QE-RAG: A Robust Retrieval-Augmented Generation Benchmark for Query Entry Errors

Anonymous ACL submission

Abstract

Current benchmarks evaluate the performance of RAG methods from various perspectives, they share a common assumption that user queries used for retrieval are error-free. However, in real-world interactions between users and LLMs, query entry errors are frequent. The impact of these errors on current RAG methods against such errors remains largely unexplored. To bridge this gap, we propose QE-RAG, the first robust RAG benchmark designed specifically to evaluate performance against query entry errors. We analyze the impact of these errors on LLM outputs and find that corrupted queries degrade model performance, which can be mitigated through query correction and training a robust retriever for retrieving relevant documents. Based on these insights, we propose a contrastive learning-based robust retriever training method and a retrieval-augmented query correction method. Extensive in-domain and cross-domain experiments reveal that: (1) state-of-the-art RAG methods including sequential, branching, and iterative methods, exhibit poor robustness to query entry errors; (2) our method significantly enhances the robustness of RAG when handling query entry errors and it's compatible with existing RAG methods, further improving their robustness.

1 Introduction

Retriever-augmented generation (RAG), which integrates retrieval mechanisms to incorporate external knowledge into large language models (LLMs), has become a widely adopted approach (Borgeaud et al., 2022; Lewis et al., 2020; Chen et al., 2024). By retrieving knowledge from external sources, RAG addresses issues such as insufficient knowledge and hallucinations in LLMs (Tonmoy et al., 2024; Gao et al., 2023b), thereby improving the accuracy and fidelity of their responses.

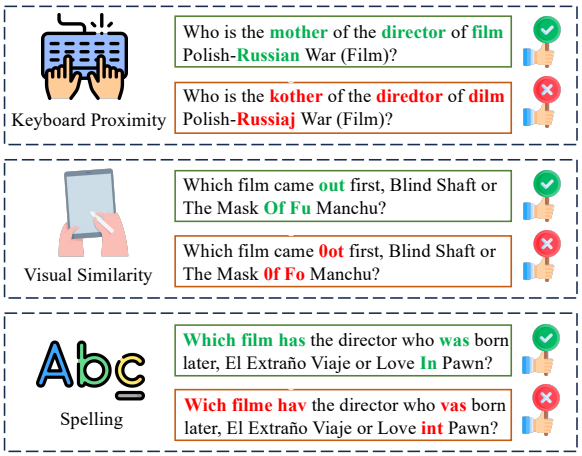


Figure 1: Examples of three types of query entry errors including keyboard proximity errors, visual similarity errors, and spelling errors.

Existing RAG benchmarks evaluate the performance of RAG methods from various perspectives. For example, Es et al. (2024) assess fidelity in LLM-generated content, Chen et al. (2024) evaluate the model's ability to refuse to answer inappropriate or unanswerable queries, and Liu et al. (2023) examine the capacity of models to handle counterfactual information. Although these studies provide valuable insights into model effectiveness across different scenarios, they universally assume that user queries are error-free. In real-world settings, as illustrated in Figure 1, user queries often contain entry errors such as keyboard proximity errors, visual similarity errors, and spelling mistakes. The impact of these errors on LLM outputs remains largely unexplored.

To fill this gap, we introduce **QE-RAG**, the first RAG benchmark specifically designed to evaluate model performance under query entry errors. We inject three common types of query errors—spelling errors, keyboard proximity errors, and visual similarity errors—into four direct QA datasets (TriviaQA (Joshi et al., 2017),

Natural Questions (Kwiatkowski et al., 2019), PopQA (Mallen et al., 2022), and WebQuestions (Berant et al., 2013)) and two multi-hop QA datasets (HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (Ho et al., 2020)). Specifically, we use the nlpaug (Ma, 2019) tool to systematically inject these errors, applying them in a 3:1:1 ratio to reflect real-world error distribution patterns. For each query, there is a 30% probability of selecting a word, and for each selected word, a 30% probability of corrupting a character. This setup realistically simulates typical user query behaviors, providing a practical evaluation environment for RAG models. Since these errors do not alter the user's underlying information need, we retain the original RAG labels for the corrupted queries. To simulate varying levels of noise, we generate two versions of the QE-RAG by corrupting 20% and 40% of the queries, representing moderate and high-error scenarios.

Based on the proposed QE-RAG dataset, we conducted preliminary experiments (§ 4.1) on the corrupted HotpotQA and Natural Questions (NQ) datasets to explore the impact of query entry errors on LLM outputs. We find that: (1) **Retrieving correct documents** for corrupted queries can enhance the RAG model's robustness to query entry errors. (2) **Correcting corrupted queries** also improves the RAG model's robustness. Therefore, (1) To retrieve correct documents, we train a robust retriever using contrastive learning based on a retrieval dataset with a 20% error query rate, enabling it to retrieve the correct document corresponding to the correct query even when faced with corrupted queries. (2) To correct corrupted queries, we adopt the current state-of-the-art LLM-based correction methods. However, considering the significant issue of overcorrection (Li et al., 2023a; Fang et al., 2023) in LLMs during correction and LLMs may have limitations in recognizing certain uncommon knowledge during query correction (Zhang et al., 2024a), we propose a query correction approach that combines RAG (based on the robust retriever we introduced earlier) with fine-tuning to mitigate overcorrection while enhancing robustness.

We selected the state-of-the-art retriever BGE (Xiao et al., 2023) from the MTEB leaderboard (Muennighoff et al., 2022) and two large language models, Qwen2 (Yang et al., 2024) and LLama3 (AI@Meta, 2024), to evaluate their robustness to query entry errors. We tested the in-domain and cross-domain performance of various existing RAG methods (e.g., trained on HotpotQA and tested on the same or other datasets) to assess their robustness against query entry errors. These RAG methods include standard RAG (Gao et al., 2023b), query reformulation (Gao et al., 2023a), document refinement (Jiang et al., 2023b), branching (Shi et al., 2024; Kim et al.) and iterative (Shao et al., 2023) methods.

Extensive experimental results show that while these state-of-the-art RAG methods demonstrate some effectiveness compared to standard RAG, their robustness to query entry errors remains limited. In contrast, the two methods we propose significantly enhance the robustness of RAG systems and can be combined with existing RAG methods to further improve their performance.

