Reinforced Retrieval to Avoid Hallucination in Large Language Models

Zhi-Xuan Tai
Institute of Electrical and Computer Engineering National Yang Ming Chiao Tung University
Hsinchu, Taiwan
will0010077.ee11@nycu.edu.tw

Liang-Hsuan Tai
Electronic and Optoelectronic System Research Labs Industrial Technology Research Institute
Hsinchu, Taiwan
Sean_Tai@itri.org.tw

Jen-Tzung Chien
Institute of Electrical and Computer Engineering National Yang Ming Chiao Tung University
Hsinchu, Taiwan
jtchien@nycu.edu.tw

***Abstract*—Large language models, while powerful, but likely produce factually incorrect or irrelevant content which results in an issue of hallucination. This issue is critical in those target domains that require high accuracy. This paper presents a reinforced retrieval framework that implements and combines a reinforcement learner in the retrieval-augmented generation to dynamically verify and optimize quality, consistency and completion for content generation. By balancing the fluency and accuracy through a tailored reward function, this method is potentially useful to mitigate hallucination. Experimental results on two benchmark datasets show that the proposed method outperforms the existing models, particularly in both long-form and short-form question-answering tasks.**

***Index Terms*—Large language model, retrieval-augmented generation, de-hallucination, reinforcement learning**

I. Introduction

Natural language generation (NLG) has become a key role in modern AI, powering various applications, e.g. automated content creation, chatbots, and text summarization. Transformer-based models such as GPT-2 [1], GPT-3 [2], and T5 [3] have significantly upgraded the machines to generate coherent and contextually relevant sentences. However, a critical challenge remains, causing the hallucination where the model produces the content that is factually incorrect or irrelevant. This issue arises because large language models (LLMs) focus on predicting the next word in a sequence based on the learned patterns, which can lead to the generation of plausible-sounding but inaccurate information. In the domains where the generation accuracy is crucial, such as the healthcare or legal services, this problem undermines the reliability of LLM systems. To address this problem, a powerful method, called the retrieval-augmented generation (RAG) [4], was developed by grounding the generated content through a retrieval of those relevant information from an external knowledge base during the text generation process.

This paper aims to strengthen the hallucination mitigation by integrating reinforcement learning (RL) with RAG, building a "reinforced retrieval" system. This system actively retrieves and verifies the information during generation, ensuring that the output is both fluent and factually accurate. By optimizing various reward measures within the RL framework, this solution balances between fluency and accuracy, guiding the model to produce trustworthy information. The goal of this study aims to train a robust LLM that considerably reduces hallucination for those critical domains where trustworthiness is essential. Some experiments are illustrated.

II. Background Survey

Hallucination in NLG is seen as a safety issue in many domains such as healthcare, law and education. It is crucial to assure reliability and factual accuracy in sentence generation.

A. Retrieval-augmented generation

An effective method to mitigate hallucination involved fine-tuning the pre-trained models on a domain-specific dataset, improving its ability to produce the accurate outputs in particular contexts. For instance, the model like InstructGPT [5] has been fine-tuned by using the manually labeled dialogue samples, aligning them closely with human preferences and reducing the likelihood of hallucination. Despite these efforts, the problem persists, especially in a scenario requiring up-to-date knowledge. To further reduce the hallucination, RAG has emerged as a promising solution. RAG basically enhances the generation process by incorporating the relevant information retrieved from the external knowledge base. In [6], a method called Self-RAG utilized the LLM-produced special tokens to retrieve, generate, and score the outputs actively. This method introduced a self-reflect concept so that LLMs could judge their own outputs. In [7], an attention mechanism was designed by feeding those retrieved documents into LLM. This mechanism allows the model to ground its output with verifiable data, reducing the reliance on the model's internal knowledge, which may be outdated or incomplete.

