Beyond Prompt Engineering: A Reinforced Token-Level Input Refinement for Large Language Models

Guang Huang[1*], **Yanan Xiao**[2*], **Lu Jiang**[3], **Minghao Yin**[2], **Pengyang Wang**[1†]

[1]Department of Computer and Information Science, SKL-IOTSC, University of Macau, China
[2]School of Computer Science and Information Technology, Northeast Normal University, China
[3]Department of Information Science and Technology, Dalian Maritime University, China
{mc35324, pywang}@um.edu.mo, {xiaoyn117, ymh}@nenu.edu.cn, jiangl761@dlmu.edu.cn

Abstract

In the rapidly developing field of automatic text generation and understanding, the quality of input data has been shown to be a key factor affecting the efficiency and accuracy of large language model (LLM) output. With the advent of advanced tools such as ChatGPT, input refinement work has mainly focused on prompt engineering. However, existing methods are often too dependent on specific contexts and are easily affected by individual expert experience and potential biases, limiting their wide applicability in diverse real-world applications. To address this problem, this study develops an **R**einforced **T**oken-**L**evel **I**nput **R**efinement, called RTLIR. We choose to optimize the input data at the fine-grained level of tokens, cleverly preserving the original text structure. Operationally, each state is defined by the token set of the current text, and each action is a binary decision process to decide whether to retain a specific token information. The agent automatically calculates and determines the selection probability of each token based on the current state, thereby optimizing the entire decision process. Through continuous exploration and learning, the agent can autonomously learn to identify the key inputs that have the greatest impact on the generation results and achieve refinement of the input data. In addition, RTLIR is a plug-and-play, LLM-agnostic module that can be used for a wide range of tasks and models. Experimental results show that RTLIR improves the performance of LLM in various input scenarios and tasks, with an average accuracy increase of 6%.

Code — https://github.com/Tom-HG/RTLIR.git.

Introduction

In the era of big data, large language models (LLMs) have become a core technology for processing vast amounts of textual information across sectors such as finance, healthcare, and media (Lee et al. 2024; Wu et al. 2023; Huang, Wang, and Yang 2023; Yang, Liu, and Wang 2023; Nazi and Peng 2024; Hadi et al. 2023). These models excel at extracting valuable information from complex data to support decision-making and automation tasks (Zhu et al. 2023).

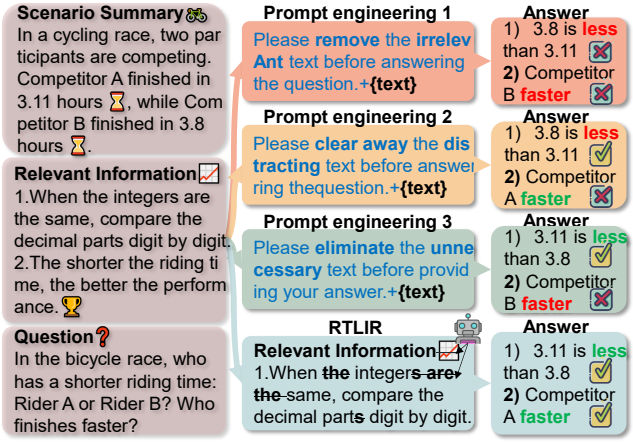


Figure 1: Examples of input refinement methods. On the left side, we present the user's input. The central section delineates the distinctions between our method and the other three prompt engineering techniques employed to refine the input. On the right side, the outputs generated by the LLM corresponding to the four methods in the central section.

However, a frequent question these models face is the presence of irrelevant information in the input data (Narayan et al. 2022; Mining 2006). This not only consumes substantial computational resources and lowers processing efficiency but can also compromise the quality and reliability of the output (Shi et al. 2023; Chen et al. 2024). In applications aimed at automatic text generation and understanding, low-quality input can severely distort the generated content, thereby reducing the actual utility and value of the model (Shi et al. 2023). Therefore, effectively identifying and deleting irrelevant information in input, and optimizing the quality of input data, are crucial for improving the performance and practicality of LLM (Petroni et al. 2019).

Since the advent of ChatGPT (Achiam et al. 2023), its powerful reasoning capabilities have made prompt engineering the focus of input refinement (Marvin et al. 2023; Liu et al. 2023; White et al. 2023). Through carefully designed prompt text, the model can effectively refine and process various inputs (Liu et al. 2023). However, despite the success of the prompting engineering in certain situations, it has

*These authors contributed equally.
†Corresponding author.

some fundamental shortcomings (Yoran et al. 2023; Zhou et al. 2023; Kojima et al. 2022). First, prompt engineering methods are often highly targeted and difficult to adapt to a wide range of real-world information inputs (Madaan et al. 2024). This limits its wide applicability in diverse application scenarios. Secondly, these methods rely heavily on the experience of different individuals and experts, which may not only lead to biased results, but also increase the threshold for use in tasks and fields (Wang, Shen, and Lim 2023). As shown in Figure 1, for the same task, different prompt engineering methods may produce very different results. These limitations highlight an urgent research challenge: How to develop an efficient and automated method to refine large-scale text data input, thereby significantly improving the processing speed and output quality of language models? We need a new approach that not only maintains understanding of the original text structure, but also automatically refines key information in the input, thereby transcending the limitations of traditional prompt engineering.