To summarize, our contributions are as follows:

- To the best of our knowledge, we are the first to investigate robustness against query entry errors in RAG research, focusing on three representative error types: keyboard proximity, visual similarity, and spelling.
- We construct a benchmark dataset, QE-RAG, based on six widely-used RAG datasets, incorporating two levels of noise through the explicit injection of three types of errors. Extensive experiments conducted on QE-RAG demonstrate that state-of-the-art RAG methods, including query reformulation, document refinement, branching, and iterative methods, exhibit poor robustness to query entry errors.
- We propose two solutions to improve robustness against query entry errors: (1) a contrastive learning-based trained robust retriever, which enhances RAG robustness; (2) a retrieval-augmented query correction method, resulting in further improvements in robustness.

2 Related Work

2.1 RAG benchmark

Existing RAG benchmarks primarily assess the quality of content generated by LLMs or the LLM's ability to process external information. RAGAS (Es et al., 2024) and ARES (Saad-Falcon et al., 2024) evaluate the contextual relevance and fidelity of LLM-generated content. RGB (Chen

Table 1: The statistics of six datasets used in QE-RAG. "Source" refers to the knowledge source of each dataset. "#Query" denotes the number of queries. "0% Prob", "20% Prob", "40% Prob" represent the proportions of corrupted queries in the dataset at 0%, 20%, and 40%, respectively. "Avg. #Char/Query" indicates the average number of characters per query. "Avg. #Words/Query" refers to the average number of words per query.

Type	Dataset	Source	#Query	Avg. #Chars/Query			Avg. #Words/Query		
				0% Prob	20% Prob	40% Prob	0% Prob	20% Prob	40% Prob
QA	NQ	Wiki	3610	48.4	48.6	48.7	9.4	9.4	9.4
	PopQA	Wiki	14267	37.1	37.4	37.7	6.7	6.8	6.9
	TrivalQA	Wiki & Web	11313	69.1	69.4	69.6	12.6	12.6	12.7
	WebQA	Google Freebase	2032	38.0	38.0	38.1	6.8	6.9	6.9
Multi-Hop QA	HotpotQA	Wiki	7405	94.5	94.8	95.1	16.4	16.4	16.5
	2wiki	Wiki	12576	68.1	68.5	68.8	12.4	12.5	12.5

et al., 2024) tests the robustness of LLM against noisy documents and the ability to refuse to answer, while RECALL (Liu et al., 2023) analyzes the LLM's processing capability regarding counterfactual information. However, they all assume that the queries used for retrieval are correct, without considering the actual scenarios where users may enter corrupted queries. In the increasingly popular era of LLM, this cannot well evaluate the real capabilities of RAG technology. Therefore, this paper focuses on establishing an RAG evaluation framework that includes corrupted queries, which can help evaluate the robustness of RAG models and promote the further development of RAG technology in the era of LLM.

2.2 Retriever Augmented Generation

Standard RAG methods (Gao et al., 2023b) supplement user queries with retrieved documents, which are then fed into the LLM to generate responses. Over time, numerous approaches have been proposed to further enhance the performance of RAG systems. Following (Jin et al., 2024), these methods can be categorized into sequential pipeline (Gao et al., 2023a), branching pipeline (Shi et al., 2024; Kim et al.), iterative pipeline (Shao et al., 2023), and so on. We provide a detailed description of them in Appendix A. In this paper, we will evaluate the robustness of these state-of-the-art RAG methods in scenarios where queries contain errors.

3 QE-RAG Dataset Construction

We focus on RAG in this study, which is formulated as follows: given a query $q \in \mathcal{Q}$ (where $\mathcal{Q}$ is the set of all possible queries) and an external knowledge base $K = \{d_1, d_2, \ldots, d_N\}$ consisting of N documents, the goal of RAG is to generate a response $a \in \mathcal{A}$ (where $\mathcal{A}$ is the set of possible answers) by leveraging both retrieval from the knowledge base and generation from a LLM. Unlike previous datasets, which assume that q is error-free, we consider a more practical scenario in which q may be corrupted by three types of query entry errors. As shown in Figure 2, our QE-RAG dataset is constructed through the following steps.

Step1: Selection of RAG Dataset. Following FlashRAG (Jin et al., 2024), we collect and extend six widely-used RAG datasets to form our **QE-RAG**, which includes four direct QA datasets (TriviaQA (Joshi et al., 2017), Natural Questions (Kwiatkowski et al., 2019), PopQA (Mallen et al., 2022), WebQuestions (Berant et al., 2013)) and two multi-hop QA datasets (HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020)). Each dataset follows the format "*question, gold answer*", representing the user query q and the gold answer a, respectively. The corpus K used for retrieval, also referred to as the external knowledge base, is set to the Wikipedia corpus. Please note that to comprehensively evaluate the robustness of existing methods against query entry errors, we conduct both in-domain and cross-domain robustness assessments. Following (Xu et al., 2024), we use HotpotQA as the source dataset, meaning we fine-tune the retrieval model exclusively on HotpotQA. Testing on HotpotQA constitutes in-domain evaluation while testing on other datasets represents cross-domain evaluation.

Step 2: Query Corruption. We utilize the nlpaug tool (Ma, 2019) to inject three types of query entry errors into the six collected datasets, forming the corrupted queries: (1) **Keyboard Proximity Errors.** When users interact with LLMs via a keyboard, mistyping may occur as a result of pressing adjacent keys. To simulate this, we replace correct letters with nearby letters on the keyboard. (2) **Visual Similarity Errors.** When users input words

