	S2 perturbation	S3 perturbation	W1 perturbation
DA	44.87	42.53	44.04	44.89	44.66	45.28
MI	36.67	34.77	37.43	37.18	35.93	37.23
EvoPrompt	44.61	44.89	45.39	47.38	46.07	49.09
BATprompt*	45.28	43.45	44.53	46.29	46.47	47.90
BATprompt	**45.79**	**45.11**	**49.50**	**47.74**	**47.35**	**49.55**

Table 4: The SARI(↑) score obtained by the prompts generated by the two methods for the task where the text simplification task.

Type	Methods	CR	SST-2	MR	SST-5	TREC	AG'news	Avg.
P1	DA	88.5	89.8	85.5	44.0	43.8	86.3	73.0
	MI	89.3	88.8	85.5	45.5	50.3	87.0	74.4
	NI	83.3	80.3	80.0	23.3	45.3	65.3	62.9
	EvoPrompt	63.8	80.5	78.1	10.5	1.5	67.3	51.0
	BATprompt*	85.5	86.0	81.0	45.3	49.8	85.5	72.1
	BATprompt	**89.8**	**90.0**	**86.3**	**46.0**	**52.3**	**88.5**	**75.4**
P2	DA	85.1	85.5	81.2	40.1	50.3	86.7	71.5
	MI	85.0	85.5	82.0	43.6	46.2	86.2	71.4
	NI	79.1	80.0	77.0	20.0	43.4	61.8	60.2
	EvoPrompt	62.8	78.1	80.6	8.6	1.5	66.8	49.7
	BATprompt*	80.6	81.5	78.1	40.9	49.9	84.3	69.2
	BATprompt	**85.7**	**86.3**	**82.8**	**44.9**	**51.6**	**87.7**	**73.2**

Table 5: Average score(↑) of the prompts from different method on six language understanding datasets. The table in the upper half is a perturbation of type P1, and the table in the lower half is a perturbation of type P2.

5.6 Data Augmentation

To evaluate the limitations of the data augmentation method, we exclude the use of BATprompt during testing. Instead, we treat all the perturbed data as the training set and allow the LLM to generate prompt based on this training set. The results for three different tasks are presented in the *DA* column across the Table 4, Table 2, Table 3 and Table 5. As shown, the prompts generated using the data augmentation method lack robustness to perturbations across all three tasks. Furthermore, their performance is even worse than unoptimized prompts when tested on handwritten perturbations. We attribute this degradation to the excessive diversity of perturbations, which hinders the LLM's ability to focus accurately on the perturbations, leading to a decline in performance.

5.7 Albation Study

To evaluate the effectiveness of adversarial training strategies for prompt optimization, we removed the adversarial training part from BATprompt and used only a simple iterative optimization method. This allowed us to isolate and assess the impact of adversarial training. In the Table 2, Table 3, Table 4 and Table 5, *BATprompt** represents the specific scores of the prompts generated using only the iterative optimization method for both the text comprehension and text generation tasks. The experimental results demonstrate that, across all tasks, prompts generated purely through iterative optimization perform worse than those generated by BATprompt across all types of perturbations. This proves the effectiveness of our adversarial training strategy. In addition, we also explore the influence of the number of training iterations on the quality of the generated prompt, and the experimental results show that the results are slightly higher when the number of iterations is 5 than other iterations. The specific result is in subsection E.1.

6 Conclusion

In this paper, we introduce **BATprompt**, a novel approach designed to optimize prompts' performance on perturbed datasets by combining prompt optimization algorithms with adversarial training. By leveraging LLMs to simulate gradients, BATprompt enables adversarial attacks and iterative optimization. Our extensive experiments demonstrate the superiority of BATprompt across multiple tasks and various types of perturbations. Compared to existing prompt optimization methods, BATprompt shows significant improvements in handling fragile perturbations and good generalization.

7 Limitation

We employed a training approach inspired by FGSM, which, while effective, could be enhanced by more advanced or specialized methods suited to particular perturbation types or task requirements. Additionally, although our work covers a range of widely used tasks and datasets, future exploration may involve expanding into more specialized or innovative application domains. Our reliance on black-box model access poses another constraint, as deeper access to model-internal gradients could potentially offer richer interpretability and more granular refinements during adversarial training. Pursuing open-source architectures or collaboration with model providers may therefore create new opportunities to improve both robustness and understanding of how perturbations interact with large language models.

References

Harika Abburi, Michael Suesserman, Nirmala Pudota, Balaji Veeramani, Edward Bowen, and Sanmitra Bhattacharya. 2023. Generative ai text classification using ensemble llm approaches. *arXiv preprint arXiv:2309.07755*.

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. 2020. Asset: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. *arXiv preprint arXiv:2005.00481*.

Stephen H Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, et al. 2022. Promptsource: An integrated development environment and repository for natural language prompts. *arXiv preprint arXiv:2202.01279*.

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.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*.

Jialiang Dong, Zhitao Guan, Longfei Wu, Xiaojiang Du, and Mohsen Guizani. 2021. A sentence-level text adversarial attack algorithm against iiot based smart grid. *Computer Networks*, 190:107956.

Yingqiang Ge, Wenyue Hua, Kai Mei, Juntao Tan, Shuyuan Xu, Zelong Li, Yongfeng Zhang, et al. 2024. Openagi: When llm meets domain experts. *Advances in Neural Information Processing Systems*, 36.

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

Kang Gu, Ehsanul Kabir, Neha Ramsurrun, Soroush Vosoughi, and Shagufta Mehnaz. 2023. Towards sentence level inference attack against pre-trained language models. *Proceedings on Privacy Enhancing Technologies*.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2023. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024. The impact of reasoning step length on large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1830–1842, Bangkok, Thailand. Association for Computational Linguistics.

Aounon Kumar, Chirag Agarwal, Suraj Srinivas, Aaron Jiaxun Li, Soheil Feizi, and Himabindu Lakkaraju. 2023. Certifying llm safety against adversarial prompting. *arXiv preprint arXiv:2309.02705*.

Philippe Laban, Wojciech Kryściński, Divyansh Agarwal, Alexander Richard Fabbri, Caiming Xiong, Shafiq Joty, and Chien-Sheng Wu. 2023. Summedits: measuring llm ability at factual reasoning through the lens of summarization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9662–9676.