To address this challenge, this study developed an **R**einforced **T**oken-**L**evel **I**nput **R**efinement for LLM, named RTLIR. Specifically, RTLIR chooses to decompose the input text into the most basic constituent units, tokens, and uses the decomposed token set as the state. By performing precise refinement at the token level, RTLIR is able to drive the agent to more accurately identify and enhance key information. In the decision-making process, the agent adopts a binary choice mechanism, that is, deciding whether to keep each specific token. Each decision calculates the selection probability based on the current state and the expected output quality. This approach enables the agent to precisely control the retention or discard of information and optimize the input data. The reward mechanism is determined by the relevance of the LLM output to each word of the input and the quality of the final output. In this way, the agent can learn to identify and automatically optimize unnecessary tokens in the LLM input without manual configuration and filtering. In addition, RTLIR is plug-and-play and LLM-agnostic, which can be adapted to the needs of various tasks and models, thus broadening its potential application scope.

In summary, the main contributions of this paper are:

- We propose a reinforcement learning-based automatic input refinement method called RTLIR, which can analyze and process tokens in real time, thereby improving the quality and efficiency of the model.
- We present a plug-and-play, LLM-agnostic input refinement module that is adaptable to a variety of tasks and models, with a wide range of applications.
- Experimental validation shows that RTLIR performs significantly better than baseline methods in handling different inputs and various tasks, demonstrating its superiority in efficiency and adaptability.

LLM Input Processing

The input for each LLM is descriptive text $\mathbf{T}$. This text undergoes several preprocessing steps, including tokenization and removal of stop words, which transform the text into a sequence of token IDs suitable for processing by the LLM. The preprocessing transformation is define as

$$\mathbf{ID}_{\text{seq}} = \text{Preprocess}(\mathbf{T}), \tag{1}$$

where $\mathbf{ID}_{\text{seq}} = \{\text{ID}_1, \text{ID}_2, \ldots, \text{ID}_i\}$ represents the sequence of token IDs derived from $\mathbf{T}$.

These token IDs are subsequently converted into corresponding embedding vectors necessary for further processing by the LLM. This transformation is captured by the embedding function Φ_{embed}, define as

$$\mathbf{v}_{\text{seq}} = \Phi_{\text{embed}}(\mathbf{ID}_{\text{seq}}), \tag{2}$$

Where $\mathbf{v}_{\text{seq}} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_i\}$ represents the embedding vector sequence, and each $\mathbf{v}_i$ is the embedding vector corresponding to ID_i. Φ_{embed} is an embedding function that converts the token ID sequence into an embedding vector sequence. Through a conversion process $\mathbf{T} \rightarrow \mathbf{ID}_{\text{seq}} \rightarrow \mathbf{v}_{\text{seq}}$, LLM to perform tasks based on the processed input.

Problem Formulation

In the task of automatic input refinement for LLM, our goal is to selectively remove uncritical or irrelevant tag IDs from the input sequence to enhance the relevance and quality of the model output. We formalize the task as a Markov decision process (MDP) (Puterman 2014), where the model needs to maximize the information value of the output while minimizing the influence of irrelevant tag IDs. In this work, we define the key components of the MDP as

- **State** S: Each state $\mathbf{s} \in S$ represents relevant information about the text currently being processed.
- **Action** A: Each action $\mathbf{a} \in A$ includes retaining or deleting the token ID currently evaluated.
- **Transition Probability** Γ: $\Gamma(s' \mid s, a)$ represents the probability of transitioning from state s to state s' when taking action a. This probability is estimated based on past data and reflects the potential effect of the action and the response of the environment.
- **Reward** R: $R(s, a, s')$ represents the reward obtained by taking action a and transitioning from state s to state s'. The effect of the refined reward feedback input is to evaluate the effectiveness of the action.
- **Environment** E: It consists of the LLM and its training data, which determines the changes of state and reward. The dynamics of the environment are controlled by the transition probability Γ and the reward R.
- **Policy** π: The rule that determines which action to choose in a given state. The policy goal is to find the behavior sequence that optimizes the reward benefit.

Our goal is to develop a policy π^* that finds the best sequence of actions to effectively remove noise and optimize the model output. This task can be formulated as:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right], \tag{3}$$

Where γ is the discount factor, representing the present value of future rewards. t represents the step size. $\mathbb{E}$ represents the expectation reward.