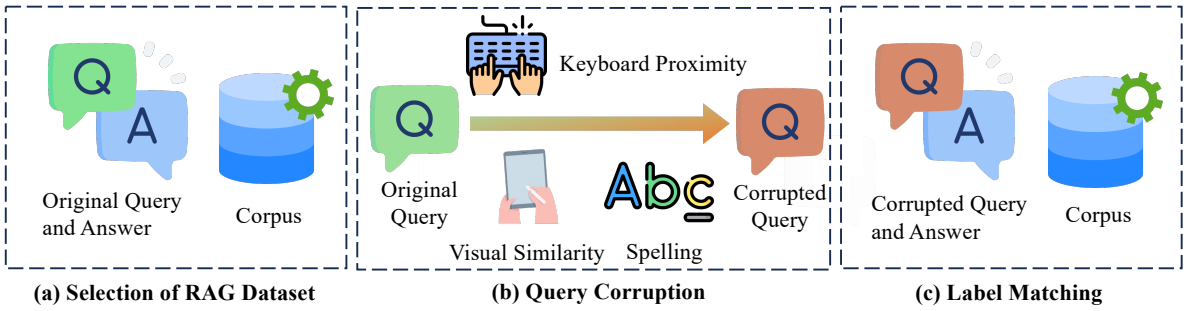


Figure 2: The construction process of QE-RAG datasets. (a) Selection of RAG Dataset. (b) Query corruption through three scenarios: keyboard proximity errors, visual similarity errors and spelling errors. (c) Label matching.

through handwriting, recognition tools may misinterpret characters due to irregular handwriting or inaccurate OCR algorithms, resulting in morphological errors. To simulate these handwriting input errors, we replace correct letters with visually similar ones. (3) **Spelling Errors.** Users may occasionally forget the correct spelling of a word and input an approximation, leading to spelling errors in the query. We simulate these errors by replacing words using a spelling error dictionary. Specifically, we apply a 30% probability of selecting a word in each query, and for each selected word, a 30% probability of corrupting a character. These probabilities reflect typical user behavior, creating a realistic test environment for RAG models.

Step 3: Label Matching. Since we set relatively low probabilities for both selecting a word and corrupting a character, we assume the corruption does not affect the underlying user information need and realistically simulates typical user query behavior. Therefore, we retain the original RAG labels for the corrupted queries. In other words, for an original data sample (q, a), we replace it with (q', a) where q' is the corrupted version of q containing one of the three entry errors, while a remains unchanged. Additionally, to evaluate model robustness under different levels of noise, we generate two versions of the QE-RAG dataset by corrupting 20% and 40% of the queries, representing moderate and high-error scenarios.

Dataset Statistics and Analysis. Table 1 presents the statistical analysis of the six datasets we constructed. It can be observed that the difference in the average number of words per query between corrupted queries (with error ratios of 20% and 40%) and original queries is not significant. This similarity indicates that our corruption strategy effectively mirrors real-world scenarios of user query entry errors. Additionally, our corruption strategy does not alter the syntactic structure of the sentences, as shown by the minimal difference in the average query length between original and corrupted queries in Table 1, further ensuring the quality of our QE-RAG dataset.

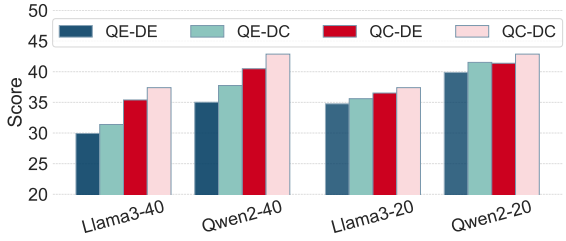
Evaluation. Following (Jin et al., 2024), QE-RAG support EM (Exact Match), F_1 (token-level F_1 score), and Acc (Accuracy) to evaluate the effectiveness and robustness against query entry errors of RAG methods. In this paper, we use F_1 for evaluation, as it better reflects the accuracy of the fine-grained information in the model's generated content. Additionally, we have developed a Python framework that facilitates the easy reproduction of experiments and the integration of new datasets and additional RAG methods. Further details on the datasets and evaluation code can be found at https://anonymous.4open.science/r/QE-RAG-DEA5.

4 Preliminary Experiments and Methodology

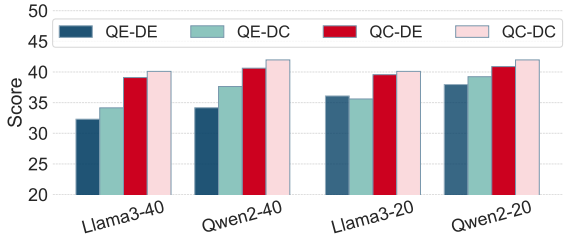
In this section, we first explore how query entry errors impact the performance of the RAG system. Then, we introduce two approaches: a contrastive learning-based robust retriever training method and a retrieval-augmented query correction method, both designed to enhance robustness against query entry errors.

4.1 Preliminary experiments

We conducted preliminary experiments on the HotpotQA and NQ datasets to investigate the impact of 40% and 20% ratio query entry errors on LLM-generated outputs when the LLMs are Llama3 and Qwen2. For this analysis, we kept the handling of correct queries unchanged and focused solely on scenarios involving corrupted queries. To evaluate the effect of various strategies



(a) Results on HotpotQA dataset.



(b) Results on Natural Questions dataset.

Figure 3: Preliminary experiments to explore the impact of query entry errors on RAG performance, where the retriever is BGE, with error ratios of 40% and 20%.

for mitigating the impact of errors, we tested the following approaches: 1) QE-DE: Query with Errors - Document Retrieved via Errors. 2) QE-DC: Query with Errors - Document Retrieved via Correct Query. 3) QC-DE: Corrected Query - Document Retrieved via Errors. 4) QC-DC: Corrected Query - Document Retrieved via Correct Query. More details are in Appendix B.

As shown in Figure 3, whether multi-hop QA (Figure 3 (a)) or direct QA (Figure 3 (b)), employing corrupted queries and their retrieved documents (QE-DE) gets poor model performance. In contrast, utilizing documents retrieved with correct queries (QE-DC) or using correct queries themselves (QC-DE) improved model performance. The combination of correct queries and the documents retrieved with those queries (QC-DC) achieved the best results.

Based on the above conclusions, we can infer that **retrieving correct documents for corrupted queries** and **query correction** can help address the issue of query entry errors and improve the model's robustness. Therefore, we design a contrastive learning-based robust retriever training method and a retrieval-augmented query correction method, which are in § 4.2 and § 4.3.

4.2 Contrastive Learning-Based Robust Retriever

In order to enable the retriever to retrieve correct documents using the corrupted query, we introduce a contrastive learning-based robust retriever training method in this section. Contrastive learning (CL) is a self-supervised learning technique designed to learn robust representations by contrasting positive and negative examples. We leverage CL to train the model to recognize and retrieve relevant documents even when queries are corrupted.

