Learning to Rewrite Prompt for Boostrapping LLMs on Downstream Tasks

Anonymous ACL submission

Abstract

As versatile agents, large-scale language models (LLMs) have demonstrated impressive performance across various domains. Their capabilities in language-based pattern recognition and machine learning have garnered significant attention and have been applied to numerous tasks with remarkable success. However, different LLMs still rely on specific instruction prompts, and the design of prompt tokens is still heavily dependent on manual design, which hinders the widespread application of LLMs. In response to this challenge, we propose a concise and effective input optimization method, which consists of two modules: original input rewriting and filtering. Inspired by the concept of collaboration between large and small models, we introduce a rewriting module between input prompts and LLMs inference. This module rewrites the input component based on the preferences of the LLMs for the data. The filtering module performs the quality inspection on the rewritten data and filters out invalid and hallucinatory data. Experimental results on language pattern recognition tasks verify that our rewriting and filtering method effectively transforms ambiguous data into more precise input prompts. In comparison to the original inputs, performance improvement is consistently observed across various tasks.

1 Introduction

Large language models (LLMs) trained on terabytes of tokenized data have achieved groundbreaking progress across a myriad of pattern recognition tasks. Those LLMs, such as GPT-3.5 (OpenAI, 2022), typically use manually crafted or predefined prompt templates as directives to guide the model in accomplishing various tasks.

Previous studies (Qin and Eisner, 2021; Liu et al., 2021b) demonstrate that LLMs exhibit sensitivity to prompts, and manual design of appropriate prompts can be a laborious and time-consuming task. To address this issue, soft prompts (Qin and Eisner, 2021) convert discrete prompt words into continuous vectors, enabling end-to-end training. Prefix Tuning (Li and Liang, 2021) inserts a series of continuous task-specific prefixes at the beginning of the input, then fine-tunes these prefixes while keeping the other parameters frozen. Furthermore, APO (Pryzant et al., 2023) optimizes the prompt by using the discrete feedback of the LLMs as gradient updates. RLPrompt (Deng et al., 2022) employs reinforcement learning to conduct a directionless Monte Carlo search in the semantic space and get impressive results.

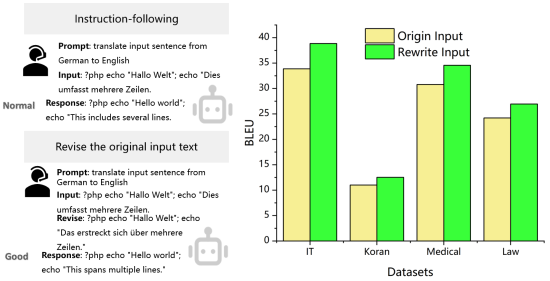

Figure 1: The example on the left illustrates the sensitivity of LLMs to input data. In the context of the same translation task, providing more detailed input (bottom) leads to better results. The right figure shows the performance improvement of the LLMs on various translation datasets after rewriting.

In the context of using LLMs, tokens typically consist of two components: *instruction* and *input*. These two components are usually concatenated and fed into the LLMs. The aforementioned approaches primarily optimize the $instruction$ to achieve improved results. However, in tasks like machine translation, the model is equally sensitive to the $input$. As depicted in Figure 1 left, in the task of translating German to English, LLMs provide better responses when the original German text is modified while preserving the original meaning. In tasks such as machine translation, sum-

marization and abstraction, the *instruction* component is typically short and may not require significant modifications. However, the *input* component plays a major role in the input tokens in these tasks. Therefore, the benefits brought by optimizing the *input* part may be greater on this type of task.

