# LongRAG: Enhancing Retrieval-Augmented Generation with Long-context LLMs

♠Ziyan Jiang, ♠Xueguang Ma, ♠Wenhu Chen
♠University of Waterloo
ziyanjiang528@gmail.com, {x93ma ,wenhuchen}@uwaterloo.ca

**Project Website:** https://tiger-ai-lab.github.io/LongRAG/

## Abstract

In traditional RAG framework, the basic retrieval units are normally short. The common retrievers like DPR normally work with 100-word Wikipedia paragraphs. Such a design forces the retriever to search over a large corpus to find the "needle" unit. In contrast, the readers only need to generate answers from the short retrieved units. The imbalanced "heavy" retriever and "light" reader design can lead to sub-optimal performance. The loss of contextual information in the short, chunked units may increase the likelihood of introducing hard negatives during the retrieval stage. Additionally, the reader might not fully leverage the capabilities of recent advancements in LLMs. In order to alleviate the imbalance, we propose a new framework LongRAG, consisting of a **"long retriever"** and a **"long reader"**. In the two Wikipedia-based datasets, NQ and HotpotQA, where the average document size is less than 1K tokens, LongRAG processes the entire Wikipedia corpus into 4K-token units by grouping related documents, making these units 30 times longer than before. By increasing the unit size, we significantly reduce the total number of units from 22M to 600K. This greatly reduces the burden on the retriever, resulting in strong retrieval performance with only a few (less than 8) top units. Compared to traditional RAG, which may require hundreds of short units to achieve similar retrieval performance, our approach minimizes the likelihood of retrieving hard negatives while maintaining semantic integrity of each unit. Then we feed these retrieved units ($\approx$ 30K tokens) to an existing long-context LLM to perform zero-shot answer generation. Without requiring any training, LongRAG achieves an EM of 62.7% on NQ and 64.3% on HotpotQA, which are on par with the (fully-trained) SoTA model. Furthermore, we test on two non-Wikipedia-based datasets, Qasper and MultiFieldQA-en, where the average document length is already above 4K tokens. LongRAG processes each individual document as a single (long) unit rather than chunking them into smaller units. By doing so, we achieve an F1 score of 25.9% on Qasper (previously 22.5%) and 57.5% on MultiFieldQA-en (previously 51.2%). Our study offers insights into the future roadmap for combining RAG with long-context LLMs.

## 1 Introduction

Retrieval-Augmented Generation (RAG) methods have long been employed to enhance large language models (LLMs) (Mialon et al., 2023). Knowledge in the form of natural language can be entirely offloaded from the parametric knowledge of LLMs by leveraging a standalone retrieval component from an external corpus. The existing RAG framework tends to use short retrieval units, such as 100-word passages in popular open-domain question-answering tasks (Chen et al., 2017; Lewis et al., 2020; Karpukhin et al., 2020). The retriever is tasked with finding the "needle" (i.e. the precise tiny retrieval unit) from the "haystack" (i.e. the massive corpus with up to tens of millions of information units). Subsequently, the retrieved units are passed to the reader to generate the final response. On the contrary, the reader only needs to extract answers from