Specifically, we use the HotpotQA dataset, introducing a 20% corrupted query ratio to construct contrastive pairs in the format (q, a) and (q', a), where q and q' respectively denotes the original and corrupted query, and a denotes the golden LLM response. We then fine-tuned BGE (Xiao et al., 2023) models using contrastive learning on this dataset, with positive examples being the relevant documents corresponding to the original queries in HotpotQA. For negative example sampling, we included a hard negative example for each corrupted query, randomly chosen from the original HotpotQA corpus, along with randomly selected in-batch soft negative examples. The training objective is:

$$\mathcal{L} = -\log \frac{e^{\text{sim}(\mathbf{q}'_\mathbf{i}, \mathbf{d}^+_\mathbf{i})/\tau}}{e^{\text{sim}(\mathbf{q}'_\mathbf{i}, \mathbf{d}^+_\mathbf{i})/\tau} + \sum_{j=1}^{N} e^{\text{sim}(\mathbf{q}'_\mathbf{i}, \mathbf{d}^-_\mathbf{j})/\tau}}, \tag{1}$$

where $\mathbf{q}'_\mathbf{i}$, $\mathbf{d}^+_\mathbf{i}$, and $\mathbf{d}^-_\mathbf{i}$ denote the embeddings of the i-th corrupted query, the positive example, and the negative example, respectively. The function $\text{sim}(\cdot)$ represents the cosine similarity function, N is the batch size, and τ is the temperature.

4.3 Retrieval-Augmented Query Correction

To better adapt to the RAG scenario, in this section, we will explore query correction using LLMs in the RAG setting. Incorporating RAG can assist LLMs in answering questions by retrieving relevant documents. However, in the presence of query errors, providing LLMs with related documents can further complicate query correction. The LLM may prioritize answering the query based on the retrieved documents rather than focusing on the correction task. This occurs because the retrieval results may overwhelm the LLM, leading it to shift its focus from correcting the query to generating a response. To address these challenges, we propose using retrieval-augmented fine-tuning (Zhang et al., 2024b) to efficiently fine tuning LLMs to leverage retrieved documents specifically for query correction. This approach ensures the model remains focused on

correcting the query without deviating from answering it. That is:

$$\mathcal{L}_{\mathrm{FT}} = -\frac{1}{|\mathcal{D}_I|}\sum_{\mathcal{D}_I}\log(P_{\theta_1+\theta_L}(y_t|x,p,y_{<t})), \quad (2)$$

where θ_1 and θ_L are the parameters of LLM and LoRA (Hu et al., 2021). y_t and $y_{<t}$ respectively denote the t-th token and tokens before y_t. x denotes the original query with the retrieved documents. p is a prompt that allows the LLM to correct the query based on the retrieved documents. D_I represents the fine-tuning dataset composed of inputs x, p and the output, the correct query y.

5 Experiments

5.1 Experimental Settings

5.1.1 Datasets and Metrics

In the main experiment, we selected our modified RAG dataset to conduct experiments on RAG tasks. Specifically, we chose four QA datasets: TriviaQA (Joshi et al., 2017), Natural Questions (NQ) (Kwiatkowski et al., 2019), PopQA (Mallen et al., 2022), WebQuestions (WebQA) (Berant et al., 2013), and two Multi-Hop QA datasets: HotpotQA (Yang et al., 2018) and 2WikiMultihopQA(2wiki) (Ho et al., 2020) for our experiments. Following (Jin et al., 2024), we used the Wikipedia data from December 2018 as the retrieval corpus. For the evaluation metrics, following (Jin et al., 2024), we selected the widely used token-level F_1 score as our evaluation metric. We also support the use of other evaluation metrics.

5.1.2 Retrieval and Generation Models

In our main experiment, we selected bge-base-en-v1.5 (Xiao et al., 2023) as the retrieval models. As § 4.2 described, We trained them on the original HotpotQA dataset as well as the HotpotQA dataset we constructed with 20% corrupted queries, obtaining retrievers R_1 and R_2 respectively. For the baseline, we used R_1 as the retriever. For our method, we used R_2 as the retriever. For the generation models, we chose the latest Llama3-8B-Instruct (AI@Meta, 2024) and Qwen2-7B-Instruct (Yang et al., 2024) as the main experimental generation models.

5.1.3 RAG Methods

We test the following RAG methods. Standard RAG, CoT-RAG, Direct-Correct, HyDE, Iter-Retgen, LongLingua, REPLUG, SuRe. Details in Appendix C. For our proposed methods: **QER-RAG**: To enhance the robustness of retrieval, we replace the retriever R_1 with our trained retriever R_2 while keeping other components of the standard RAG method unchanged. **RA-QCG**: This method integrates our query correction approach into standard RAG. The original query is corrected using retrieved documents, and the corrected query is then used for RAG.

5.1.4 Implementation Details

We set the generation parameter do_sample to false to improve the reproducibility of the results. Except for the experiment in § 5.5 on the impact of the number of retrievals on robustness, in all RAG tasks, three documents are retrieved for each query given the computational costs. For the training of contrastive learning models in § 4.2, we set the learning rate to 2e-5, batch size to 64, and epoch to 1. We set the maximum input length to 4096 for the generation models. Following (Jin et al., 2024), we test 1000 queries for each RAG dataset. All experiments are conducted on Nvidia A6000 GPUs. More details can be found at the link provided in the Evaluation part of § 3. More details in Appendix D.

5.2 Main Results

Table 2 shows the main experimental results of different methods in six QE-RAG datasets with two different corrupted query proportions (20%, 40%) when the retrieval model is BGE. From the table, we can draw the following observations:

The Poor Robustness of Existing SOTA RAG Methods. It can be observed that when the dataset contains corrupted queries (with error ratios of 20% or 40%), the performance of existing SOTA RAG methods in performing is suboptimal. As the proportion of corrupted queries increases, the model's performance deteriorates progressively, indicating its lack of robustness when handling query entry errors. This phenomenon underscores the critical importance of handling query entry errors for the success of RAG tasks.