Inspired by ALMs (Mialon et al., 2023), we propose the Rewriting Original Inputs (ROI) strategy, which aims to optimize the important *input* component before feeding it into LLMs. Rewriting involves improving the grammar, expression, and other linguistic aspects of the input while maintaining the original meaning intact. We explore two rewriting methods: 1) utilizing the LLMs themselves. 2) employing a language model fine-tuned on the rewritten dataset. Neither of those methods requires training of the LLMs. We draw inspiration from back translation in machine translation and use this way to construct rewriting data and train the rewriting model. As shown in Figure 1 right, on several translation datasets, the results of LLMs show varying degrees of improvement when the original input data is rewritten while keeping the original meaning. For tasks that are unsuitable for constructing rewritten data, we use LLMs themselves to modify the *input* component, which turns out to be effective. Furthermore, we find that not all rewriting yields positive gains. So we incorporate a filtering module to eliminate sentences that introduce hallucinations or alter the original meaning. For those data, we revert to using the original *input*. We conduct experiments on both natural language understanding (NLU) and language generation (NLG) tasks. For the NLG task, we performed experiments on four machine translation datasets, while for NLU tasks, we chose the GLUE benchmark.

Regarding the contributions of this paper, we observe that existing prompt engineering methods yield limited benefits for tasks where the *input* component plays a predominant role. Building upon this observation, we then introduce the Rewrite Original Input (ROI) module, coupled with a filtering algorithm, to enhance the performance of LLMs on these downstream tasks. In this method, there is no need to train any parameters in the LLMs and the framework is applicable to a wide range of different LLMs. The experimental results on both NLU and NLG tasks verify that the ROI module effectively transforms ambiguous data into more precise and explicit input prompts. Compared to the original input, our ROI method reaches consistent and notable performance improvements across all tasks.

2 Related Work

Large Language Models. Significant strides in the domain of deep learning have been facilitated by the advent of large language models. These models commonly utilize transformer-based (Vaswani et al., 2017) architecture and amass models that comprise hundreds of millions of parameters via layer stacking. Such models are pre-trained on vast quantities of unlabeled data, utilizing techniques like Masking. Examples of such models include GPT-3.5 (Brown et al., 2020), LLaMA (Touvron et al., 2023), PaLM (Chowdhery et al., 2022), and more. The extensive parameter scale and voluminous training data equip LLMs with formidable natural comprehension capabilities. To further improve the performance of LLMs on unseen tasks, FLAN (Wei et al., 2021) incorporates an additional instruction-tuning stage after pre-training, enhancing the ability of the LLMs to handle diverse and complex user instructions.

Augmenting Large Language Model without Training. Training large language models from scratch poses a significant challenge for researchers due to their massive parameter size and the need for extensive pre-training data. LLMs exhibit excellent context-learning capabilities, allowing the completion of specific tasks through contextual prompts, known as in-context learning (ICL) (Dong et al., 2022). Unlike supervised learning, in-context learning does not require parameter updates but directly uses LLMs for prediction. LLMs can understand given demonstrations and make accurate predictions. The performance of ICL heavily depends on the nature of demonstrations, including both their format and sequence. KATE (Liu et al., 2021a) indicates that the selection of nearest-neighbor samples as context instances can significantly enhance the performance of LLMs. Additionally, Gonen (Gonen et al., 2022) proposes selecting instances with low perplexity, while Rubin (Rubin et al., 2021) puts forth a two-stage, retrieval-based method for demonstration selection. To handle specific inputs, an unsupervised retriever is first constructed to identify examples similar to candidate instances, following this, a supervised retriever selects appropriate demonstrations from among these candidates.

Augmenting Large Language Model with

Prompt Tuning. The utilization of a shared model across tasks has significantly propelled the application and development of LLMs. However, the reliance on textual prompts requires manual design, and even with carefully crafted prompts, their performance still falls short compared to model fine-tuning. As a result, current work primarily aims to enhance the performance of LLMs through differentiable tuning of prompts. Brian (Lester et al., 2021) and Li (Li and Liang, 2021) propose a method called prefix tuning to adjust soft prompts for tuning frozen models. The tokens of soft prompts are learnable vectors, and they append the soft prompt vectors at the beginning of the input text, inputting the combined sequence into the model, thus realizing end-to-end training on the training set. Similarly, P-Tuning (Liu et al., 2021b) adds an encoder module in front of LLMs to fine-tune prompts at the embedding level, which is more flexible compared to prefix tuning. In addition, APE (Zhou et al., 2022) and RLPrompt (Deng et al., 2022) incorporate reinforcement learning into prompt optimization. They design scoring functions in response to model feedback and make discrete-level corrections to prompts. Beyond prompt optimization, methods for parameter-efficient fine-tuning (PEFT) have also been proposed to make LLMs adapt to downstream tasks efficiently without the need to fine-tune all parameters. Techniques like Adapter-Tuning (He et al., 2021) insert smaller neural network layers or blocks into pre-trained networks, and only these adapter parameters are updated for downstream tasks. Similarly, LoRA (Hu et al., 2021) approximates the parameter update of the weight matrix W of the model by learning a low-rank matrix with fewer parameters. All the aforementioned methods can be categorized as part of ALMs. Besides these techniques, ALMs also improve the performance of LLMs by retrieving external information (Borgeaud et al., 2022) or training auxiliary models (Yang et al., 2022).