Haoran Li, Yiran Liu, Xingxing Zhang, Wei Lu, and Furu Wei. 2023. Tuna: Instruction tuning using feedback from large language models. *arXiv preprint arXiv:2310.13385*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Guang Lin and Qibin Zhao. 2024. Large language model sentinel: Advancing adversarial robustness by llm agent. *arXiv preprint arXiv:2405.20770*.

Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. 2024. Large language models as evolutionary optimizers. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*.

Aleksander Madry. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Honglei Miao, Fan Ma, Ruijie Quan, Kun Zhan, and Yi Yang. 2024. Autonomous llm-enhanced adversarial attack for text-to-motion. *arXiv preprint arXiv:2408.00352*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

L PaNgB. 2005. Exploitingclassrelationshipsforsentimentcate gorizationwithrespectratingsales. *IN: ProceedingsofACL r05*.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*.

Vyas Raina, Adian Liusie, and Mark Gales. 2024. Is llm-as-a-judge robust? investigating universal adversarial attacks on zero-shot llm assessment. *arXiv preprint arXiv:2402.14016*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Abhay Sheshadri, Aidan Ewart, Phillip Guo, Aengus Lynch, Cindy Wu, Vivek Hebbar, Henry Sleight, Asa Cooper Stickland, Ethan Perez, Dylan Hadfield-Menell, et al. 2024. Targeted latent adversarial training improves robustness to persistent harmful behaviors in llms. *arXiv preprint arXiv:2407.15549*.

KaShun Shum, Shizhe Diao, and Tong Zhang. 2023. Automatic prompt augmentation and selection with chain-of-thought from labeled data. *arXiv preprint arXiv:2302.12822*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Hao Sun, Alihan Hüyük, and Mihaela van der Schaar. 2023. Query-dependent prompt evaluation and optimization with offline inverse rl. In *The Twelfth International Conference on Learning Representations*.

Florian Tramer and Dan Boneh. 2019. Adversarial training and robustness for multiple perturbations. *Advances in neural information processing systems*, 32.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207.

Haoyu Wang, Guozheng Ma, Cong Yu, Ning Gui, Linrui Zhang, Zhiqi Huang, Suwei Ma, Yongzhe Chang, Sen Zhang, Li Shen, et al. 2023a. Are large language models really robust to word-level perturbations? *arXiv preprint arXiv:2309.11166*.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. 2023b. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*.

Sophie Xhonneux, Alessandro Sordoni, Stephan Günnemann, Gauthier Gidel, and Leo Schwinn. 2024. Efficient adversarial training in llms with continuous attacks. *arXiv preprint arXiv:2405.15589*.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Xilie Xu, Keyi Kong, Ning Liu, Lizhen Cui, Di Wang, Jingfeng Zhang, and Mohan Kankanhalli. 2023. An llm can fool itself: A prompt-based adversarial attack. *arXiv preprint arXiv:2310.13345*.

Xilie Xu, Keyi Kong, Ning Liu, Lizhen Cui, Di Wang, Jingfeng Zhang, and Mohan Kankanhalli. 2024. An LLM can fool itself: A prompt-based adversarial attack. In *The Twelfth International Conference on Learning Representations*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.

Jia-Yu Yao, Kun-Peng Ning, Zhen-Hui Liu, Mu-Nan Ning, Yu-Yang Liu, and Li Yuan. 2023a. Llm lies: Hallucinations are not bugs, but features as adversarial examples. *arXiv preprint arXiv:2310.01469*.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, et al. 2023b. Retroformer: Retrospective large language agents with policy gradient optimization. *arXiv preprint arXiv:2308.02151*.

Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Jialin Pan, and Lidong Bing. 2023a. Sentiment analysis in the era of large language models: A reality check. arxiv. *arXiv preprint arXiv:2305.15005*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Yue Zhang, Leyang Cui, Deng Cai, Xinting Huang, Tao Fang, and Wei Bi. 2023b. Multi-task instruction tuning of llama for specific scenarios: A preliminary study on writing assistance. *arXiv preprint arXiv:2305.13225*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Andy Zhou, Bo Li, and Haohan Wang. 2024a. Robust prompt optimization for defending language models against jailbreaking attacks. *arXiv preprint arXiv:2401.17263*.

Huichi Zhou, Zhaoyang Wang, Hongtao Wang, Dongping Chen, Wenhan Mu, and Fangyuan Zhang. 2024b. Evaluating the validity of word-level adversarial attacks with large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 4902–4922.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.

Zihao Zhou, Qiufeng Wang, Mingyu Jin, Jie Yao, Jianan Ye, Wei Liu, Wei Wang, Xiaowei Huang, and Kaizhu Huang. 2024c. Mathattack: Attacking large language models towards math solving ability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19750–19758.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, et al. 2023. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts. *arXiv preprint arXiv:2306.04528*.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Weakness of Tasks

We used the same dataset as in section 5, and GPT-3.5-turbo was used as the test model. When constructing the perturbed data set, we also adopt the data generation way in section 5.

A.1 Language Understanding

The Figure 5 illustrates the performance scores for a text classification task under various types of perturbations. Six datasets were analyzed, with each dataset employing two system prompts: manual instruction and natural instruction to guide the tasks. The scores for each task were averaged to determine the specific impact of each type of perturbation under different prompts. The results reveal that under manual instruction, the S1 and C3 perturbations have minimal impact on the final task scores. Consequently, these two types of perturbations were excluded from further consideration during selection.

A.2 Language Generation

The detailed results for text summarization and text simplification tasks are presented in the Figure 6 and Figure 7. Using manual instruction, we computed the Rouge-1/2/L scores and SARI values for the two datasets, XSum and ASSET, under various perturbations. For the text summarization task, the results indicate that the perturbations C3 and W2 do not significantly affect the Rouge-2 and Rouge-L scores. Therefore, these perturbations were not identified as weaknesses in the text summarization dataset. Similarly, for the text simplification task, perturbations C1, C2, and C3 have no negative impact on the SARI values, and thus, they were not considered weaknesses for the text simplification dataset.