The Effectiveness of QER-RAG. Our proposed QER-RAG method builds upon the standard RAG with improvements. Specifically, QER-RAG differs from standard RAG in that it uses a retriever trained on a dataset containing corrupted queries. Experimental results show that QER-RAG achieves significant improvements at both error ratios (20% and 40%). This result demon-

Table 2: The overall performance of the RAG task under six datasets and two different error proportions of query scenarios when the retrieval model is BGE, and the generator models are Llama3 and Qwen2. The **"overall"** column represents the average result of that row, which is the average result of the method across all datasets and the two LLMs. The optimal **"overall"** results are presented in bold.

Dataset	HotpotQA	NQ	PopQA	TrivalQA	WebQA	2wiki	HotpotQA	NQ	PopQA	TrivalQA	WebQA	2wiki	Overall
Method	Llama3						Qwen2						
	40% Corrupted Queries												
Standard RAG	29.92	32.30	33.22	52.27	28.65	16.94	35.02	34.16	35.90	52.85	31.16	30.26	34.39
CoT-RAG	29.58	32.24	33.04	52.31	28.94	16.97	36.49	36.07	37.54	54.07	32.42	32.00	35.14
Direct-Correct	22.26	30.29	32.33	36.30	24.92	16.15	34.78	33.92	35.95	52.94	31.33	30.07	31.77
HyDE	7.16	17.82	2.36	19.58	12.79	4.35	25.10	23.33	29.68	33.21	22.14	25.07	18.55
Iter-Retgen	29.29	32.24	32.99	52.02	28.99	27.19	9.72	14.99	8.13	24.96	14.19	5.66	23.36
REPLUG	26.39	29.93	28.08	49.40	29.12	17.63	31.14	26.49	27.80	47.65	25.50	27.24	30.53
LongLingua	28.02	29.24	30.66	50.38	29.84	20.55	25.75	21.85	19.96	42.92	24.06	24.87	29.01
SuRe	24.50	32.96	38.42	47.84	31.35	14.81	31.48	27.91	31.02	51.44	30.48	28.40	32.55
QER-RAG	30.10	35.12	35.17	55.01	29.22	17.53	33.59	36.56	38.36	51.45	33.64	25.19	35.08
RA-QCG	31.23	38.44	35.86	57.87	30.30	17.80	38.19	39.04	38.17	57.00	33.44	32.98	**37.52**
	20% Corrupted Queries												
Standard RAG	34.76	36.09	36.89	57.76	30.65	18.06	39.88	37.95	39.83	58.33	33.40	32.83	38.04
CoT-RAG	34.01	36.09	36.50	57.62	30.84	18.07	39.65	37.70	39.96	58.34	33.51	32.98	37.94
Direct-Correct	23.59	31.57	31.31	34.60	24.85	15.84	25.12	23.64	30.84	32.95	22.84	25.51	26.89
HyDE	7.28	18.76	2.20	20.32	11.47	4.27	9.85	15.79	7.66	26.61	15.87	5.80	12.16
Iter-Retgen	34.35	35.80	36.79	57.34	30.98	17.16	35.65	28.40	31.06	53.16	26.96	29.60	34.77
REPLUG	30.24	33.55	31.55	54.27	31.26	20.27	28.89	25.43	22.25	47.05	26.46	26.93	31.51
LongLingua	33.30	33.43	32.96	56.94	31.42	23.21	34.74	31.58	34.26	55.59	33.14	31.23	35.98
SuRe	27.91	36.97	42.17	53.17	34.81	16.19	38.23	40.31	43.78	56.29	35.54	27.78	37.76
QER-RAG	33.31	38.24	38.71	58.85	31.83	18.95	39.84	39.21	41.43	58.66	34.83	35.24	39.09
RA-QCG	35.08	39.64	39.02	60.55	32.26	19.62	41.65	40.71	41.84	59.77	35.74	36.03	**40.16**

Table 3: The compatibility with existing RAG methods when the error rate is 20% and the LLM is Llama3.

Method	HotpotQA	NQ	PopQA	TrivalQA	WebQA	2wiki
Iter-Retgen	34.35	35.80	36.79	57.34	30.98	17.16
+RA-QCG	**35.45**	**38.94**	**38.94**	**60.84**	**32.71**	**19.27**
REPLUG	30.24	33.55	31.55	54.27	31.26	20.27
+RA-QCG	**31.19**	**36.25**	**34.53**	**58.78**	**32.86**	**22.01**
LongLingua	33.30	33.43	32.96	56.94	31.42	23.21
+RA-QCG	32.76	**35.56**	**34.22**	**58.15**	**32.92**	**23.50**
SuRe	27.91	36.97	42.17	53.17	34.81	16.19
+RA-QCG	**29.41**	**39.12**	**44.28**	**55.21**	**35.91**	**18.33**

strates the effectiveness of the contrastive learning approach we introduced in training the retriever with a dataset containing corrupted queries. By incorporating a certain proportion (specifically, 20%) of corrupted queries into the retriever's training data, we can significantly improve the retriever's robustness, allowing it to still retrieve relevant documents in the face of corrupted inputs and helping the LLM generate more accurate responses.

The Effectiveness of RA-QCG. Building on QER-RAG, we further propose the RA-QCG method, which introduces a query correction mechanism based on RAG. Experimental results show that RA-QCG achieves optimal overall performance at both error ratios (20% and 40%), and in the case of a 40% error ratio, RA-QCG's performance even approaches the best baseline performance observed at the 20% error ratio. This result fully validates the effectiveness of our RAG-assisted query correction approach.

In Appendix E, we present a comparison of the model's performance on queries that are entirely correct and also demonstrate the robustness of our approach. Additionally, Appendix G qualitatively demonstrates the robustness of our approach.

5.3 Compatibility with SOTA RAG

From the main experiments in § 5.2, we observe that state-of-the-art RAG methods offer notable improvements over standard RAG methods. This inspired us to explore whether our proposed approach is compatible with these methods, potentially further enhancing RAG system performance and robustness. In this section, we investigate the effectiveness of combining our method with four advanced RAG methods—IterGen, LongLingua, RePlug, and Sure—under the setting where the LLM is LLama3 and the query error rate is 20%.

The results are shown in Table 3. The performance gains are observed across all tested RAG methods, demonstrating its generalizability and flexibility in complementing diverse retrieval and reasoning strategies. By incorporating our query correction mechanism and robust retrieval approach, these methods show enhanced robustness when handling queries with entry errors.