3 Method

In this section, we begin by giving a definition of the process of how LLMs are utilized to generate outputs. Subsequently, we present our method in a comprehensive manner. Finally, we analyze the differences between our approach and other prompt engineering methods.

To start, we offer a formalized definition of how LLMs complete downstream tasks. LLMs are constructed based on the transformer architecture, which comprises deep networks with multiple layers of stacked multi-head attention mechanisms. Unlike conventional language models, LLMs are characterized by the parameter size, pre-training data, and computational demands. LLMs employ prompt-based inference mechanisms, enabling the expression of various natural language tasks through instruction prompts. For specific tasks, corresponding instruction templates $P = \{p_1, p_2, ..., p_m\}$ are often designed by human beings, where m is the length of the instruction. Meanwhile, some tasks are accompanied by corresponding input sequences $X = \{x_1, x_2, ..., x_n\}$. After embedding the input sequences into the instruction templates or splicing with them, we get the complete input prompts. For example, for sentiment analysis tasks, the input prompt can be transformed into

$$\mathrm{X}_{prompt} = Is, the, ..., positive, or, negative, ?\{X\} \quad (1)$$

Then, LLMs generate responses $Y = \{y_1, y_2, ..., y_k\}$ based on complete instructions. k is the output length. The generation of each token y_j can be represented as

$$y_j = argmax_{y_j \in V} P_M(y_j | X_{prompt}, y_{i<j}), \quad (2)$$

where V represents the vocabularies and the prediction of the token j relies on both the X_{prompt} and the preceding tokens. The generation process of LLMs ceases when it produces an end-of-sentence (eos) token, thereby generating a complete response sequence Y.

3.1 Rewriting Original Input (ROI)

We first introduce our rewriting module in detail. Previous studies have demonstrated that the LLMs are highly sensitive to the instruction P, highlighting that even slight modifications can result in significant variations in the outputs of the model. As a consequence, a considerable amount of research has emerged that aims to optimize the design of instruction prompts. However, for some tasks the instruction P is relatively fixed, the benefits of optimizing this aspect are limited. Instead, the input component X emerges as more crucial.

We observe that input sentences expressing the same meaning may elicit different responses from the LLMs under the same instruction template. In other words, LLMs are also sensitive to input components. In real-world scenarios, LLMs face the