B. Reinforced language modeling

Reinforcement learning has also been explored as a means of improving the accuracy and the contextual relevance in the LLM-generated outputs [8], [9]. The technique like proximal policy optimization (PPO) [10] has been employed to fine-tune a model by optimizing a reward function that balances the trade-off between text fluency and factual accuracy. Such an

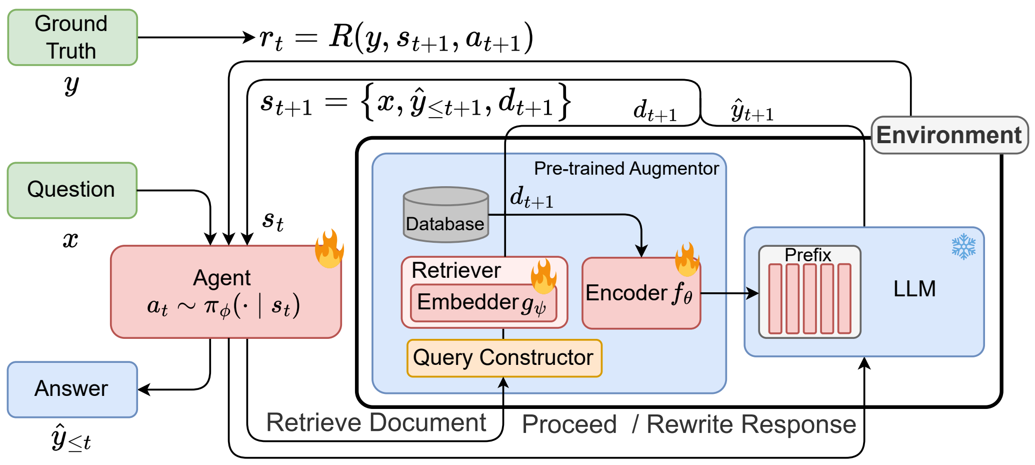

Fig. 1. Illustration of the reinforced retrieval system. This system integrates RAG process with RL. The agent interacts with the environment, utilizing the retrieval and the pre-trained knowledge encoder combined with LLM to dynamically retrieve the documents to generate the contextually informed responses. The agent's actions, including retrieving, proceeding, or rewriting, are guided by feedback from the environment, optimizing RAG performance through continuous learning.

RL framework guided the model in generating the responses that meet human expectations, thereby reducing the hallucination. Traditional RAG systems often retrieved documents only once, limiting their ability to adapt to new information especially when the generation process unfolded dynamically. Building on the foundations of traditional RAG and RL methods, in this study, modern approaches are explored via modular and adaptive RAG systems. The proposed systems can adjust the retrieval strategies based on the evolving context of the generated sentences, ensuring that the information used is both relevant and up-to-dated.

III. Reinforced Retrieval

In this study, the retrieval-augmented generation (RAG) is implemented with a reinforced learner (RL) to capture the dynamics of the retrieved documents in the refined sentence generation. This approach encapsulates the RAG process within an environment where the agent can interact and make decision at each step. Such an agent take different actions during the RAG learning process, including *retrieve*, *proceed*, and *rewrite*. The reward signal calculated at each time step is accumulated to train the agent through RL, allowing the system to dynamically adapt its mixed retrieval-generation strategy. Figure 1 illustrates such a system, demonstrating how the integration of RAG with RL. The RAG environment includes a pre-training stage to ensure that system can effectively handle complex user queries across a wide range of domains by integrating document retrieval, knowledge augmentation, and LLM training. Frozen and trainable components are shown.

A. Dynamic Retrieval-Augmented Generation

The complete response is run by a multi-step generation process $\hat{y} = \{\hat{y}_n\}_{n=1}^N$ where $\hat{y}_n$ is a segment in the final response with length N. Additional retrieval or rewriting can be applied during this generation process. In each time step t, this dynamic system may perform the following actions including

- generate a segment of response $\hat{y}_{t+1}$,
- rewrite the previous response by removing $\hat{y}_t$, and generate $\hat{y}_{t+1}$ to replace it,
- retrieve the relevant document d_{t+1} to enhance the following generation process.

Here, t is the generation time step. Since the action "retrieval" does not produce a new response and "rewriting" may remove an existing segment $\hat{y}_t$, the time step t can exceed N.

B. Information Retriever

The training of the retriever follows LexMAE [11]. This work introduces a novel pre-training framework known as the lexicon-bottlenecked masked autoencoder, which addresses the gap between language modeling and lexicon-weighting retrieval by learning the importance-aware lexicon representations, enabling the pre-trained model to capture the significance of different words within the lexicon space. At the fine-tuning stage, the inner product between query embedding $g_\psi(q)$ and document embedding $g_\psi(d_+)$ is maximized with respect to ψ, ensuring efficient retrieval of relevant documents. The retriever is guided by a contrastive loss

$$\mathcal{L}(\psi) = -\log \frac{\exp(g_\psi(q)^\top g_\psi(d_+))}{\sum_{d_- \in B} \exp(g_\psi(q)^\top g_\psi(d_-))} \quad (1)$$

where the denominator computes the sum of similarities between a query embedding and its negative document embeddings $g_\psi(d_-)$ in a minibatch, i.e. $d_- \in B$. Minimizing this loss enables a dynamic retrieval through maximizing the inner product between query and document.