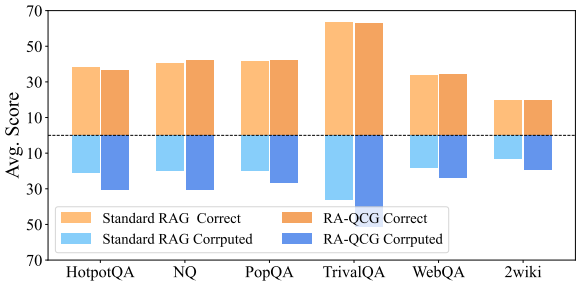


Figure 4: The robustness comparison of correct and corrupted queries to the average F_1 score when the retrieval model is Standard RAG and RA-QCG, the generative model is Llama3 and the error rate is 20%. Above and below the X-axis represent the average token level F_1 value of the correct and corrupted query, respectively.

5.4 Robustness Comparison of Correct and Corrupted Query

Table 2 in the main experiment shows the overall RAG performance for all queries (correct and corrupted queries), but we are unaware of how the RAG model performs on correct versus corrupted queries individually. RA-QGC improves upon standard RAG. Therefore, in this section, we explore the average F_1 scores of RA-QGC and standard RAG across six datasets with a 20% corrupted query ratio when the LLM is Llama3. We have a total of 1000 queries, of which 200 are corrupted and 800 are correct.

The results are shown in Figure 4. It can be seen that the average performance on correct queries is similar across all six datasets, while for corrupted queries, RA-QGC demonstrates a significant advantage, with its average score outperforming standard RAG across all datasets. This experiment illustrates that RA-QGC can effectively improve the robustness of the RAG method in both in-domain and cross-domain datasets when faced with query entry errors, thus enhancing the overall performance of the RAG method.

5.5 Robustness on the Number of Documents Retrieved

In this section, we explore the impact of retrieving different numbers of documents on the robustness of RAG methods. We test standard RAG, Direct-Correct, and RA-QGC with Llama3 as the LLM, using retrievals of 1, 3, 5, and 15 documents to supplement the LLM's knowledge. The results are shown in Figure 5. The following conclusions can be drawn: (a) RA-QGC consistently achieves improvements. (b) Selecting an appropriate number of documents for retrieval is crucial. (c) The necessity of query correction based on RAG. Details in Appendix F.

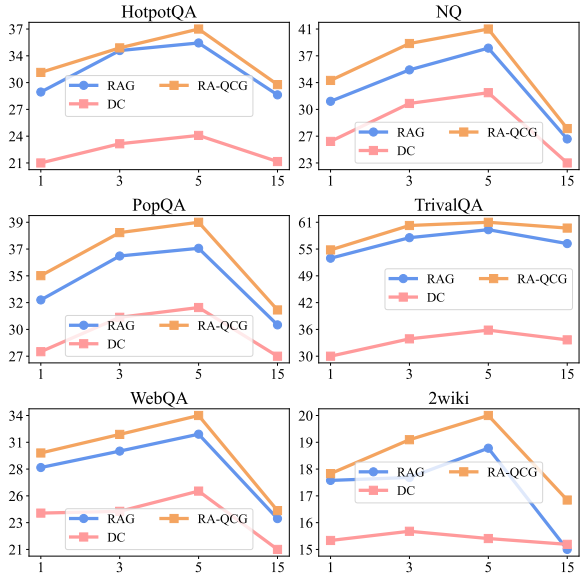


Figure 5: The results of Standard RAG (RAG), Direct-Correct (DC) and RA-QCG retrieving varying numbers of documents on six datasets when the LLM is Llama3 and the error rate is 20%. The x-axis represents the number of retrieved documents, specifically 1, 3, 5, and 15, while the y-axis indicates the token-level F_1 score.

6 Conclusion

In this paper, we present the first comprehensive investigation into the robustness of retrieval-augmented generation against query entry errors. We build the QE-RAG by simulating three types of query errors: "keyboard proximity, visual similarity, and spelling" based on six RAG datasets with varying error ratios. We find that corrupted queries lead to a performance drop in the RAG methods, but this can be alleviated through query correction and retrieval model adjustments. Based on QE-RAG, we test standard RAG, existing SOTA RAG methods (including query reformulation, document compression, branching, and iterative methods), as well as our proposed robust retrieval method, which is trained using contrastive learning on corrupted queries and retrieval-augmented query correction method. The results show that existing RAG methods exhibit poor robustness to query entry errors, while our two proposed methods effectively enhance the robustness of the RAG methods.

7 Limitations

We believe that QE-RAG can promote the development of the LLM and RAG fields, yet it still has the following limitations: First, QE-RAG currently includes only six datasets for QA and Multi-Hop QA. In the future, we plan to expand the benchmark to encompass a wider range of RAG-related datasets, such as fact-checking (Petroni et al., 2021; Thorne et al., 2018), multiple-choice (Hendrycks et al.; Lin et al., 2022) tasks, and others for the community to test. Second, our work primarily focuses on query entry errors. However, in RAG scenarios, retrieved documents may also be incorrect. How to jointly address errors in both queries and retrieved documents is a direction for future research.

References

AI@Meta. 2024. Llama 3 model card.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

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.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158.

Tao Fang, Shu Yang, Kaixin Lan, Derek F Wong, Jinpeng Hu, Lidia S Chao, and Yue Zhang. 2023. Is chatgpt a highly fluent grammatical error correction system? a comprehensive evaluation. *arXiv preprint arXiv:2304.01746*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023a. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

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*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023a. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023b. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*.

Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. *arXiv preprint arXiv:2405.13576*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. Sure: Summarizing retrievals using answer candidates for open-domain qa of llms. In *The Twelfth International Conference on Learning Representations*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions*

of the Association for Computational Linguistics, 7:453–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Yinghui Li, Haojing Huang, Shirong Ma, Yong Jiang, Yangning Li, Feng Zhou, Hai-Tao Zheng, and Qingyu Zhou. 2023a. On the (in) effectiveness of large language models for chinese text correction. *arXiv preprint arXiv:2307.09007*.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023b. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6342–6353.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252.