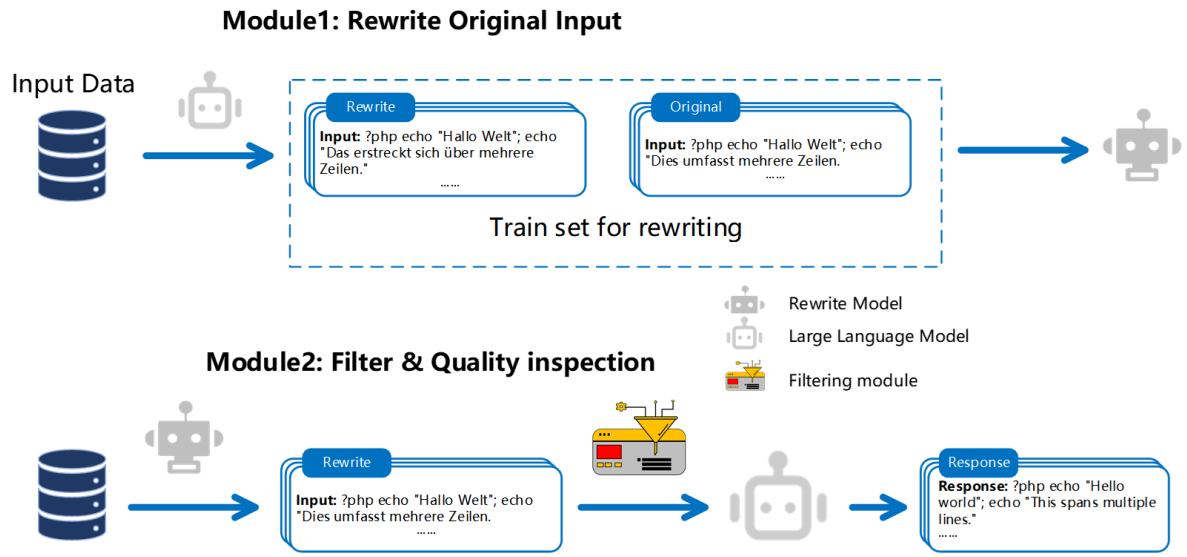

Figure 2: The pipeline of our proposed method for boostrapping Large Language Models. The raw data is first input to the LLMs to construct the rewriting data. Subsequently, a filtering process is applied to retain only the rewritten data that demonstrates improved performance, while the remaining data continues to utilize the original data. The generated rewritten data is then used to train the rewriting model. During testing, the original data is first input to the rewriting model to obtain rewritten sentences. These rewritten sentences are subsequently input to the LLMs to generate the final results.

challenge of dealing with different writing styles and preferences from users. They need to be capable of producing sensible outputs for these diverse expressions. To this end, we propose to modify the input data before it is processed by the LLMs. We introduce a rewriting module that operates on the input data as depicted in Figure 2. This can be expressed by the following equation:

$$y_j = argmax_{y_j \in V} P_M(y_j | R(X_{Prompt})). \quad (3)$$

We propose that LLMs have their own preferences regarding the data they process, which may diverge from conventional human expression patterns. Therefore, we design a process where the original input data is rewritten using either the LLMs itself or a language model with fewer parameters. Specifically, inspired by the technique of back translation in machine translation, we utilize LLMs to write back the training set output as input and form the rewritten data with the original input of the training set, as shown in the first part of figure 2. We then use this train set to fine-tune a language model with fewer parameters and we call this model a rewriting model. The rewriting model learns the preferences of the LLMs towards input data. When new test data is available, we first input the input component to the rewriting model and then pass the rewritten result to LLMs for further processing.

We give an example of our rewriting method. For the machine translation task from German to English, we first back-translate the training data from English to German and combine it with the original German input to form the rewritten data. This data is used to train the rewriting model. During testing, the German input is first optimized through the rewriting model and then input into LLMs.

For judgment tasks involving grammar, sentiment, etc., there is no one-to-one correspondence between input and output. In this scenario, where it is not possible to construct a rewriting dataset, we leverage the capabilities of the LLMs themselves to perform rewriting, so that the input component adapts to the preferences of LLMs themselves.

Similar to other prompt engineering methods, our rewriting method is not limited to a single dataset. Furthermore, since any LLMs can be paired with a rewritten model, our method is generalizable to different models. To verify this, we test on different 7B LLMs. Our method is not applicable to tasks involving reasoning, planning, and other abilities. In these tasks, the input component is relatively fixed, and prompts need to focus more on activating the reasoning capabilities of the LLMs, so methods such as Chain of Thoughts (Wei et al., 2022) are more suitable. Furthermore, we have found that not all data receive positive ben-

efits from rewriting. Due to the unstable output of LLMs, they sometimes produce so-called hallucinations. Therefore, it is necessary to filter and select the data after rewriting.