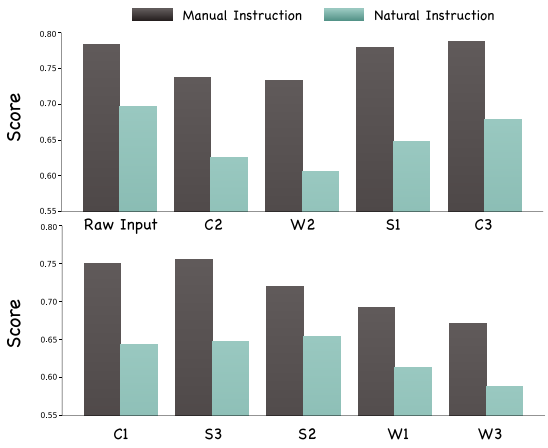

Figure 5: An example of adding a perturbation to an input, where the top half indicates that the prompt gets the correct output under normal input, and the bottom half indicates that the prompt gets the wrong result under perturbed input

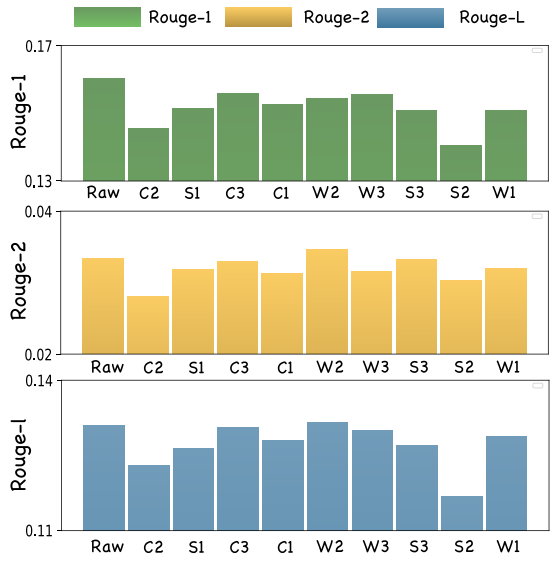

Figure 6: An example of adding a perturbation to an input, where the top half indicates that the prompt gets the correct output under normal input, and the bottom half indicates that the prompt gets the wrong result under perturbed input

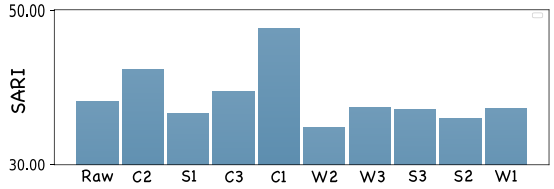

Figure 7: An example of adding a perturbation to an input, where the top half indicates that the prompt gets the correct output under normal input, and the bottom half indicates that the prompt gets the wrong result under perturbed input

B Adversarial Attack Results

In our adversarial attack process, we use both semantic and structural similarity of the text before and after the attack as constraints. This ensures that, while the adversarial attack negatively impacts the target task, the resulting text maintains a high degree of similarity to the original. Specifically, for P1-type attacks, we calculate Levenshtein distance, while for P2-type attacks, we compute semantic similarity. The Table 6 the similarity between the attacked text and the original text. From the table, we can observe that for long text inputs, such as those in the XSum dataset, the Levenshtein distance and semantic similarity between the original and perturbed text remain above 98% for both types of perturbations (P1 and P2). In contrast, for shorter text inputs, such as those in the Asset dataset and the six language understanding tasks,

Datasets	Levenshtein distance	Semantic similarity
XSum	98.42	98.31
Asset	89.55	91.17
CR	93.38	85.92
SST-5	91.22	79.75
AG'News	96.63	83.91
MR	92.42	82.33
TREC	91.41	81.44
SST-2	91.96	80.86

Table 6: The averagesemantic similarity and Levenshtein distance before and after attack in 3 tasks, 8 datasets

Add Perturbation

In the <Task Name> task, the original text is <Primary Input>

You need to generate a new text by the guidance of the following

1. <Perturbation Guidance>
2. Make the LLM get the bad task performance
3. Keeping the meaning of the original sentences.

OutPut: Perturbed Input

Figure 8: The template of generating the optimization gradient.

even small changes can significantly impact the similarity measures. However, our experiments show that the Levenshtein distance similarity for P1 perturbed data remains above 90%, while the semantic similarity for P2 perturbed data stays above 80%, except for the SST-5. This indicates that our attack effectively preserves the essential information of the original sentence while introducing controlled perturbations.

C Experiment Settings

C.1 Datasets

The Table 7 shows the statistics of datasets we made for language understanding, text simplification, and text summarization tasks under different perturbations. Each dataset contains multiple sub-datasets, each of which is a specific class of perturbation for which the dataset feels weak. For instance, Xsum dataset contains 7 sub-datasets including Xsum under C2, C1, S1, W3, S3, S2, W1 perturbations.

C.2 Hpyper Parameters

Our BATprompt algorithm is based on GPT-3.5-turbo for generation, with a total of 5 adversarial training iterations. During the adversarial attack

Dataset	Task	|Train|	|Test|
SST-2	language understanding {positive, negative}	1000	200
CR	language understanding {positive, negative}	1000	200
MR	language understanding {positive, negative}	1000	200
SST-5	language understanding {terrible, bad, okay, good, great}	1000	200
AG's News	language understanding {World, Sports, Business, Tech}	1000	200
TREC	language understanding {Description, Entity, Expression, Human, Location, Number}	1000	200
ASSET	Text Simplification	787	200
XSUM	Text Summarization	520	200

Table 7: Statistics of our generating datasets for language understanding task and language generation task used in this work.

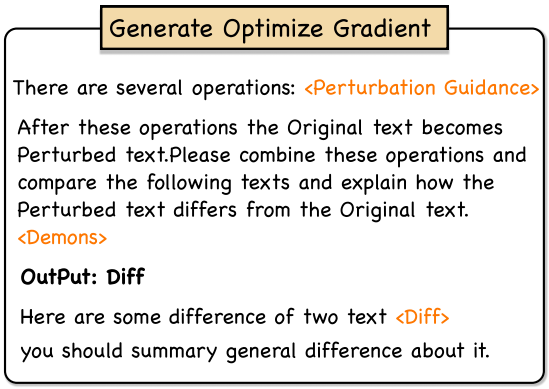

Figure 9: The template of generating the optimization gradient.