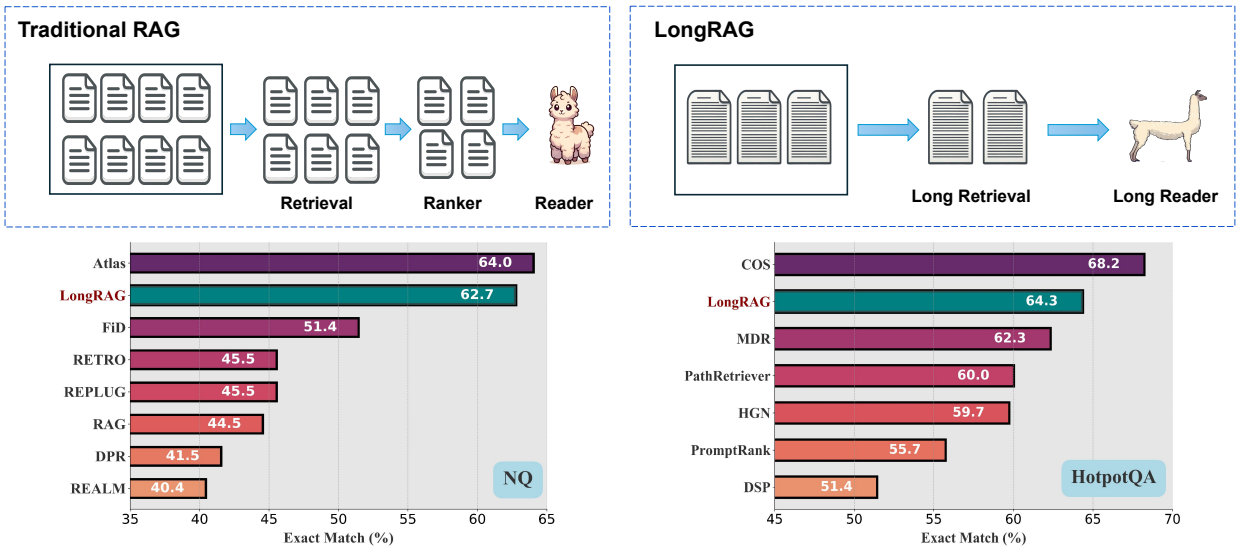


Figure 1: Traditional RAG vs. LongRAG. (Up) Traditional RAG operates on short retrieval units, where the retriever needs to scan over a massive amount of units to find the relevant piece. In contrast, LongRAG operates on long retrieval units (30x longer). The retriever of LongRAG has a significantly reduced workload, achieving strong retrieval quality by leveraging only a few top units without the need for additional ranking mechanisms or other complex components. LongRAG could fully exploit the ability of long-context language models to achieve strong performance. (Down) QA performance compared with other methods on the NQ dataset and the HotpotQA dataset.

these retrievals, which is a fairly easy task. This kind of imbalanced design, with a "heavy" retriever and a "light" reader, puts too much pressure on the retriever. Therefore, existing RAG models (Izacard & Grave, 2020b) have to recall huge amounts of units, such as the top-100/200, combined with additional re-ranker to achieve the best performance. Moreover, short retrieval units can lead to semantic incompleteness due to document truncation. This can result in the loss of contextual information, which may ultimately harm overall performance. This design choice was made in an era when the reader models were heavily restricted by their ability to handle long and contexts. With the recent advances in long-context language models, the reader can potentially handle up to 128K or even millions of tokens as input (Reid et al., 2024; Achiam et al., 2023). In this paper, we propose to revisit this design choice for open-domain question answering and propose the LongRAG framework as a solution to balance the workload between the retriever and the reader, as illustrated in Figure 1. There are three important designs in our novel framework:

**Long Retrieval Unit**: By using entire documents or grouping multiple related documents, we can construct long retrieval units with more than 4K tokens. This design could also significantly reduce the corpus size (number of retrieval units in the corpus). This makes the retriever's task much easier by providing more complete information, allowing the retriever's architecture to be simplified without the need for additional re-rankers or iterative retrieval.

**Long Retriever**: The long retriever will identify coarse relevant information for the given query by searching through all the long retrieval units in the corpus. Only a few top retrieval units (1 to 8 retrieval units in the four datasets we tested on), without re-ranking, are used for the next step. Compared to retrieving hundreds of short units, the long retriever only needs to retrieve a few candidates, which significantly reduces the likelihood to encounter hard negatives (it will confuse the reader).

**Long Reader**: The long reader will further extract answers from the concatenation of retrievals, which is normally around 30K tokens. We simply prompt an existing long-context LM (like Gemini or GPT4) with the question to produce the answers in a zero-shot fashion.