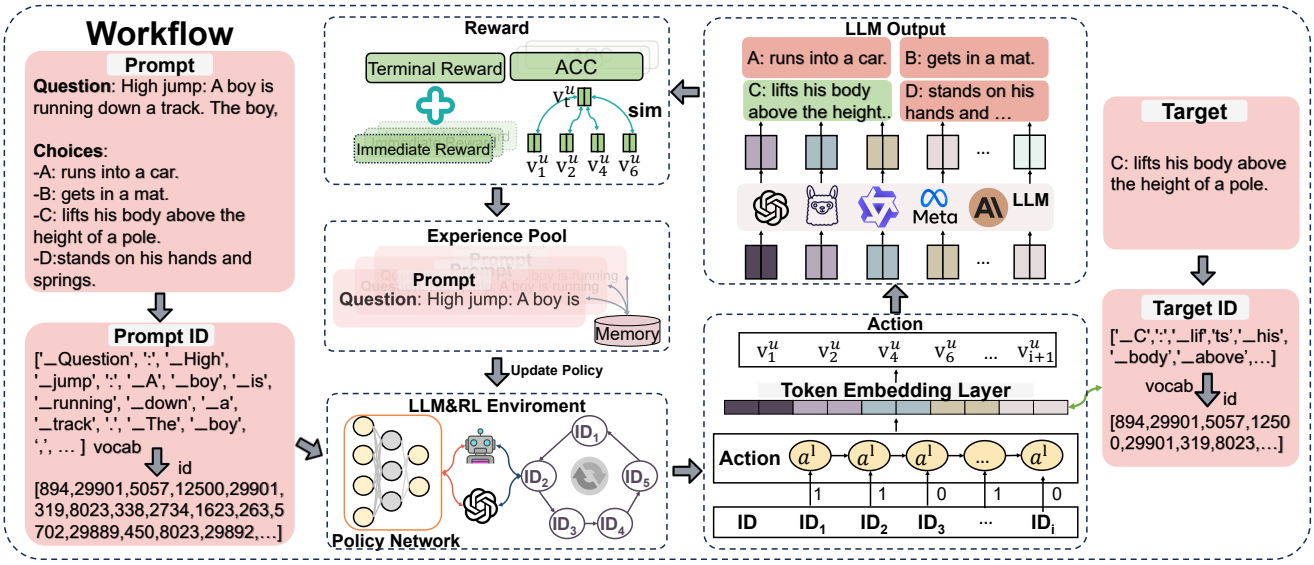


Figure 2: The RTLIR workflow starts by preprocessing the input text into token IDs and embeddings, establishing the initial state for decision-making. The reinforcement learning agent then iteratively refines the input by evaluating and adjusting each token's retention, leading to an optimized and cleaned text output.

Automatic Input Refinement Driven by Reinforcement Learning

In this section, we will explore in depth the design of RTLIR states, actions, and rewards, which is the basis for achieving efficient automatic input refinement.

Workflow of RTLIR

As shown in Figure 2, the RTLIR framework first preprocesses the input text and converts the text into token IDs and corresponding embedding vectors. This step establishes the initial state for the subsequent decision-making process, which consists of token IDs and their embeddings. Subsequently, the reinforcement learning agent decides to keep or delete the action based on the current state by evaluating the immediate and terminal rewards of each token. The immediate reward focuses on the processing effect of a single token, while the terminal reward evaluates the input refinement result of the entire text. In this way, the agent continuously learns and updates its policy to optimize the input refinement process and improve the relevance and quality of the output text. The entire process finally outputs the cleaned text, achieving effective input refinement of the data.

State & Action Design

In RTLIR, the state encapsulates the current context of the text being processed and is crucial for effectively applying input refinement strategies. The state s integrates the embedding of a token and its corresponding action, defined as

$$s = \text{CONCATENATE} \begin{bmatrix} \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_i \\ a_1, a_2, \ldots, a_i \end{bmatrix}, \tag{4}$$

Where $\mathbf{v}_i$ is the embedding vector of the i-th token derived from the embedding layer of the LLM.

a_i is a binary flag indicating the action decision for the i-th token, where 1 means keep and 0 means remove. At model startup, all actions are set to 1 by default, indicating that all tokens are retained by default, defined as

$$a_1 = a_2 = \ldots = a_i = 1, \tag{5}$$

Where $a_i = 1$ to ensure that no potentially important information is lost at the beginning of the learning process.

This design enables RTLIR to incorporate semantic and behavioral context in each decision step, thereby achieving more precise input refinement policy adjustment to adapt to the diversity of text content and task requirements.

Reward Design

The reward function is one of the key factors that determine the learning effect in reinforcement learning. We designed two rewards: immediate reward and terminal reward to measure the impact of keeping or removing operations and guide the model to optimize the quality of input data.

Immediate reward. The immediate reward is calculated immediately after each operation. It quantifies the direct effect of a decision by calculating the cosine similarity between the embedding vector $\mathbf{v}_i$ of the current token and the target embedding vector $\overline{\mathbf{v}}$, define as

$$r_i(s_i, a_i) = \textbf{sim}(\mathbf{v}_i \mid \overline{\mathbf{v}}) - \beta, \tag{6}$$

Where β is the threshold, which is used as an adjustment to balance the impact of similarity, making the model more cautious in retaining or removing decisions. This design ensures that the model can make immediate feedback adjustments based on the current similarity with the target.

Terminal reward. The terminal reward is used to evaluate the overall effect after the entire text processing process

is completed. Unlike the immediate reward, which focuses on the direct impact of a single operation, the terminal reward considers the change in information quality before and after input refinement the entire text sequence. If the input refinement operation makes the result close to the optimal solution, even a small improvement should be given a relatively high reward to encourage the model to make subtle adjustments to approach the best performance, defined as

$$r_t(s_t, a_t) = \frac{\log p(\hat{\mathbf{v}}_{\text{seq}}, a_t) - \log p(\mathbf{v}_{\text{seq}}, a_t)}{\log p(\mathbf{v}^*_{\text{seq}}, a_t) - \log p(\mathbf{v}_{\text{seq}}, a_t) + 1}, \quad (7)$$

Where $\mathbf{v}_{\text{seq}}, \hat{\mathbf{v}}_{\text{seq}}, \mathbf{v}^*_{\text{seq}}$ respectively represent the original, input refinement, and optimal input refinement text sequence embedding. It encourages the agent to consider the quality of the overall text when input refinement, and also the search for the processing path that is closest to the ideal state.