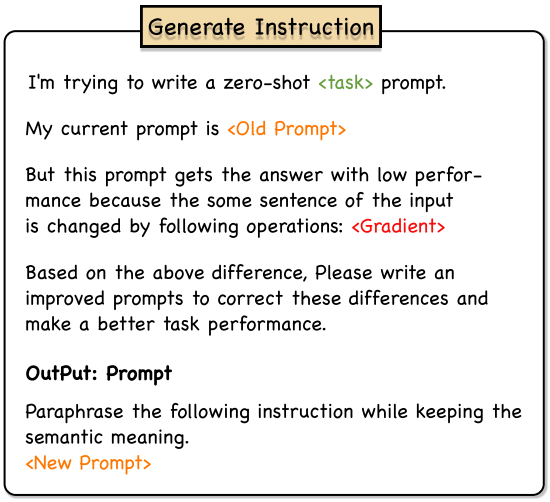

Figure 10: The template of using gradient to generate new instructions and paraphrase them.

phase, we set the number of iterative attacks to 3. In the combined adversarial attack and prompt optimization stages, we configured GPT-3.5-turbo with a Top-p value of 0.95 and a temperature of 1 to ensure both the robustness of the adversarial attack and the diversity of the generated outputs. For the testing phase, we set Top-p to 1 and temperature to 0, ensuring that the model produces consistent, fixed outputs.

C.3 Template

In this section, we give a complete template for the perturbation adding phase. Figure 8, the prompt optimization phase. Figure 9 provides a detailed illustration of the gradients generated during the optimization phase. Figure 10 offers a comprehensive explanation of the prompt generation process, where gradients guide the LLM, and highlights how prompt richness is enhanced through the rewriting process.

Here is the Implementation details of BATprompt.

D Additional Results

D.1 Cost Analysis

In this section, we evaluate the overhead of BATprompt in generating prompts. The primary overhead arises from the evaluation and generation processes during adversarial training. The total cost is represented by the following relation: $N \times (A+O)$, where N is the number of iterations, A represents the cost of the adversarial attack phase, and O represents the cost of the optimization phase. When calling the LLM API, the cost is primarily determined by the number of tokens processed, including both input and output tokens. To estimate the cost of our method, we calculate the number of tokens required for executing tasks on three different datasets (XSum, Asset, and SST-5) across three types of tasks. This provides an understanding of the computational overhead associated with our approach. The results are in Table 8 From the results in the table, it can be observed that for relatively

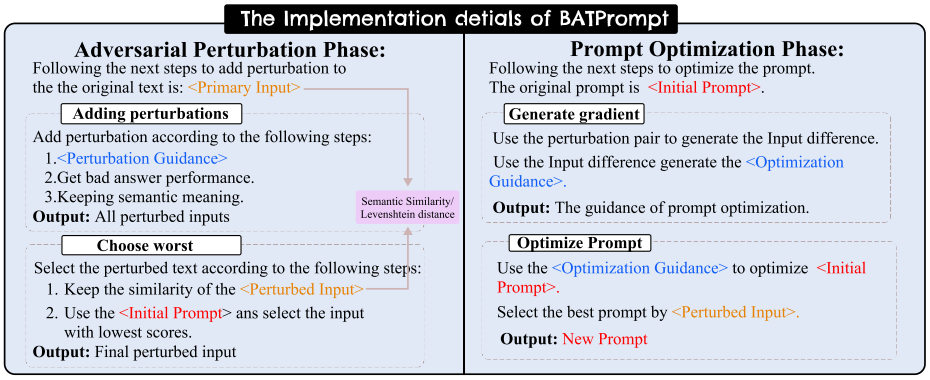

Figure 11: BATprompt is based on the details of the LLM implementation, with the top half representing the adversarial attack phase and the bottom half representing the optimization phase, where orange represents the input text, blue represents the gradient, red represents the prompts

Datasets	Perturb Phase	Optimize Phase	Total
SST-5(P1)	0.0064M	0.0194M	0.0258M
SST-5(P2)	0.0025M	0.0369M	0.0394M
Asset(P1)	0.0289M	0.1604M	0.1893M
Asset(P2)	0.0554M	0.8292M	0.8846M
XSum(P1)	0.2793M	0.5692M	0.8485M
XSum(P2)	0.2006M	0.8395M	1.0401M

Table 8: The cost of BATprompt in three dataset under two kinds of perturbations

simple tasks (such as language understanding), the token consumption of the BATprompt algorithm is only 0.0258M. In contrast, for more complex tasks (such as text summarization), the token consumption remains at a low 0.0258M. After completing one round of BATprompt, the total token consumption is just 1.04M. This demonstrates that, once trained, BATprompt does not incur a large number of tokens. In comparison, Evoprompt consumes at least 4.20M tokens when converging on the SST-5 dataset, which has relatively simple input, significantly exceeding the token consumption of our method on XSum. This indicates that BATprompt offers substantial cost savings.

D.2 Language Understanding Task

In this section, we provide additional details of the data used in the experimental evaluation for the language understanding tasks. Results are in Figure 13. We present the performance indicators of the prompts generated by BATprompt, alongside those obtained using other methods, across seven different perturbations on six datasets. The results show that BATprompt outperforms the other methods in most perturbation scenarios. Notably, in multi-label classification tasks such as SST-5 and TREC, BATprompt demonstrates stronger robustness to perturbations. This may be because binary classification tasks, with fewer labels and simpler task structures, are less affected by perturbations. In contrast, multi-label classification tasks are more complex, with a larger number of target categories, making them more vulnerable to perturbations. As a result, the prompts generated by BATprompt perform better in these more challenging scenarios.

D.3 Model Transferbility

Effectiveness on GPT-4o-mini. To demonstrate the universality of our method in different LLMs, we select the text summarization task and use **GPT-4o-mini** as the LLM backbone. The prompts generated by BATprompt, along with those generated by several other methods, are evaluated under different perturbations using Rouge-1, Rouge-2, and Rouge-L metrics. The results obtained are averaged, shown in the Table 9. According to the experimental results, on different models, the prompts generated by BATprompt are still robust to different types of perturbations, and perform significantly better than other methods. In addition, we also tested the text simplification task and language understanding task on GPT-4o-mini. For the langugae-understanding task, We selected binary classification problem (CR) and multi-classification problem (SST-5) in sentiment classification datasets: as well as topic classification datasets. The specific experimental results are shown in the Table 14 and Table 12. In summary, our method also performs well on GPT-4o-mini.