3.2 The Filtering Algorithm

During the rewriting process, it is inevitable that some noise data will be generated, and not all rewrites are beneficial. To address this, we introduce a filtering mechanism that follows the rewriting model. This mechanism helps eliminate noise data by replacing unhelpful rewrites with original sentences. This noise elimination process also contributes to enhancing the performance of the training of the rewriting model. Specifically, for different tasks, we calculate similarity using pertinent evaluation metrics and set thresholds for filtering. The relevant algorithm is illustrated in Algorithm 1. For instance, in a translation task, we can use word-level edit distance to calculate the similarity between the original text and the rewritten sentences. When the similarity between the rewritten sentences and the original text is low, it might be because LLMs have outputted hallucinations, or that extensive rewriting increases the training difficulty for the rewriting model. Therefore, we replace them with the original text, preserving only the rewritten data that have a small degree of change and are effective.

We first utilize the ROUGE-L metric to calculate the similarity. Only when the ROUGE-L score between the original and the rewritten sentence surpasses a certain threshold, we add it to the rewritten dataset. Furthermore, as rewriting is analogous to a language translation task, we use BLEU as another metric to evaluate similarity. Rewriting often involves rearranging word orders, deleting inappropriate words, adding new terms, etc., which is directly related to the concept of edit distance. Therefore, we also adopt edit distance as a similarity measure. The relevant formula is as follows:

$$sim = \frac{L_{total} - l_{dist}}{L_{total}}; \tag{4}$$

The L_{total} in the formula indicates the sum of the lengths of the two sentences. l_{dist} denotes the editing distance between two sentences, which is the minimum number of editing operations required to convert from one to the other.

Algorithm 1 Filtering Algorithm

Input: Rewrite dataset $\mathcal{R} = \{\emptyset\}$, Rwrite function $\mathcal{F}$, Original dataset $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$
Output: Rewrite dataset $\mathcal{R}$

```
Rewrite the original statement and qualify it
for (x_i, y_i) ∈ D do
    r_i = F_task(x_i) or r_i = F_task(x_i, y_i);
    sim_score = metric(r_i, x_i);
    if sim_score < γ then
        R_i = x_i;
    else
        R_i = r_i;
    end if
end for
return R;
```

4 Experiments

4.1 Datasets and Setup

We conduct experiments on both NLU and NLG tasks to investigate the impact of our method on various downstream tasks. For the NLU task, we chose the GLUE benchmark (Wang et al., 2018), as it combines instruction prompts and data inputs. Therefore, we modify and optimize the entire input. For the NLG task, we focus on translation tasks and utilize datasets from different domains. We tailor our ROI module based on the requirements and characteristics of each task. This approach is adopted to enhance the effectiveness and efficiency of our solution across different downstream tasks. Detailed descriptions of the corresponding datasets and rewriting methods are provided as follows:

CoLA dataset is a binary classification task dataset, the objective of which is to determine whether a sentence is grammatically correct.

SST-2 dataset is a text classification dataset targeted for sentiment analysis. By leveraging the sentiment orientation of sentences.

MRPC dataset focuses on examining the understanding of a language model of similar sentences. Each data entry provides two sentences and requires the model to judge whether they express the same meaning.

QNLI dataset is a collection of question-sentence pairs designed for question-answering tasks. In this dataset, each example consists of a question and a corresponding sentence, and the model is required to determine whether the sentence entails or is logically implied by the question.

		CoLA	SST-2	MRPC	QNLI	WNLI	QQP	RTE	AVG
					Accuracy ↑				
llama-7b-hf	origin input	0.7133	0.8268	0.6637	0.4455	0.4913	0.5353	0.3754	0.5787
	Rewrite	0.6711	0.7729	0.6649	0.4958	0.5086	0.5422	0.4765	**0.5902**
					F1 Score ↑				
	origin input	0.8132	0.8015	0.5829	0.5951	0.3952	0.1905	0.3268	0.5293
	Rewrite	0.772	0.7537	0.6118	0.6615	0.5015	0.2791	0.4413	**0.5744**
					Accuracy ↑				
flan-alpaca-gpt4	origin input	0.6567	0.9025	0.5367	0.2811	0.5774	0.6507	0.3898	**0.5707**
	Rewrite	0.6673	0.8990	0.5220	0.2912	0.4929	0.6524	0.4255	0.5643
					F1 Score ↑				
	origin input	0.7879	0.8986	0.5190	0.3832	0.6105	0.1293	0.199	0.5039
	Rewrite	0.7974	0.8954	0.5208	0.4019	0.6250	0.1476	0.1675	**0.5079**

Table 1: The experimental results from the GLUE benchmark feature two different versions of the alpaca model: llama-7b-hf and flan-alpaca-gpt4. The former has 7 billion parameters, while the latter holds 3 billion.