Special Case Discussion. When designing rewards, we considered several special cases:

- **Full text removal case:** If the model's operation result is $\hat{\mathbf{v}}_{\text{seq}} = \phi$, that is, all text content is removed, we retain the original text information by default. We tried many experiments, and this situation is extremely rare.
- **Original text optimal case:** If the original text can be output correctly, the normal operation should not involve any intervention. In this case, we still perform input refinement operations to extract valuable learning information from each input refinement attempt and further optimize the model's adaptability to specific tasks.

This combination of immediate and terminal rewards enables the model to optimize the actions of each decision and improve the entire text processing input refinement pipeline to adapt to different input conditions and task requirements.

LLM Input Refinement Training

In this section, we will introduce how to further optimize the input refinement ability of the RTLIR framework through policy learning to ensure that the model can make the best decisions in various text processing scenarios.

Value Estimation for Input

In RTLIR, the optimization input refinement process is achieved by learning an effective action policy π. The goal is to screen the token most relevant to the answer given the current input token id as the state, and select the best action that maximizes the expected total reward. We adopt a value-based Q learning method to estimate the expected utility of each possible action. In order to accurately evaluate the potential value Q value of each action, define as

$$Q(s, a; \theta) = \mathbb{E}\left[r_t + \gamma \max_{a'} Q(s', a'; \theta) \mid s, a\right], \quad (8)$$

Where θ represents the network parameters, r_t is the reward, and γ is the discount factor, which represents the current value of future rewards. a' represents one of the possible actions to be taken in the next state s'.

In addition, in order to separate the intrinsic value of the state from the added value of a specific action, we adopt a double learning network and use value decomposition to improve the estimation of the Q function, define as

$$Q(s, a) = V(s) + \mathcal{A}(s, a) - \frac{1}{|\mathcal{A}|} \sum_{a'} \mathcal{A}(s, a'), \quad (9)$$

Where $V(s)$ represents the best performance that can be achieved under state s without considering the specific action. $\mathcal{A}(s, a)$ represents the additional value of taking action a. This approach helps the model balance the choice between retaining key and removing noise information.

Prioritized Experience Replay with Input Refinement Policy

When processing different task inputs, it is necessary to respond immediately to guide the model to learn the most effective input refinement policy. To this end, the RTLIR framework adopts a priority experience replay mechanism. This mechanism optimizes the storage and management process of experience data in the input refinement policy, allowing the model to learn from key experiences faster. All interaction data is stored in the form of tuples, defined as

$$(s_t, a_t, r_t, s_{t+1}), \quad (10)$$

Where each tuple represents a complete transition process, including the starting state s_t, the action taken a_t, the reward obtained r_t, and the result state s_{t+1}.

In addition, in order to effectively select important experiences for accelerated learning, we adopt a priority experience replay mechanism. In this mechanism, each experience tuple is prioritized according to the absolute value of its time difference (TD) (Sutton 1988) error. TD error is a key indicator for evaluating the importance of experience, and the priority of each experience is defined as

$$\mathbf{p} = |r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)| + \epsilon, \quad (11)$$

where $\mathbf{p}$ is the priority of storing experience, and ϵ is an extremely small positive constant that ensures that each stored experience has a non-zero probability of being reviewed. This setting prevents any potentially useful experience from being ignored during the learning process.

RTLIR focuses on those experiences with the greatest expected learning value and the tokens that are most relevant to the correct output of LLM, thereby accelerating the learning process of key information and more effectively adapting to complex input refinement tasks.

Gradient Descent

We use the gradient descent method to continuously adjust the network parameters θ according to the mean square error (MSE) (James and Stein 1992) between the predicted Q value and the target Q value, so that the model outputs the result with the maximum total reward, that is, the optimal output result can be produced after the input text is refined. The loss function is defined as

$$\begin{aligned} L(\theta) &= \mathbb{E}\left[(y_t - Q(s_t, a_t; \theta))^2\right] \\ y_t &= r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-), \end{aligned} \quad (12)$$