Yi Liu, Lianzhe Huang, Shicheng Li, Sishuo Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Recall: A benchmark for llms robustness against external counterfactual knowledge. *arXiv preprint arXiv:2311.08147*.

Edward Ma. 2019. Nlp augmentation. https://github.com/makcedward/nlpaug.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*, 7.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. 2021. Kilt: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. Ares: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8364–8377.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819.

SM Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. 2024. A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313*.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9414–9423.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2023. C-pack: Packaged resources to advance general chinese embedding. *arXiv preprint arXiv:2309.07597*.

Haike Xu, Zongyu Lin, Yizhou Sun, Kai-Wei Chang, and Piotr Indyk. 2024. Sparsecl: Sparse contrastive learning for contradiction retrieval. *arXiv preprint arXiv:2406.10746*.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Dezhi Ye, Bowen Tian, Jiabin Fan, Jie Liu, Tianhua Zhou, Xiang Chen, Mingming Li, and Jin Ma. 2023. Improving query correction using pre-train language model in search engines. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 2999–3008.

Kepu Zhang, Zhongxiang Sun, Xiao Zhang, Xiaoxue Zang, Kai Zheng, Yang Song, and Jun Xu. 2024a. Trigger^3: Refining query correction via adaptive model selector. *arXiv preprint arXiv:2412.12701*.

Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E Gonzalez. 2024b. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131*.

A More related work

In the sequential pipeline, query reformulation methods focus on improving the input query to optimize the retrieval process. These techniques operate under the assumption that user queries may not always be optimal for retrieval tasks: HyDE (Gao et al., 2023a): The LLM generates a hypothetical document based on the query, which is then used as the query for retrieval. This approach assumes that the generated document aligns better with the retrieved documents. Query2doc (Wang et al., 2023) concatenates the LLM-generated pseudo-document with the original query to form a new query for retrieval. Rewrite-Retrieve-Read (Ma et al., 2023) proposes fine-tuning a query rewriter to optimize query reformulation. BEQUE (Ye et al., 2023) employs a combination of fine-tuning and reinforcement learning to rewrite queries, particularly improving retrieval performance for long-tail queries. The above query reformulation methods do not consider that the query itself is corrupted, thus ignoring that query reformulation may accumulate and amplify errors, which will seriously affect the final RAG performance. Another line of work involves processing the retrieved documents to make them more useful for the LLM: Selective-Content (Li et al., 2023b) compresses the provided context by removing redundant information using self-information metrics. LLMLingua (Jiang et al., 2023a) uses smaller models to detect and remove unnecessary tokens in the prompt, making the remaining content more interpretable for the LLM (even if humans may find it less comprehensible). LongLLMLingua (Jiang et al., 2023b) extends LLMLingua by incorporating question-aware techniques to extract key information from retrieved documents, improving their alignment with the LLM's processing capabilities.

Branching pipelines process multiple paths in parallel to enhance performance: REPLUG (Shi et al., 2024) integrates document relevance into the LLM's response generation, improving the accuracy and contextual alignment of generated outputs. SuRe (Kim et al.) utilizes summarization techniques to select the most suitable answer from multiple candidate responses. Iterative pipelines aim to refine the retrieval process dynamically Iter-RetGen (Shao et al., 2023) enhances the retrieval query by iteratively incorporating LLM responses into the query, leveraging the generated feedback to refine retrieval results.

B Preliminary Experiments

The following is a detailed introduction to the preliminary experiments.

- QE-DE (Query with Errors - Document Retrieved via Errors): The corrupted query is used to retrieve three documents (the same as below), which are then fed to the LLM for generation. This represents the baseline performance when corrupted queries are directly used without any correction.

- QE-DC (Query with Errors - Document Retrieved via Correct Query): The corrupted query is paired with the documents retrieved using the corresponding correct query. Both are provided to the LLM for generation. This method evaluates whether providing documents retrieved with the correct query can mitigate the negative impact of query errors.

- QC-DE (Corrected Query - Document Retrieved via Errors): The corrected query (corresponding to the corrupted query) is used alongside the documents retrieved using the corrupted query. This tests the effectiveness of query correction in improving LLM outputs despite inaccurate retrieval.

- QC-DC (Corrected Query - Document Retrieved via Correct Query): The corrected query is paired with documents retrieved using the corresponding correct query. This represents the optimal scenario, where both the query and retrieval documents are corrected, and serves as an upper bound for the performance improvements achievable by correcting queries and retrieval results.

C Details of RAG methods

We begin by evaluating the **Standard RAG** method, where the LLM generates responses directly based on the retrieved documents. We extend this baseline by introducing **CoT-RAG**, which prompts the LLM to consider whether the original query contains errors while generating a response. For query reformulation baselines, we focus on cost-effective, training-free approaches for evaluation: **Direct-Correct**: The LLM corrects the input query directly, and the corrected query is used for retrieval. **HyDE** (Gao et al., 2023a): The LLM generates a pseudo-document answering the query, which is then used as the new query for retrieval. **Iter-Retgen** (Shao et al., 2023): This method iteratively refines retrieval by leveraging the LLM's responses combined with the original query as new retrieval queries. To evaluate methods that refine retrieved documents, we consider **LongLingua** (Jiang et al., 2023b), which uses the LLM to modify the retrieved documents based on the query perplexity, making them more interpretable and better aligned with the LLM's contextual understanding. For branching methods, we evaluate: **REPLUG** (Shi et al., 2024): Enhances response generation by integrating document relevance into the output. **SuRe** (Kim et al.): Summarizes multiple candidate answers to determine the most appropriate response. All the above methods use R_1 as the retriever.

D Implementation Details

We employed the HuggingFace Transformers (Wolf et al., 2020) in PyTorch for the experiments. We use LoRA (Hu et al., 2021) for efficient fine-tuning of LLMs, using the Adam optimizer (Kingma and Ba, 2014), setting the initial learning rate to 5e-5, batch size to 16, and employing a cosine learning rate schedule. We train for 3 epochs with 1,000 pieces of data from the training dataset of HotpotQA with a 20% error rate. For Iter-Retgen, we iterate one round. For LongLingua, we use LLM itself as the compressor, with the compression rate set to 0.5 and the rest consistent with the original paper. For REPLUG, we keep its original settings. For SuRe, we use the prompt provided in the original paper to summarize and select candidate answers.