C. Knowledge Encoder

The knowledge encoder with parameter θ, inspired by LLaMA adapter [12], represents the retrieved documents to generate the prefixes $f_\theta(d)$ that enhance the LLM training. A supervised fine-tuning is performed by minimizing the negative log-likelihood of true segment y_t given an input x, previous segments $y_{<t}$ and retrieved document d to ensure that the encoder maps the knowledge from document d to response y

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \log p(y_t | y_{<t}, x, f_\theta(d)). \tag{2}$$

D. State Representation

Accurately representing the current state at each step is crucial for an agent to make an informed decision during the reinforced retrieval. This study represents each state consisting of four parts separated by special token [SEP] including the action history y_n, which captures the sequence of actions taken by the agent, the current query x and previous responses $\hat{y}_{\leq t}$, which help the agent assess the relevance and completeness of its outputs, and the retrieved documents $f_\theta(d_t)$, which provide the necessary context for determining whether the information remains pertinent or requires updating. The token-based state representation is shown in a form of

[Action history][SEP][Question][SEP][Previous response]
[SEP][Retrieved documents].

E. Agent Action

In a traditional RAG system, documents are typically retrieved in a single step based on the initial user query, which considerably constrains the flexibility in response generation. To deal with this issue, the RAG process should be adaptive and could be simulated as an RL problem where the agent can choose from three distinct actions throughout the whole generation process, which include

Retrieve (RT) allows the agent to discard previously retrieved documents and form a new query with the current user question and the responses generated so far. The query is concatenated as $q_t = \text{concat}(x, \hat{y}_{\leq t})$, and the relevant documents are retrieved using $d_{t+1} = \text{ret}(q_t)$. This ensures that the retrieved information remains relevant and up-to-dated, aligning the response generation with the accurate context.

Proceed (PC) involves the LLM generating the next response segment $\hat{y}_{t+1}$ by using the current state. This process is represented as $\hat{y}_{t+1} \sim p(\cdot | y_n, x, \hat{y}_{\leq t}, f_\theta(d_t))$.

Rewrite (RW) allows the agent to discard and regenerate a previous response if it is found unsatisfactory. The previous response $\hat{y}_t$ is removed, and a new response is generated as $\hat{y}_{t+1} \sim p(\cdot | y_n, x, \hat{y}_{\leq t}, f_\theta(d_t))$. These actions give the adaptive RAG system the flexibility to adjust the generation process thus improving overall system performance dynamically.

F. Reward Function

The reward function is a critical component in the RL framework, substantially guiding the model to generate accurate text while minimizing the hallucination. The following measures are considered in construction of reward function.

Retrieval Quality: For each retrieval action, the quality or the relevance of the retrieved document d_t to the ground-truth response y is measured by using the metric BLEU [13]

$$r_{a,t} = \text{BLEU}(d_t, y). \tag{3}$$

Generation Quality: For each proceed (PC) or rewrite (RW) action, a generation likelihood reward is calculated by

$$r_{b,t} = \exp\left(\frac{1}{|y_n|}\sum_{i=1}^{|y_n|} \log p(y_{n,i} | y_{n,<i}, x, \hat{y}_{<t}, f_\theta(d_t))\right). \tag{4}$$

Consistency: Consistency reward is measured from the generated text, similar to generation quality reward $r_{b,t}$, as

$$r_{c,t} = \exp\left(\frac{1}{|y_n|}\sum_{i=1}^{|y_n|} \log p(\hat{y}_{t,i} | y_{n,<i}, x, \hat{y}_{<t}, f_\theta(d_t))\right). \tag{5}$$

Completion: Upon task completion, a reward can be calculated by using the BERTScore [14] between the generated response $\hat{y}$ and the ground truth y

$$r_{d,t} = \text{BERT}(\hat{y}_{\leq t}, y) - \text{BERT}(\tilde{y}, y) \tag{6}$$

where $\tilde{y} \sim p(y, x, f_\theta(d_t))$ represents a basic response sampled directly in a single step as a standard RAG response.