Setting	Qwen2-1.5B		Gemma-2B		Llama-2-7B		vicuna-7B-v1.3		Llama-3-8B	
	Origin	RTLIR	Origin	RTLIR	Origin	RTLIR	Origin	RTLIR	Origin	RTLIR
English	37.60%	**42.75%**	35.15%	**50.15%**	31.25%	**45.35%**	32.25%	**48.00%**	31.95%	**36.80%**
	5.15 ± 1.09 % ↑		15.00 ± 1.50 % ↑		14.10 ± 1.05 % ↑		15.75 ± 1.06 % ↑		4.85 ± 1.06 % ↑	
Chinese	44.10%	**51.55%**	51.55%	**51.75%**	39.70%	**55.30%**	49.75%	**55.30%**	40.55%	**49.50%**
	7.45 ± 1.11 % ↑		0.20± 1.12 % ↑		15.60 ± 1.09% ↑		5.55 ± 1.11 % ↑		9.05 ± 1.11 % ↑	
Spanish	37.00%	**42.75%**	40.20%	**46.90%**	35.70%	**45.35%**	40.75%	**45.35%**	34.95%	**42.55%**
	5.75 ± 1.10 % ↑		6.70 ± 1.11% ↑		9.65 ± 1.08 % ↑		4.60 ± 1.11% ↑		7.60 ± 1.09 % ↑	
French	**52.10%**	50.80%	**48.65%**	47.00%	**47.45%**	45.15%	**47.50%**	45.10%	42.75%	**47.75%**
	-1.30 ± 1.11 % ↓		-1.65 ± 1.12 % ↓		-2.30 ± 1.12 % ↓		-2.40 ± 1.11 % ↓		5.00 ± 1.11 % ↑	
German	41.05%	**42.35%**	46.30%	**51.50%**	36.50%	**55.25%**	42.95%	**55.25%**	34.00%	**44.55%**
	1.30 ± 1.10 % ↑		5.20 ± 1.12 % ↑		18.75 ± 1.08 % ↑		12.30 ± 1.08 % ↑		10.55 ± 1.08 % ↑	
Japanese	51.05%	**53.65%**	49.90%	**55.05%**	42.95%	**55.85%**	49.40%	**55.85%**	44.95%	**51.15%**
	2.65 ± 1.11 % ↑		-5.15 ± 1.11 % ↓		12.90 ± 1.11 % ↑		6.45 ± 1.11 % ↑		6.20 ± 1.11 % ↑	
korean	47.15%	**48.90%**	52.95%	**53.10%**	46.65%	**55.20%**	51.55%	**54.85%**	47.55%	**49.80%**
	1.75 ± 1.12 % ↑		0.15 ± 1.12% ↑		8.55 ± 1.12 % ↑		3.30 ± 1.12 % ↑		2.25 ± 1.12 % ↑	
Average	44.29%	**47.54%**	47.15%	**49.66%**	40.03%	**51.06%**	44.88%	**51.39%**	39.53%	**46.01%**
	3.25 ± 0.42% ↑		2.51 ± 0.42% ↑		11.03 ± 0.41% ↑		6.51 ± 0.41 % ↑		6.48 ± 0.42% ↑	

Table 1: The performance of different expressions of the same sentence (in different languages).

Where θ^- is the parameter of the target network, and the learned parameters are updated regularly from the main network to ensure training stability.

Optimization

Through continuous evaluation and adjustment during training, we optimize the decision policy π to predict the action that maximizes the Q value, define as

$$\pi^*(s) = \arg\max_a Q(s, a; \theta), \tag{13}$$

Where this process ensures that each decision π selects the best set of token ids based on the policy.

Experiments

To comprehensively evaluate the effectiveness of RTLIR in improving LLM performance through input refinement. We designed a series of experiments to explore its performance in various text scenarios. The main goal of the experiments is to answer the following key research questions:

Q1: How does RTLIR perform when processing different inputs of the same problem?

Q2: How adaptable and performant is RTLIR in different types of language processing tasks?

Q3: What is the specific impact of reward design on the performance and decision-making process of RTLIR?

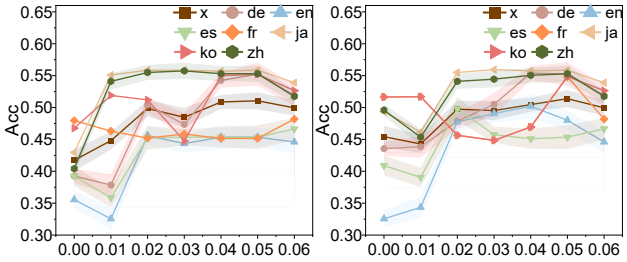
Q4: What types of token IDs does RTLIR remove in actual applications? What is the basis for its decision?

Experimental Settings

Datasets. The experimental data comes from two parts: The first part is used to verify input diversity. The dataset used is PAWS-X (Zhang, Baldridge, and He 2019; Yang et al. 2019), which includes 23,659 manually translated PAWS evaluation pairs and 296,406 machine translated training pairs, covering six different types of languages: French, Spanish, German, Chinese, Japanese and Korean. It aims to test the stability and effectiveness of the model when dealing with inputs with high semantic similarity but different surface forms. The second part is used to verify task diversity. The data covers a variety of different types of tasks. These tasks are all from the Language Model Evaluation Harness (Gao et al. 2021) - the backend of Hugging Face's popular Open

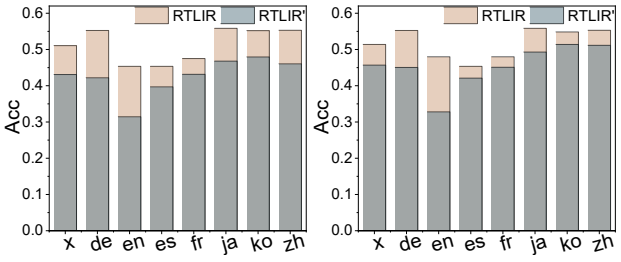
Dataset	Metric	Llama-2			
		Origin		RTLIR	
		Value	±Std.	Value	±Std.
Ancient Chinese	Acc	39.52	1.51	**39.71**	1.51
Chinese Civil Service	Acc	24.37	3.40	**25.62**	4.12
Chinese Foreign Policy	Acc	28.04	4.36	**28.97**	2.48
College Actuarial Science	Acc	31.13	4.52	**32.08**	4.56
Computer Science	Acc	26.47	3.10	26.47	3.10
Computer Security	Acc	25.73	3.35	**26.32**	3.38
Conceptual Physics	Acc	25.85	3.62	**26.53**	3.65
Machine Learning	Acc	23.77	3.87	23.77	3.87
Professional Accounting	Acc	23.43	3.21	23.43	3.21
CoLA	mcc	-2.33	2.88	**0.58**	3.12
MRPC	F1	81.52	1.62	**81.71**	1.62
QQP	F1	53.35	0.26	**53.36**	0.26
WNLI	Acc	45.07	5.95	**47.89**	5.97
SOCIAL IQA	Acc	46.06	1.13	**46.16**	1.13
KorMedMCQA Doctor	EM	5.96	1.41	**6.32**	1.44
RACE	Acc	39.52	1.51	**39.71**	1.51

Table 2: The performance of RTLIR on various tasks.