E Robustness on Correct Queries

In this section, we investigate the robustness of our proposed method when the query error rate is 0%. Specifically, we aim to assess whether focusing on handling corrupted queries negatively impacts performance on correct queries. For this evaluation, we use the same models and RAG methods as in the main experiments, but the dataset consists entirely of correct queries.

The results, presented in Table 4, demonstrate that our method achieves the best overall performance when all queries are correct. This highlights the robustness of our approach, which does not compromise its ability to handle correct queries despite its emphasis on addressing corrupted queries. Additionally, comparing Table 2 with Table 4 reveals that the performance of all methods improves when the queries are error-free. This observation further validates the findings from our preliminary experiments in § 4.1: correcting query entry errors such as keyboard proximity errors, visual similarity errors, and spelling mistakes can enhance the overall performance of RAG systems. By improving the accuracy and relevance of retrieved documents, such corrections contribute to a better user experience. Overall, these results confirm that our method effectively balances robustness across both corrupted and correct queries, ensuring high performance in real-world scenarios where query quality varies.

F More Details of Robustness on the Number of Documents Retrieved

The details of the Figure 5 are: (a) Regardless of the number of documents retrieved, RA-QGC consistently achieves improvements. This indicates that RA-QGC is more robust and is not limited by the number of retrieved documents, meaning it works effectively across various resource configurations (retrieving different numbers of documents). (b) The performance of RAG increases and then decreases as the number of retrieved doc-

Table 4: The overall performance of the RAG task under six datasets and 0% error proportions of query scenarios when the retrieval model is BGE, and the generator models are Llama3 and Qwen2. The **"overall"** column represents the average result of that row, which is the average result of the method across all datasets and the two LLMs. The optimal **"overall"** results are presented in bold.

Dataset	HotpotQA	NQ	PopQA	TrivalQA	WebQA	2wiki	HotpotQA	NQ	PopQA	TrivalQA	WebQA	2wiki	Overall
Method	Llama3						Qwen2						
	0% Corrupted Queries												
Standard RAG	37.40	40.10	40.83	63.32	33.56	20.72	42.87	41.97	43.66	64.44	36.74	36.49	41.84
CoT-RAG	36.84	40.03	40.44	63.07	33.95	20.68	42.52	41.94	43.84	64.46	36.98	36.31	41.76
Direct-Correct	23.14	33.22	37.53	33.99	27.22	16.80	26.02	23.39	32.64	31.74	22.64	26.81	27.93
HyDE	8.06	19.92	2.27	22.08	12.12	4.93	9.99	16.51	7.43	28.11	17.12	5.48	12.83
Iter-Retgen	36.81	39.82	40.49	63.08	33.78	19.37	38.84	31.76	34.42	58.69	26.03	32.84	37.99
REPLUG	33.83	37.53	34.39	59.99	34.98	21.83	32.42	28.60	24.45	52.71	28.78	29.52	34.92
LongLingua	35.47	37.31	36.14	61.88	34.77	25.92	38.18	35.69	37.90	61.15	35.33	34.51	39.52
SuRe	30.20	41.54	46.12	59.12	39.17	19.10	42.09	44.43	48.70	62.74	39.92	31.60	42.06
QER-RAG	36.32	41.55	41.59	63.67	34.45	20.69	42.92	42.73	44.57	63.70	37.70	38.09	**42.33**
RA-QCG	36.22	41.50	41.59	63.70	34.51	20.68	42.90	42.73	44.57	63.71	37.77	38.09	**42.33**

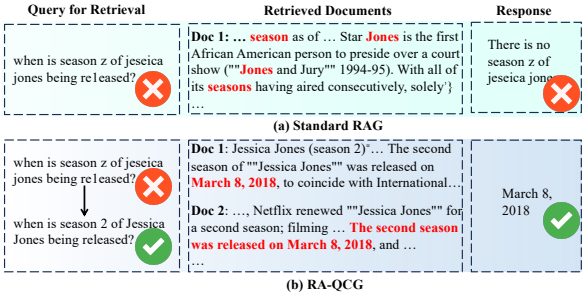


Figure 6: The case study of Standard RAG and RA-QCG.

uments changes, similar to the pattern observed in correct query scenarios (Jin et al., 2024). This suggests that selecting an appropriate number of documents for retrieval is crucial, balancing resources and RAG performance while accounting for the potential noise introduced by more documents. (c) It can be seen that using LLM-based direct correction significantly worsens RAG performance, and increasing the number of retrieved documents does little to alleviate the over-correction issue in LLMs. This highlights the necessity of query correction based on RAG, which leads to more accurate corrections and, as a result, improved RAG performance.

G Qualitative Analysis on Robustness

To investigate how our proposed method enhances model robustness, we conduct a qualitative analysis. Given that our method builds on the standard RAG, we compare the performance of RA-QCG with the standard RAG using a randomly selected example from the NQ dataset, with Llama3 as the LLM. This analysis examines three key components: the query used for retrieval, the documents retrieved, and the final responses generated by the LLM.

The results are illustrated in Figure 6. **Query for Retrieval.** In standard RAG, the corrupted query provided by the user is directly used for retrieval. In contrast, RA-QCG identifies and corrects the errors in the query before the retrieval stage, effectively mitigating the impact of input inaccuracies. This step ensures that the subsequent retrieval process operates on a more accurate representation of the user's intent. **Retrieved Documents.** Due to the use of the corrupted query, the standard RAG retrieves documents that are misaligned with the user's intended question. As a result, the retrieved documents lack the necessary information to answer the query correctly. Conversely, RA-QCG, by utilizing the corrected query, retrieves documents that are well-aligned with the user's intent, containing the relevant information needed to address the query effectively. **Response.** The shortcomings of the standard RAG are evident in the response generation stage. The misaligned documents retrieved by it lead to an incoherent or incorrect response that fails to answer the user's question. On the other hand, RA-QCG benefits from the corrected query and the retrieval of relevant documents, enabling the LLM to generate a response that is accurate and contextually appropriate. This analysis highlights how RA-QCG successfully corrects the query, retrieves documents that provide the necessary context and produces accurate answers. RA-QCG improves the robustness and reliability of the RAG system.