Therefore, the final reward function is defined by

$$r_t = \begin{cases} \frac{\alpha}{N} \cdot r_{a,t}, & \text{if } a_t \text{ is } \textit{RT} \\ \frac{\beta}{N} \cdot r_{b,t} + \frac{\gamma}{N} \cdot r_{c,t} + \frac{c}{N}, & \text{if } a_t \text{ is } \textit{PC} \\ \frac{\beta}{N} \cdot (r_{b,t} - r_{b,t-1}) \\ \quad + \frac{\gamma}{N} \cdot (r_{c,t} - r_{c,t-1}), & \text{if } a_t \text{ is } \textit{RW} \\ r_{d,t}, & \text{if step is done} \end{cases} \tag{7}$$

This design ensures that the system would generate high-quality text by guiding the agent to retrieve relevant documents, rewrite inappropriate responses, and use BERT scores to measure the final generation quality. Hyperparameters α, β, and γ are used to control the RL training, while a constant c is used to encourage the continuous generation. When rewriting the response, a trick is employed to estimate the improvements of $r_{b,t}$ and $r_{c,t}$ of current step t. The reward of each step is divided by the number of segments N to limit or normalize the total reward for each turn.

G. Reinforcement Learning for Optimization

PPO is known as an effective method for training an agent to choose actions that make the final response which aligns with human preferences, mitigating hallucinations and ensuring contextual relevance. PPO is a policy gradient method that updates the policy by taking small and stable steps to stabilize the updates during training. The algorithm optimizes the policy by maximizing a clipped surrogate objective function, ensuring that the new policy does not deviate significantly from the old policy, thereby maintaining the stability. By using the reward function described earlier to represent the training objective, PPO is used to train the agent's policy to improve the system performance in generating the dehallucinated text. The so-called reinforced retrieval is implemented.

IV. Experiments

A. Experimental Settings

During RL training, this study considerably utilized a pseudo-generation mechanism that leveraged the teacher forcing to simulate actual response generation, significantly reducing the computation time required by using LLMs. Instead of waiting for the LLM to generate each response token-by-token, this work fed the ground truth segment directly into the model and resampled the tokens from the output probability distribution. This approach mimics the real sampling process while minimizing the generation time, enabling more training iterations and faster convergence.

The reward function (7) incorporates several hyperparameters. These hyperparameters balance the contribution of various reward sources. The tuning of these hyperparameters is critical for optimizing the model's performance. We experimented with different combinations of α, β, γ, c, and N to explore the impact on reward calculation for different actions a_t. By systematically varying these parameters and evaluating the model's behavior, the reward function can be adjusted to achieve the balance between reward sources and optimize performance by encouraging different action combinations. In the experimental results, the combination with the best performance was achieved with $\alpha = 0.1$, $\beta = 0.05$, $\gamma = 0.05$, $c = 0.1$, and the number of segments $N \leq 6$, which provided an effective balance in weighting the different reward components.

B. Experimental Results

This study summarizes the performance of various methods on the Natural Question (NQ) [15] and TriviaQA [16] datasets, evaluated across multiple metrics including the Exact Match (EM), BLEU-1 (B-1), ROUGE-1 (R-1), ROUGE-2 (R-2), ROUGE-L (R-L), and BERTScore (BS) [13], [14]. The best results for each metric are highlighted in **bold**.

As shown in Table I, the proposed reinforced retrieval method demonstrates superior performance compared to the other RAG methods, particularly in terms of ROUGE and BERTScore metrics. Notably, the reinforced retrieval method improves the overall quality of generated content, as indicated by achieving higher scores across in most of the metrics.

It is important to note that the Self-RAG [6] method, while generally effective, underperforms in this evaluation. This is because the Self-RAG approach involves training on a dataset designed to improve RAG performance, however, it does not include the dataset for long-form generation tasks, namely NQ task. As a result, the generated outputs are consistently shorter than the ground truth, leading to worse performance across the evaluated metrics.

In addition, as shown in Table II, the reinforced retrieval method outperforms the other RAG methods across various metrics, particularly in short-form question-answering, hear meaning the TriviaQA task. From the result, this study suggests that the reinforced retrieval method remains robust for the tasks that require concise answers.

TABLE I
COMPARISON OF THE RSULTS ON NQ DATASET.