(a) Llama-2 (b) Vicuna

Figure 3: An ablation study of Immediate Rewards.



(a) Llama-2 (b) Vicuna

Figure 4: An ablation study of Terminal Rewards.

LLM Leaderboard, which has been used in hundreds of papers and is used internally by dozens of organizations such as NVIDIA, Cohere, BigCode, and Mosaic ML. It aims to evaluate the generalization ability and performance of the model when dealing with different types of tasks.

Baseline Methods. To comprehensively evaluate the performance of RTLIR, we compared its results with those of mainstream LLM models before and after input refinement. These models include Qwen2-1.5B (Yang et al. 2024), Gemma-2B (Team et al. 2024), Llama-2-7B (Touvron et al. 2023), Llama-3-8B (Dubey et al. 2024), and Vicuna-7B-v1.3 (Zheng et al. 2024). More details in Appendix.

Evaluation Metrics. To comprehensively evaluate the improvement of RTLIR on LLM performance before and after input refinement, we use accuracy (ACC), Matthews correlation coefficient (MCC), F1 score and exact match (EM) as core evaluation metrics. All evaluation results are calculated based on the lm-eval-harness library (Gao et al. 2021). These metrics intuitively demonstrate the actual improvement of input refinement on model performance by comparing the consistency between model prediction results and true labels. For details, please see the Appendix.

The Study of Input Diversity (Q1)

Table 1 shows the performance indicators before and after the combination of RTLIR and LLM. For inputs with the same meaning in different languages, the higher accuracy indicates that the semantic quality of the model input has been significantly improved. The experimental results show that the method using RTLIR outperforms the original model in all aspects on the PAWS-X dataset. Specifically, RTLIR improves the accuracy of Qwen2, Gemma, Llama-2, Llama-3, and Vicuna by 3.25%, 2.15%, 11.03%, 6.51%, and 6.48%, respectively. Among the seven languages including English, RTLIR mostly achieves good performance except French (only llama3 can achieve good performance). The analysis found that this is due to the significant differences between French language structure and expressions and other languages. After the model has a deeper semantic understanding and perception capabilities, the effect of RTLIR is also significant. Overall, RTLIR significantly improves the quality of the processing results by effectively removing noise from the input. This not only enhances the adaptability of the model to different language variants but also improves the accuracy and reliability of information processing.

The Study of Task Diversity (Q2)

Table 2 shows the performance of different LLMs on various tasks before and after applying the RTLIR input refinement policy. We present 16 datasets with varying task types and present results using multiple evaluation metrics. More detailed indicators and results of different LLMs can be found in the appendix, and all data will be published simultaneously on Hugging Face for community sharing and verification. Experimental results show that RTLIR significantly improves multiple performance indicators of LLM and shows high stability. Especially in 14 datasets, RTLIR + Llama-2 outperforms the original model. These results further demonstrate the versatility of RTLIR in input refinement and improving model performance. It shows that it not only performs well in different task scenarios, but also significantly improves the accuracy and reliability of the model.

The Study of Reward Design (Q3)

The Design of Immediate Rewards. Immediate rewards are evaluated based on the immediate results of a single action being performed. We tested the impact of similarity threshold β from 0 to 0.06 on different datasets to optimize the similarity between the model output and the target. Figure 3 shows that the threshold β has a significant impact on the model input refinement efficiency. When the threshold β is set to 0.05, the model performs best and the accuracy increases by 10% to 20%. However, thresholds that are too high or too low can lead to performance degradation, indicating that setting the immediate reward appropriately is crucial to maintaining a balanced model performance.

The Design of Terminal Rewards. The terminal reward is used to evaluate the overall effect of the entire input refining process and is calculated based on the change in model output performance before and after input refining. We designed a variant RTLIR' that does not include terminal rewards. As shown in Figure 4, after introducing terminal rewards on different data sets, the overall quality of model output is significantly improved, and the average accuracy rate is increased by 10%. This result shows that terminal rewards can not only effectively guide the model to optimize text processing strategies, but also significantly improve the accuracy and consistency of the final output, ensuring better output results in different scenarios.