We plot the specific Rouge scores for each perturbation in the figure. The results show that, across all perturbations, our method demonstrates full robustness, outperforming the other prompt gener-

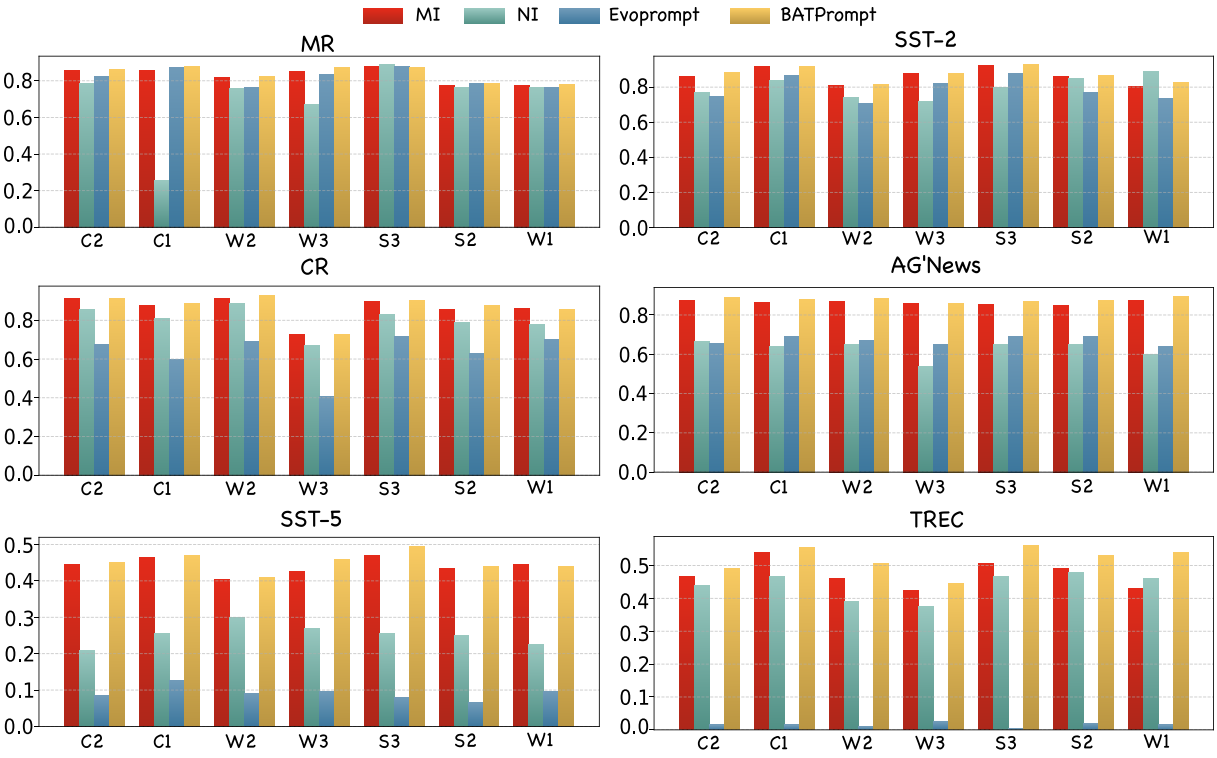

Figure 12: In language understanding, specific indicators of 6 datasets on different 7 kinds of perturbations

Type	Method	Rouge-1	Rouge-2	Rouge-L
P1	MI	16.06	2.86	13.02
	EvoPrompt	16.88	2.62	14.44
	BATprompt	**19.72**	**3.55**	**14.82**
P2	MI	16.14	2.82	12.98
	EvoPrompt	16.78	2.63	14.22
	BATprompt	**19.26**	**3.71**	**15.12**

Table 9: The average of Rouge-1(↑), Rouge-2(↑) and Rouge-L(↑) of prompts generated by different methods on the text summarization task on GPT-4o-mini.

Type	Method	Rouge-1	Rouge-2	Rouge-L
P1	MI	16.55	3.05	13.71
	EvoPrompt	14.82	2.62	12.78
	BATprompt	**17.62**	**3.53**	**14.25**
P2	MI	16.35	2.85	13.88
	EvoPrompt	14.78	2.49	12.91
	BATprompt	**18.98**	**3.74**	**15.17**

Table 10: The average of Rouge-1(↑), Rouge-2(↑) and Rouge-L(↑) of prompts generated by different methods on the text summarization task on Llama2-7b.

ation methods by a significant margin. Table 12 and Table 14 present the specific values of the prompt generated by BATprompt for the text simplification and language understanding tasks, respectively. In the text simplification task, BATprompt demonstrates optimal performance under various perturbations, with notable improvements in task scores, especially on GPT-4o-mini. Similarly, in the language understanding task, we selected several representative datasets, and the results show that our method retains robustness to perturbations across multiple datasets.

Effectiveness on Llama2-7b. We also transferred the instructions generated by BATprompt to a white-box model to assess the effectiveness of our prompts. For the text summarization task, we used **Llama2-7b** to compute Rouge-1, Rouge-2, and Rouge-L scores. To evaluate the robustness of our adversarial training prompts under various perturbations, we present the results in the Table 10. Our method still has optimal results. Further details on the text simplification and language understanding tasks can be found in the subsection D.3.

There are also additional details on the experiments conducted using Llama2-7b as the backbone for the text summarization task, as well as the experimental data for text simplification and language understanding. The results for the text summarization task are presented in the Figure 14, where it is evident that BATprompt consistently outperforms baseline methods across all data items, as measured by Rouge-1, Rouge-2, and Rouge-L scores.

The SARI scores for the text simplification task on the Asset dataset are presented in the table. Sim-

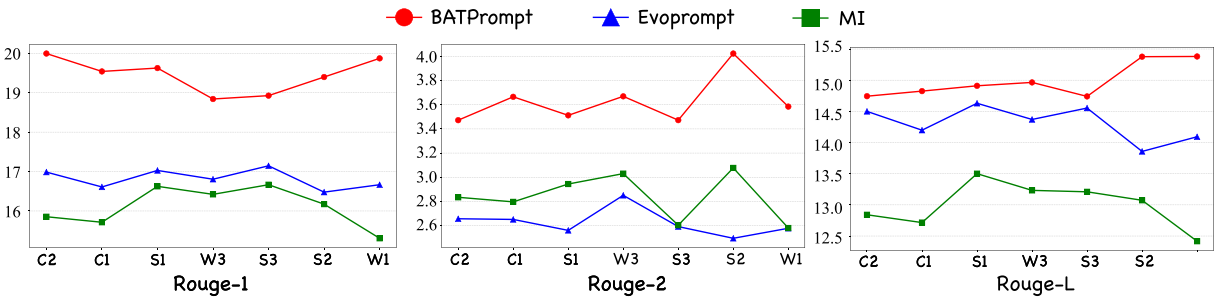

Figure 13: In the text summarization task, using GPT-4o-mini as backbone, the prompts produced by BATprompt and Rouge-1(↑), Rouge-2(↑), Rouge-L(↑) of the remaining methods on different disturbances.