WNLI dataset comprises a set of sentence pairs, each corresponding to a binary classification task. In each example, there is a sentence containing a pronoun (e.g., "he" or "she"), and another sentence containing the possible antecedent of that pronoun. The task for LMs is to determine whether the pronoun in the first sentence correctly refers to the antecedent in the second sentence.

QQP is also a dataset for detecting sentence similarity. Based on the meaning of the sentences.

RTE provides a premise and a hypothesis for each data entry. LLMs need to judge whether the hypothesis can be inferred from the premise.

For NLG tasks, we conduct experiments on machine translation task. We utilized four different domain-specific German-to-English translation datasets: **IT**, **Medical**, **Koran**, and **Law**.

4.2 Implementation Details

We conduct experiments on different versions of the Alpaca (Taori et al., 2023) model with varying parameter sizes and pre-training data. The original alpaca model is based on the LLaMA model and is fine-tuned with $52k$ instructions. For the rewrite models, we select models with fewer parameters, such as mBart (Liu et al., 2020) and mT5 (Xue et al., 2020). For different versions of the alpaca model, we set the temperature coefficient to 0.1 and the num beams to 4. The initial learning rate is set to 2e-5, the batch size is 4, and the dropout is 0.3. During the inference stage, depending on the task, we would first input the original X_{prompt} into the rewrite model or the LLMs self to get the rewrite sentence, and then complete the corresponding downstream tasks.

4.3 Main Results

Table 1 presents the accuracy and F1 scores of ROI and original inputs on various NLU tasks. We use the same set of training parameters for all tasks. For each model and each evaluation metric, the original and our rewritten results are listed separately. The results demonstrate that ROI can be applied to various downstream tasks and, in most cases, improve the accuracy and F1 scores of the rewritten inputs. For example, on the RTE dataset, the performance of ROI is about 10% higher than that of the original inputs on llama-7b-hf. On the QQP dataset, our method improves F1 by 8% compared to the original input on llama-7b-hf model. This observation holds across different parameter sizes of the alpaca model.

It can be seen that there is some inconsistency in the improvement of our method on different datasets. For example, ROIs perform poorly on the SST-2 and CoLA datasets. Compared to other tasks, these two datasets are relatively simple, and the performance of LLMs alone is already high, making it difficult for ROI to significantly improve the performance. Additionally, we find that the improvement of ROI is more pronounced on larger parameter alpaca models. For instance, on the RTE dataset, the alpaca model with 7B parameters achieves an improvement of about 10% in accuracy, while the model with 3B parameters only achieves an improvement of about 4%. Roughly the same trend is observed in other datasets. We analyze that LLMs with larger parameters have a better understanding of language and a stronger knowledge reserve which can carry out more effective rewriting.

The experimental results of using ROI in NLG tasks are presented in Table 2. We use the same

		BLEU ↑	Edit Dist ↑
Origin Input	IT	27.75	**0.6220**
	Koran	12.37	**0.5758**
	Medical	31.67	0.6370
	Law	24.21	0.6350
Rewrite	IT	**28.06**	0.6130
	Koran	**12.51**	0.5668
	Medical	**34.57**	**0.6434**
	Law	**26.94**	**0.6369**

Table 2: The experimental results for NLG task. IT, Koran, Medical, and Law are four different fields of translation datasets of de-en. The rewrite model is uniformly used in all experiments with mbart-cc-25.

training parameters for all datasets in different domains. It can be seen that the alpaca model does not perform well on the translation dataset for these four domains which did not appear in the pre-training phase. However, by using appropriate ROI, the translation performance improves to a certain extent in most domains. In the Medical domain, the translation performance shows the most significant improvement. Compared to using the original inputs, the ROI method achieves a BLEU score increase of 2.9 and an improvement of 0.64% in Edit Distance Similarity.