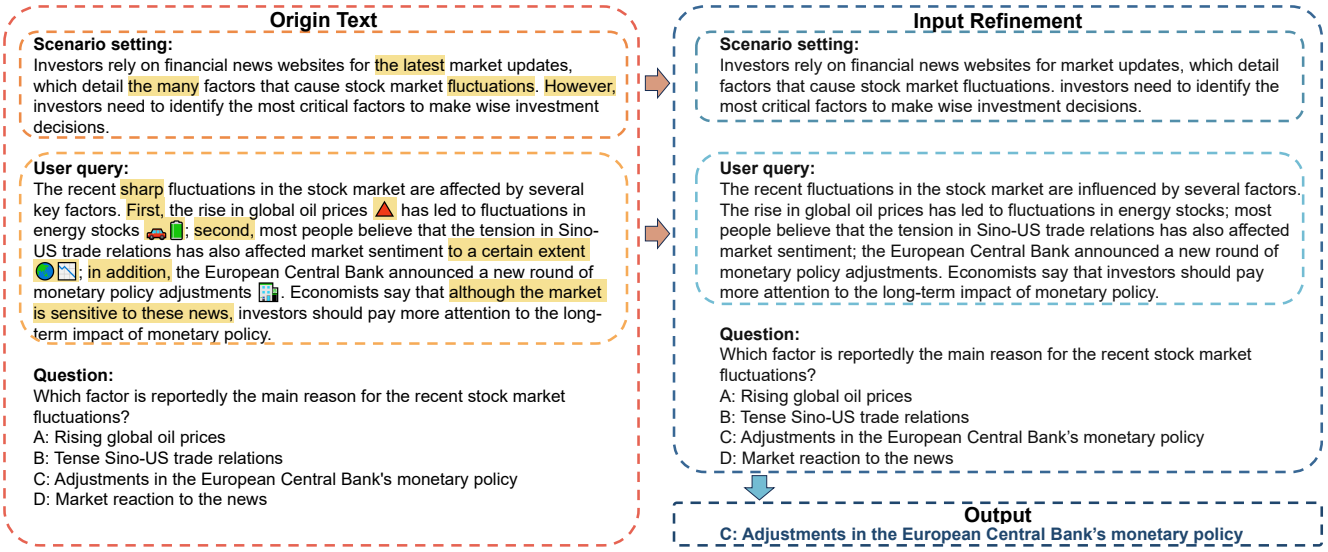


Figure 5: Case studies of RTLIR.

Case Study (Q4)

To demonstrate in detail how RTLIR can effectively refine input automatically in real applications, we show an example of a question-answering task in Figure 5 . During the input refinement process, the following unnecessary text or noise was removed: (1) Redundant Descriptions: The word "sharp" and the phrase "although the market is sensitive to these news" were removed. (2) Repetitive Information: The word "latest" was removed since "market updates" inherently implies the most recent information. (3) Unnecessary Details: The word "many" was removed because "factors" already indicates a variety of influences. (4) Emojis: Emojis were removed as they add a casual tone that is not necessary in a formal text. The refinements made the text concise, retaining essential information and aiding reader comprehension. This example shows RTLIR's capacity to boost information processing accuracy and efficiency, and its practical utility in enhancing LLM input refinement.

Related Work

Large Language Model Input Refinement

Large Language Model (LLM) input refinement aims to improve model performance by filtering and removing model inputs to improve input quality (Mining 2006; Zhang et al. 2023; Narayan et al. 2022). With the rise of chatbots (such as ChatGPT), people began to try to set different prompts to let the model refine the input itself (Yoran et al. 2023; Zhou et al. 2023; Kojima et al. 2022). RePrompt aims to improve the accuracy of emotional expression in AI artwork generated from text prompts (Wang, Shen, and Lim 2023). Data-Juicer build a system that can efficiently generate diverse data recipes, explore different data mixing possibilities, and evaluate their impact on model performance (Chen et al. 2024). However, these methods are often highly targeted and rely heavily on the experience of different individuals and experts, which may not only lead to biased results but also raise the threshold for use in tasks and fields.

Reinforcement Learning

Reinforcement learning (RL) optimizes an agent's behavior to maximize rewards (Kaelbling, Littman, and Moore 1996; Li 2017; Wiering and Van Otterlo 2012) by iterative experimentation in the environment, and its ability to adapt in dynamic environments makes it an ideal tool for dealing with complex problems (Zhang et al. 2024; François-Lavet et al. 2018). RL is able to dynamically adjust the input refinement policy based on immediate feedback from the environment, providing a more flexible and efficient solution than traditional methods (Liang, Lin, and Ma 2023; Szepesvári 2022). In contrast, traditional input refinement methods usually rely on preset input refinement rules and lack real-time feedback and adaptive adjustment mechanisms. (Zhang et al. 2021; Tong, Wang, and Niu 2023). However, there is little research on using RL to refine the LLM input, and there is currently no common method or benchmark.

Conclusion

In this study develops RTLIR, an automatic input refinement framework based on large language models (LLM). Facing the challenge of constantly changing input patterns, we adopt a reinforcement learning paradigm to refine the input at the token-level. This approach enables RTLIR to achieve efficient automatic input refinement operations through continuous interactive learning, and finally obtain the optimal input refinement results. RTLIR can not only adapt to various models and data, but also can be used as a module, plug and play. Through comprehensive experiments, we verify the effectiveness of RTLIR and demonstrate its strong potential in the field of automatic input refinement.

Acknowledgments

This research is funded by the Science and Technology Development Fund (FDCT), Macau SAR (file no. 0123/2023/RIA2, 001/2024/SKL).

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Chen, D.; Huang, Y.; Ma, Z.; Chen, H.; Pan, X.; Ge, C.; Gao, D.; Xie, Y.; Liu, Z.; Gao, J.; et al. 2024. Data-juicer: A one-stop data processing system for large language models. In *Companion of the 2024 International Conference on Management of Data*, 120–134.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M. G.; Pineau, J.; et al. 2018. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4): 219–354.