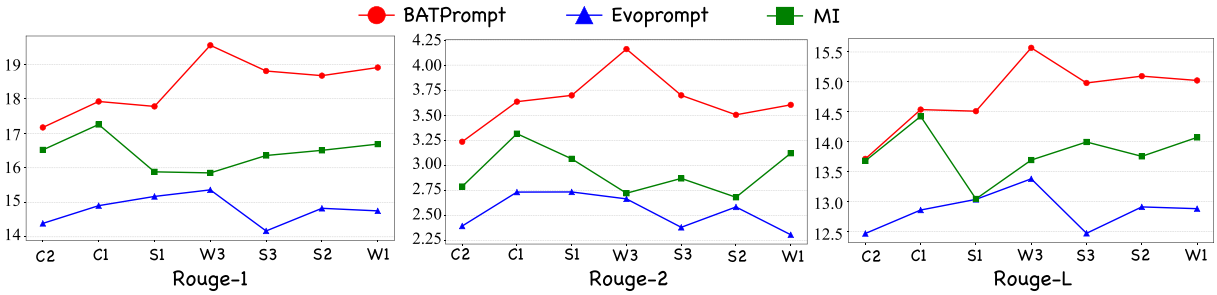

Figure 14: In the text summarization task, using Llama2-7b as backbone, the prompts produced by BATprompt and Rouge-1, Rouge-2, Rouge-L of the remaining methods on different disturbances.

Methods	S1	W3	W1	S2	S3	W1
MI	41.00	37.58	39.65	41.74	39.58	41.87
EvoPrompt	35.25	28.80	30.12	32.12	32.64	31.62
BATprompt	**41.60**	**40.54**	**40.13**	**42.82**	**41.83**	**43.54**

Table 11: SARI(↑) values of the instruction generated by BATprompt under two types of perturbations, P1 and P2, under different perturbations on Asset in Llama2-7b

Methods	S1	W3	W1	S2	S3	W1
MI	45.32	40.20	43.26	44.63	45.56	44.47
EvoPrompt	47.06	45.22	46.40	47.89	47.02	49.53
BATprompt	**47.47**	**45.57**	**46.51**	**48.03**	**48.22**	**50.14**

Table 12: SARI(↑) values of the instruction generated by BATprompt under two types of perturbations, P1 and P2, under different perturbations on Asset in GPT-4o-mini

ilarly, the results demonstrate that the prompt generated by BATprompt outperform other methods across multiple types of perturbations.

Our experimental results on the language understanding task on the model of Llama2-7b are shown in the Table 13. We calculated the average scores of each dataset under each perturbation. It can be seen that on the white-box model, the overall indicators of all datasets are lower. However, under the guidance of the instruction generated by BATprompt, the results of Llama2-7b have optimal values under both P1 and P2 perturbations. Through the above experiments, it can be seen that our method has sufficient robustness on both white-box and black-box.

D.4 Perturbation Transferbility

In this experiment, we test prompts trained on a P1 perturbation against a P2 perturbation, prompts trained on a P2 perturbation against a P1 perturbation. The results in the Table 15. In the results, E-BATprompt refers to a prompt trained with a different type of perturbation. When tested on the current category, we observe that, except for the Rouge-1 and Rouge-2 scores of E-BATprompt in P1, which are higher than those of BATprompt, BATprompt achieves the best performance across the remaining metrics. Moreover, BATprompt still outperforms other methods significantly on unfamiliar perturbations.

E Discussion

E.1 Number of iterations

In our adversarial training process, we set the number of iterations to five, as we observed that the prompts optimized by the LLM achieved optimal performance at the fifth iteration. To illustrate this, we used the text summarization task, a computa-

Type	Methods	CR	SST-2	MR	SST-5	TREC	AG'news	Avg.
P1	MI	31.0	36.8	31.8	15.0	23.5	48.5	31.1
	NI	34.0	40.3	37.0	24.5	22.0	74.0	38.6
	EvoPrompt	37.5	39.3	43.0	27.0	17.3	58.3	37.1
	BATprompt	**58.3**	**41.5**	**44.5**	**27.3**	**24.5**	**76.8**	**45.5**
P2	MI	28.4	33.0	31.2	17.5	21.2	55.8	31.8
	NI	31.9	36.1	31.9	24.1	23.2	76.8	37.3
	EvoPrompt	32.8	30.9	38.8	30.5	18.3	63.5	35.8
	BATprompt	**34.1**	**36.7**	**39.0**	**36.0**	**23.9**	**78.2**	**41.3**

Table 13: Average score(↑) of the prompts from different method on six language understanding datasets using Llama2-7b. The table in the upper half is a perturbation of type P1, and the table in the lower half is a perturbation of type P2

Method	CR		SST-5		TREC	
	w/o Pert.	w/ Pert.	w/o Pert.	w/ Pert.	w/o Pert.	w/ Pert.
MI	86.3	83.0	41.5	41.6	63.3	59.2
EvoPrompt	84.5	82.6	2.25	3.7	2	1.6
BATprompt	**86.8**	**83.5**	**44.5**	**42.5**	**65.3**	**61.1**

Table 14: The average of the scores(↑) of the instructions generated by BATprompt under two types of perturbations, P1 and P2, under different perturbations on the three datasets in GPT-4o-mini.

Type	Method	Rouge-1	Rouge-2	Rouge-L
P1	EvoPrompt	17.52	3.14	14.97
	E-BATprompt	**21.35**	**4.59**	16.10
	BATprompt	21.25	4.44	**16.66**
P2	EvoPrompt	17.19	2.93	14.69
	E-BATprompt	21.26	4.31	16.33
	BATprompt	**21.90**	**4.73**	**16.81**

Table 15: The average of Rouge-1(↑), Rouge-2(↑) and Rouge-L(↑) of prompts trained by different perturbation on the text summarization task on GPT-3.5-turbo.

tionally intensive text generation task, as an example. We tested the performance of prompts generated across six iterations (from 1 to 6), and the results are presented in the Figure 15.