We observe that our method shows limited effectiveness in some domains. In the koran domain, our method shows a positive enhancement of 0.14 in BLEU and a slight decrease in edit distance. In the IT domain, our method has an improvement of 0.31 in BLUE and a slight decrease in edit distance. Therefore, the effectiveness of the ROI in improving the performance of LLMs is dependent on the dataset used. Similar to NLU tasks, the ROI method exhibits significant variations in performance improvements across different datasets. Despite these variations, it remains applicable to the majority of datasets.

4.4 Ablation Studies

In this section, we perform ablation studies on various parts of our method. Initially, we conduct ablation experiments on different filtering algorithms. Subsequently, we explore the impact of various rewriting models. Finally, we examine the volume of training data required for the rewriting model.

The experimental results of comparing different filtering metrics in NLG tasks are presented in Table 3. We use three evaluation metrics to construct similarity scores for filtering out noisy or poorly rewritten data. When the similarity score between the rewritten sentence and the original sentence is low, it often means that the rewritten sentence contains hallucinations. In such cases, we still use the original data as inputs for LLMs. Additionally, a high similarity score between the rewritten sentence and the original sentence indicates that the changes made by the rewrite model are minimal, making it easier to train the model. However, this also means that the effectiveness of the rewrite is not significant. Conversely, a low similarity score between the rewritten sentence and the original sentence indicates that the model made significant changes to the original sentence, which may lead to better rewrite results. However, this also increases the difficulty of training the rewrite model. To balance these two cases, we conduct experiments with different threshold values. The results in Table 3 show that the RougeL metric performs the best for filtering out noisy data, and setting a threshold of 0.5 results in the best performance.

The experimental results of comparing different filtering algorithms in NLU tasks are presented in Table 4. Similar to NLG tasks, we use three evaluation metrics and test different threshold values. Unlike translation tasks, we require the model to modify the entire prompt input, making it easier to introduce noisy data during the rewriting process. Additional irrelevant information can interfere with the judgment of LLMs, resulting in irrelevant responses. The experimental results show that the RougeL metric still performs the best, followed by the Edit Distance metric. Moreover, setting a filtering threshold of 0.5 results in the best rewrite performance. When using the RougeL metric with a threshold of 0.5, the accuracy can reach 0.4765,

	thresholds	BLEU ↑	Edit Dist ↑
RougeL	0.3	**4.002**	0.486
	0.5	3.310	0.453
	0.8	4.673	**0.497**
BLEU	0.3	5.915	0.313
	0.5	5.915	0.313
	0.8	5.195	0.313
Edit Dis	0.3	2.797	0.2154
	0.5	4.532	0.3044
	0.8	5.915	0.3133

Table 3: The same thresholds are set on the three evaluation metrics for the experiments. For this experiment, we uniformly use the IT domain de-en translation data.

and when using the Edit Distance metric with a threshold of 0.8, the F1 score can reach 0.4429.

The experimental results of comparing different rewrite models in the translation task are presented in Table 5. We select three language models with different parameter sizes and pre-training data. the results show that the large parameter rewrite model outperforms the small parameter rewrite model significantly. On the four datasets, the performance of using mbart-cc-25 is significantly higher than that of using tiny-mbart. We analyze that the large parameter model has a more abundant pre-training dataset and stronger understanding ability. In addition, when the parameter sizes are comparable, the rewrite performance of mbart is better than that of mT5. As shown in Table 5, using mbart performs significantly better than using mT5 on the four domains.