Gao, L.; Tow, J.; Biderman, S.; Black, S.; DiPofi, A.; Foster, C.; Golding, L.; Hsu, J.; McDonell, K.; Muennighoff, N.; et al. 2021. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, 10: 8–9.

Hadi, M. U.; Qureshi, R.; Shah, A.; Irfan, M.; Zafar, A.; Shaikh, M. B.; Akhtar, N.; Wu, J.; Mirjalili, S.; et al. 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*.

Huang, A. H.; Wang, H.; and Yang, Y. 2023. FinBERT: A large language model for extracting information from financial text. *Contemporary Accounting Research*, 40(2): 806–841.

James, W.; and Stein, C. 1992. Estimation with quadratic loss. In *Breakthroughs in statistics: Foundations and basic theory*, 443–460. Springer.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4: 237–285.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213.

Lee, J.; Stevens, N.; Han, S. C.; and Song, M. 2024. A survey of large language models in finance (finllms). *arXiv preprint arXiv:2402.02315*.

Li, Y. 2017. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Liang, C.; Lin, H.; and Ma, H. 2023. Reinforcement learning-based denoising model for seismic random noise attenuation. *IEEE Transactions on Geoscience and Remote Sensing*, 61: 1–17.

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9): 1–35.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Marvin, G.; Hellen, N.; Jjingo, D.; and Nakatumba-Nabende, J. 2023. Prompt engineering in large language models. In *International conference on data intelligence and cognitive informatics*, 387–402. Springer.

Mining, W. I. D. 2006. Data mining: Concepts and techniques. *Morgan Kaufinann*, 10(559-569): 4.

Narayan, A.; Chami, I.; Orr, L.; Arora, S.; and Ré, C. 2022. Can foundation models wrangle your data? *arXiv preprint arXiv:2205.09911*.

Nazi, Z. A.; and Peng, W. 2024. Large language models in healthcare and medical domain: A review. In *Informatics*, volume 11, 57. MDPI.

Petroni, F.; Rocktäschel, T.; Lewis, P.; Bakhtin, A.; Wu, Y.; Miller, A. H.; and Riedel, S. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Shi, F.; Chen, X.; Misra, K.; Scales, N.; Dohan, D.; Chi, E. H.; Schärli, N.; and Zhou, D. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, 31210–31227. PMLR.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3: 9–44.

Szepesvári, C. 2022. *Algorithms for reinforcement learning*. Springer nature.

Team, G.; Riviere, M.; Pathak, S.; Sessa, P. G.; Hardin, C.; Bhupatiraju, S.; Hussenot, L.; Mesnard, T.; Shahriari, B.; Ramé, A.; et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Tong, X.; Wang, P.; and Niu, S. 2023. Reinforcement learning-based denoising network for sequential recommendation. *Applied Intelligence*, 53(2): 1324–1335.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wang, Y.; Shen, S.; and Lim, B. Y. 2023. Reprompt: Automatic prompt editing to refine ai-generative art towards precise expressions. In *Proceedings of the 2023 CHI conference on human factors in computing systems*, 1–29.

White, J.; Fu, Q.; Hays, S.; Sandborn, M.; Olea, C.; Gilbert, H.; Elnashar, A.; Spencer-Smith, J.; and Schmidt, D. C. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.

Wiering, M. A.; and Van Otterlo, M. 2012. Reinforcement learning. *Adaptation, learning, and optimization*, 12(3): 729.

Wu, S.; Irsoy, O.; Lu, S.; Dabravolski, V.; Dredze, M.; Gehrmann, S.; Kambadur, P.; Rosenberg, D.; and Mann, G. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Yang, H.; Liu, X.-Y.; and Wang, C. D. 2023. Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*.

Yang, Y.; Zhang, Y.; Tar, C.; and Baldridge, J. 2019. PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification. In *Proc. of EMNLP*.

Yoran, O.; Wolfson, T.; Ram, O.; and Berant, J. 2023. Making retrieval-augmented language models robust to irrelevant context. *arXiv preprint arXiv:2310.01558*.

Zhang, H.; Dong, Y.; Xiao, C.; and Oyamada, M. 2023. Large language models as data preprocessors. *arXiv preprint arXiv:2308.16361*.

Zhang, R.; Zhu, J.; Zha, Z.; Dauwels, J.; and Wen, B. 2021. R3l: Connecting deep reinforcement learning to recurrent neural networks for image denoising via residual recovery. In *2021 IEEE International Conference on Image Processing (ICIP)*, 1624–1628. IEEE.

Zhang, Y.; Baldridge, J.; and He, L. 2019. PAWS: Paraphrase Adversaries from Word Scrambling. In *Proc. of NAACL*.

Zhang, Z.; Xiao, Y.; Jiang, L.; Yang, D.; Yin, M.; and Wang, P. 2024. Spatial-Temporal Interplay in Human Mobility: A Hierarchical Reinforcement Learning Approach with Hypergraph Representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 9396–9404.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Zhou, W.; Zhang, S.; Poon, H.; and Chen, M. 2023. Context-faithful prompting for large language models. *arXiv preprint arXiv:2303.11315*.

Zhu, Y.; Yuan, H.; Wang, S.; Liu, J.; Liu, W.; Deng, C.; Dou, Z.; and Wen, J.-R. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*.