For each iteration, we evaluate the prompts across different perturbation types and calculate their average performance. The results show a consistent improvement in prompt performance during the initial rounds, with the optimal performance observed around the fourth or fifth iteration. This trend suggests that the LLM effectively refines the prompts through iterative gradient-guided optimization and semantic space exploration, progressively approaching the optimal solution. However, beyond the fifth iteration, the performance of the prompts havea slightly declines. In order to speed up the inference experiment and reduce the cost, we choose the number of iterations to be 5

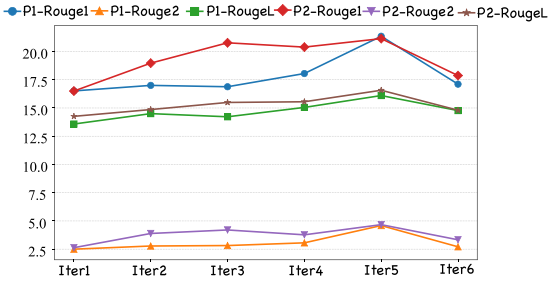

Figure 15: The relationship between the number of iterations and the performance of the prompt generated by BATprompt.

E.2 Performance on the original dataset

In this section, we evaluate the performance metrics of prompts generated by BATprompt when applied to tasks on unperturbed datasets. This assessment demonstrates that our method is not only robust to perturbed data but also has greate performance on unperturbed data. The results for the language understanding tasks are illustrated in the Figure 4. P1 and P2 represent different prompts generated by BATprompt for two types of perturbations. As shown in the Figure 4, BATprompt achieves the best performance on several datasets (e.g., SST-5 and TREC). For other tasks, even when it does not outperform all methods, its performance is comparable to the best results (e.g., SST-2 and CR).

Similarly, the results for text generation tasks are presented in Figure 16, with the left figure illustrating the text summarization task and the right figure depicting the text simplification task. The experimental findings indicate that BATprompt also achieves the best performance on text generation tasks. Notably, in the text summarization task, its metrics are significantly higher than those of the second-best method. In conclusion, the prompts generated by BATprompt not only maintain robustness on perturbed datasets but also deliver strong

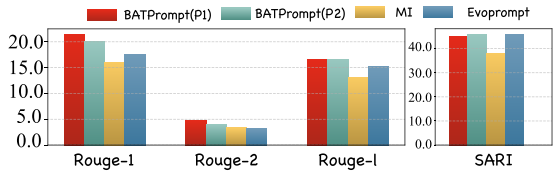

Figure 16: The performance of the prompts generated by BATprompt on undisturbed datasets of text simplification task and text summarization task.

Pert.	SST-5		AG'News		Asset	
	w/o Pert.	w/ Pert.	w/o Pert.	w/ Pert.	w/o Pert.	w/ Pert.
C1	——	——	——	——	38.12	47.64
C2	——	——	——	——	38.12	42.32
C3	36.0	37.5	78.3	79.3	38.12	39.47
S1	36.0	41.3	——	——	——	——

Table 16: Examples of perturbations that have a positive effect on the dataset when performing a task. Where the Score of SST-5 and AG 'news is their prediction accuracy, and the Score of Asset is the SARI value.

performance on original, unperturbed datasets.

F Future Work

Exceptions Explore: During our testing of the effects of different types of perturbations on task pairs in section 3, some intriguing observations emerged. While most perturbations negatively impacted task performance, certain perturbations surprisingly enhanced task execution. As shown in the Table 16, on the SST-5, AG's News, and Asset datasets, some perturbations appeared to unlock the latent potential of the large model, enabling it to perform better on the tasks.

We hypothesize that applying these perturbations to the input introduces diversity, encouraging the LLM to engage its reasoning abilities rather than relying on specific representations. Additionally, such perturbations may influence the model's attention distribution, helping to resolve ambiguities in the input and ultimately enhancing task performance. This presents a highly valuable avenue for exploration. We could investigate a fixed perturbation strategy that consistently enhances the performance of LLMs when processing text inputs.

G Optimal Prompts

We publish the prompts that are optimal on different tasks after BATprompt generation and the prompts for Manual Instruction and Natural Instruction as baseline. Including language understanding(Table 17 and Table 18), text summarization(Table 19), and text simplification(Table 20).