To validate the potential of using ROI in practical applications, we conduct experiments on translation datasets from various domains by training the rewriting model with varying amounts of data. We start with a small number of randomly selected data through ROI and gradually increase the size of the training data. As shown in Figure 3, as the training data increases, the rewriting quality improves, and the translation performance improves. Besides, only about 5000 pieces of rewritten data need to be used to exceed the translation performance using the original input on several domains. This suggests that by constructing a small amount of rewriting data, we can enable the rewriting model to learn suitable rewriting patterns, thereby demonstrating the practical applicability of ROI. Moreover, we find that due to the unstable quality of the ROI, the translation performance of LLMs does not continue to improve when we increase the number of rewritten data to 10000. This indirectly confirms the effectiveness of the filtering algorithm in selecting high-quality rewriting instances.

	thresholds	Accuracy ↑	F1 Score ↑
RougeL	0.3	0.4257	0.4270
	0.5	**0.4765**	0.4413
	0.8	0.4228	0.4272
BLEU	0.3	0.3754	0.3268
	0.5	0.3754	0.3268
	0.8	0.3754	0.3268
Edit Dis	0.3	0.3826	**0.4429**
	0.5	0.4115	0.4398
	0.8	0.4007	**0.4429**

Table 4: The same thresholds are set on the three evaluation metrics for the experiments. For this experiment, we uniformly chose the RTE dataset.

		BLEU ↑	Edit Dist ↑
mbart-cc-25	IT	3.31	0.4530
	Koran	11.8938	0.5354
	Medical	6.8341	0.4218
	Law	14.8217	0.5299
tiny-mbart	IT	0.0941	0.1514
	Koran	0.0927	0.3230
	Medical	0.0565	0.2545
	Law	0.0733	0.3515
mT5	IT	0.1977	0.1378
	Koran	0.2482	0.2770
	Medical	0.9383	0.3028
	Law	0.4808	0.2480

Table 5: mbart-cc-25 tiny-mbart mT5 denotes three rewrite models, each with different parameter counts.

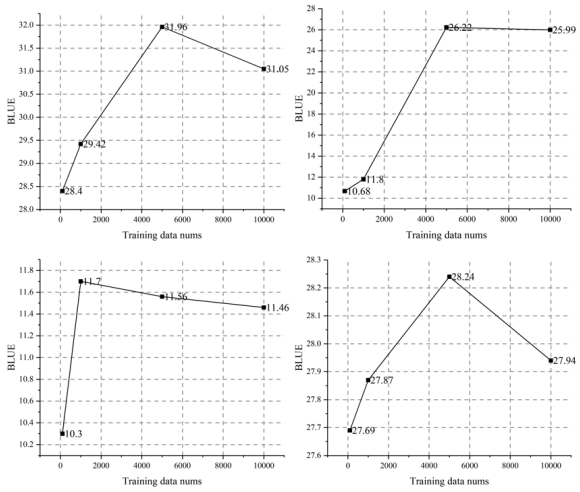

Figure 3: Translation performance when training a rewriting model using 100, 1000, 5000, and 10000 pieces of rewritten data, respectively.

5 Conclusion

In this paper, we propose original input rewriting with filtering, a simple and versatile framework for optimizing input components to LLMs. This method mainly focuses on tasks in that the instruction component is relatively simple but the input part is important. We optimize the input component by rewriting model to make it more consistant with the preferences of LLMs for data. Through extensive experiments on multiple benchmarks, we validate its effectiveness. The simplicity and efficacy of our framework make it a promising approach with substantial potential.

6 Limitation

Our proposed method has demonstrated promising performance on various versions of the Alpaca model. However, we acknowledge that we have not yet conducted experiments on larger-scale LLMs like GPT-3.5. At the same time, our method is primarily limited to single-turn question-answering language tasks. We look forward to addressing all of these issues in our future work.

References

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

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*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A Smith, and Luke Zettlemoyer. 2022. Demystifying prompts in language models via perplexity estimation. *arXiv preprint arXiv:2212.04037*.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. *arXiv preprint arXiv:2106.03164*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021a. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. 2023. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*.

OpenAI. 2022. Introducing chatgpt.

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*.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Kevin Yang, Nanyun Peng, Yuandong Tian, and Dan Klein. 2022. Re3: Generating longer stories with recursive reprompting and revision. *arXiv preprint arXiv:2210.06774*.

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*.