Method	B-1	R-1	R-2	R-L	BS
Original	**17.91**	25.84	8.75	17.39	52.64
Standard RAG	16.29	27.43	12.43	20.14	53.05
Self-RAG	4.37	18.61	7.43	14.89	46.72
Proposed	17.84	**28.96**	**13.61**	**21.26**	**54.43**

TABLE II
COMPARISON OF THE RESULTS ON TRIVIAQA DATASET.

Method	EM	B-1	R-1	R-L	BS
Original	37.50	39.76	46.48	46.48	67.06
Standard RAG	45.09	44.21	49.44	49.44	70.11
Self-RAG	27.50	28.11	31.27	31.27	62.59
Proposed	**47.52**	**47.19**	**49.81**	**49.57**	**72.14**

C. Ablation Studies

To evaluate the contribution of different components within the proposed system, we perform a series of ablation studies. Table III evaluates the performance impact due to including or excluding the specific modules. In a scenario where the knowledge encoder is removed, this paper implemented a document concatenation to apply retrieval augmentation without retraining the agent. This result demonstrates that the agent is able to transfer and adapt to different scenarios, indicating that the knowledge encoder and the agent significantly enhance the system's overall performance.

TABLE III
ABLATION STUDY ON SHOWING THE IMPACT OF DIFFERENT COMBINATIONS OF USING KNOWLEDGE ENCODER AND RETRIEVAL AGENT.

Enc.	Agent	B-1	R-1	R-2	R-L	BS
×	×	16.29	27.43	12.43	20.14	53.05
×	✓	16.69	27.73	12.92	20.87	53.50
✓	×	17.14	28.51	13.14	21.08	54.17
✓	✓	**17.84**	**28.96**	**13.61**	**21.26**	**54.43**

V. Conclusions

This paper has presented a reinforced retrieval framework that integrated RAG with RL to address the issue of hallucinations in NLG. By dynamically optimizing the retrieval and generation processes, this approach significantly improved the accuracy and contextual relevance of the generated content. Experimental results on both long-form and short-form question-answering tasks demonstrate the effectiveness of this method. Additionally, while the framework provides a new solution for deploying NLG models in the critical domains where factual accuracy is essential. There remains a challenge in tuning the hyperparameters for the reward function. Future work will focus on refining the reward structure and broadening the system applicability across diverse tasks.

References

[1] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, “Language models are unsupervised multitask learners,” *OpenAI Blog*, 2019.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[3] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[4] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Advances in Neural Information Processing Systems*, 2020.

[5] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, “Training language models to follow instructions with human feedback,” in *Advances in Neural Information Processing Systems*, vol. 35, pp. 27730–27744, 2022.

[6] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, “Self-RAG: Learning to retrieve, generate, and critique through self-reflection,” in *The International Conference on Learning Representations*, 2024.

[7] Z. Li, R. Guo, and S. Kumar, “Decoupled context processing for context augmented language modeling,” in *Advances in Neural Information Processing Systems*, 2022.

[8] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, N. Joseph, S. Kadavath, J. Kernion, T. Conerly, S. El-Showk, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, T. Hume, S. Johnston, S. Kravec, L. Lovitt, N. Nanda, C. Olsson, D. Amodei, T. Brown, J. Clark, S. McCandlish, C. Olah, B. Mann, and J. Kaplan, “Training a helpful and harmless assistant with reinforcement learning from human feedback,” *arXiv:2204.05862*, 2022.

[9] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, *et al.*, “Constitutional AI: Harmlessness from AI feedback,” *arXiv:2212.08073*, 2022.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv:1707.06347*, 2017.

[11] T. Shen, X. Geng, C. Tao, C. Xu, X. Huang, B. Jiao, L. Yang, and D. Jiang, “LexMAE: Lexicon-bottlenecked pretraining for large-scale retrieval,” in *The International Conference on Learning Representations*, 2023.

[12] R. Zhang, J. Han, C. Liu, A. Zhou, P. Lu, Y. Qiao, H. Li, and P. Gao, “LLaMA-adapter: Efficient fine-tuning of large language models with zero-initialized attention,” in *International Conference on Learning Representations*, 2024.

[13] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2002.

[14] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, “BERTScore: Evaluating text generation with BERT,” in *International Conference on Learning Representations*, 2020.

[15] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov, “Natural questions: A benchmark for question answering research,” *Transactions of the Association for Computational Linguistics*, 2019.

[16] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer, “TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension,” in *Proc. of the Annual Meeting of the Association for Computational Linguistics*, 2017.