Dataset	Method	Content
SST-2	Manual Instruction	Please perform Sentiment Classification task. Given the sentence, assign a sentiment label from ['negative', 'positive']. Return label only without any other text.
	Natural Instruction	In this task, you are given sentences from movie reviews. The task is to classify a sentence as "great" if the sentiment of the sentence is positive or as "terrible" if the sentiment of the sentence is negative.
	BATprompt(P1)	· Decide if the text expresses a 'negative' or 'positive' sentiment without taking into account any extra details, even if there are mistakes in spelling and typos compared to the original text. · For this assignment, you will receive sentences that have been altered from movie reviews. Your job is to determine whether the sentiment of the sentence is positive or negative and classify it accordingly as either "positive" or "negative.
	BATprompt(P2)	· Conduct sentiment analysis by categorizing the sentiment of a sentence as ńegativeór ṕositive. Output only the sentiment label with no other information. · For this assignment, you will be provided with sentences taken from movie reviews. Your goal is to determine whether each sentence conveys a positive or negative sentiment by classifying it as either "positive" or "negative." The reviews may discuss specific characters, actions, critiques, relationships between characters, performances, or negative aspects of the works.
CR	Manual Instruction	Please perform Sentiment Classification task. Given the sentence, assign a sentiment label from ['negative', 'positive']. Return label only without any other text.
	Natural Instruction	In this task, you are given sentences from movie reviews. The task is to classify a sentence as "great" if the sentiment of the sentence is positive or as "terrible" if the sentiment of the sentence is negative.
	BATprompt(P1)	· Analyze the sentiment of the modified text as either 'negative' or 'positive', disregarding any intentional spelling and grammar mistakes, and provide only the corresponding label as the result. · For this assignment, you will receive original movie reviews. Your goal is to determine if a sentence has a positive sentiment by classifying it as "positive," or if it has a negative sentiment by classifying it as "negative.
	BATprompt(P2)	· Complete a Sentiment Classification task by analyzing a modified text with enhanced details and extra information. Provide a sentiment label of either ńegativeór ṕositive, and only submit the label. · For this assignment, you will receive modified sentences from movie reviews. Your goal is to categorize a sentence as "positive" if it expresses positive sentiment or as "negative" if it expresses negative sentiment, regardless of any changes made to the text such as including negative adjectives or conveying different meanings using synonyms.
MR	Manual Instruction	Please perform Sentiment Classification task. Given the sentence, assign a sentiment label from ['negative', 'positive']. Return label only without any other text.
	Natural Instruction	In this task, you are given sentences from movie reviews. The task is to classify a sentence as "great" if the sentiment of the sentence is positive or as "terrible" if the sentiment of the sentence is negative.
	BATprompt(P1)	· Analyze the sentiment of the revised text given, ignoring any mistakes in spelling, typos, or edits made. Categorize the sentiment as either 'positive' or 'negative' without any additional details. · In this assignment, you will be given altered versions of movie reviews. Your goal is to classify each sentence as either positive for examining deeper themes and emotions, or negative for focusing solely on surface-level aspects of the films.
	BATprompt(P2)	· Complete the Sentiment Classification task by assigning a sentiment label of either 'negative' or 'positive' to the provided shortened sentence and return the label only. · For this task, you will be given distorted sentences from movie reviews. Your objective is to identify whether a sentence conveys a negative sentiment and label it as ńegative, or if it conveys a positive sentiment and label it as ṕositive.
SST-5	Manual Instruction	Please perform Sentiment Classification task. Given the sentence, assign a sentiment label from ['terrible', 'bad', 'okay', 'good', 'great']. Return label only without any other text.
	Natural Instruction	In this task, you are given sentences from movie reviews. Based on the given review, classify it to one of the five classes: (1) terrible, (2) bad, (3) okay, (4) good, and (5) great.
	BATprompt(P1)	· Perform a Sentiment Classification task by assigning a sentiment label from ['terrible','bad', 'okay', 'good', 'great'] to the modified sentence. Only include the label, no extra information needed. · During this activity, you will be presented with sentences from movie reviews that have been modified to provide more general and subjective opinions on the movies. Your task remains the same: classify each review into one of the categories - terrible, bad, okay, good, or great - based on its sentiment and tone.
	BATprompt(P2)	· Perform Sentiment Classification. Given the altered text, assign a sentiment label from ['terrible', 'bad', 'okay', 'good', 'great']. Return the label only. · In this task, you are given sentences from movie reviews. Based on the given review, classify it to one of the five classes: (1) great, (2) good, (3) okay, (4) bad, and (5) terrible.

Table 17: Specific prompt of Manual Instruction(baseline), Natural Instruction, BATprompt in P1 and BATprompt in P2 in language understanding task.

Dataset	Method	Content
AG's News	Manual Instruction	Please perform News Classification task. Given the news item, assign a label from ['World', 'Sports', 'Business', 'Tech']. Return label only without any other text.
	Natural Instruction	In this task, you are given a news article. Your task is to classify the article to one out of the four topics "World", "Sports", "Business", "Tech" if the article"s main topic is relevant to the world, sports, business, and technology, correspondingly. If you are not sure about the topic, choose the closest option.
	BATprompt(P1)	· Complete a News Classification assignment. Select a category from [Ẃorld, Śports, Ƃusiness, Ťech] for a news item that may have different spellings and minor grammar mistakes compared to the original text. Only provide the label without extra information. · For this assignment, you will be provided with a news article. Your goal is to categorize the article into one of four topics: "World," "Sports," "Business," or "Tech" based on its main focus. If the article discusses trends in the U.S. stock market or gaming news, select either the "Business" or "Tech" category depending on the emphasis. If the article includes nonsensical or random words that do not make sense, choose the "Other" category.
	BATprompt(P2)	· Classify news by assigning a label from the options ['World', 'Sports', 'Business', 'Tech'] to each news item. Only provide the assigned label, without any extra text. · For this task, you will receive a news article and your objective is to classify it under one of the four categories: "World", "Sports", "Business", "Tech" based on the articleś main subject. If you are uncertain about the category, please select the most appropriate option. The Revised text includes extra details on different subjects and individuals like Tommy Tuberville and Major League Soccer, which are not referenced in the Original text. Additionally, the Revised text offers more precise information and examples, such as the mishandling of the conferenceś top position and the difficulties faced by Major League Soccer.
TREC	Manual Instruction	Please perform Question Classification task. Given the question, assign a label from ['Description', 'Entity', 'Expression', 'Human', 'Location', 'Number']. Return label only without any other text.
	Natural Instruction	You are given a question. You need to detect which category better describes the question. Answer with "Description", "Entity", "Expression", "Human", "Location", and "Number".
	BATprompt(P1)	· Determine the appropriate category for the text by providing only the label without any extra details. Choose from Description, Entity, Expression, Human, Location, or Number. · Identify the appropriate category for the given text by selecting from "Description", "Entity", "Expression", "Human", "Location", and "Number" depending on the context.
	BATprompt(P2)	· Complete a Question Classification activity where you are provided with a shortened version of the original question without key words or phrases, and categorize it into one of the following labels: Description, Entity, Expression, Human, Location, or Number. Provide only the assigned label as the output. · Decide on the correct category for the text provided. Select from "Description", "Entity", "Expression", "Human", "Location", and "Number".

Table 18: Specific prompt of Manual Instruction(baseline), Natural Instruction, BATprompt in P1 and BATprompt in P2 in language understanding task.

Method	Content
Manual Instruction	How would you rephrase that in a few words?
BATprompt(P1)	Please give a concise overview of the main idea communicated in the text.
BATprompt(P2)	Identify the main idea or central theme of the text.

Table 19: Manual Instructions as the baseline and instructions with best performance generated by BATprompt (either P1 or P2) on Xsum.

Method	Content
Manual Instruction	Simplify the text.
BATprompt(P1)	Rephrase the text using easier words without including any additional details.
BATprompt(P2)	Rephrase the text using more straightforward language and clearer wording to enhance understanding.

Table 20: Manual Instructions as the baseline and instructions with best performance generated by BATprompt (either P1 or P2) on Asset.