# Towards Adaptive Memory-Based Optimization for Enhanced Retrieval-Augmented Generation

Anonymous ACL submission

## Abstract

Retrieval-Augmented Generation (RAG), by integrating non-parametric knowledge from external knowledge bases into models, has emerged as a promising approach to enhancing response accuracy while mitigating factual errors and hallucinations. This method has been widely applied in tasks such as Question Answering (QA). However, existing RAG methods struggle with open-domain QA tasks because they perform independent retrieval operations and directly incorporate the retrieved information into generation without maintaining a summarizing memory or using adaptive retrieval strategies, leading to noise from redundant information and insufficient information integration. To address these challenges, we propose **A**daptive **m**emory-**b**ased optimization for **e**nhanced **R**AG (**Amber**) for open-domain QA tasks, which comprises an Agent-based Memory Updater, an Adaptive Information Collector, and a Multi-granular Content Filter, working together within an iterative memory updating paradigm. Specifically, **Amber** integrates and optimizes the language model's memory through a multi-agent collaborative approach, ensuring comprehensive knowledge integration from previous retrieval steps. It dynamically adjusts retrieval queries and decides when to stop retrieval based on the accumulated knowledge, enhancing retrieval efficiency and effectiveness. Additionally, it reduces noise by filtering irrelevant content at multiple levels, retaining essential information to improve overall model performance. We conduct extensive experiments on several open-domain QA datasets, and the results demonstrate the superiority and effectiveness of our method and its components. The source code is available [1].

## 1 Introduction

In recent years, Large Language Models (LLMs) (Brown et al., 2020; Achiam et al., 2023; Touvron et al., 2023b; Anil et al., 2023) have demonstrated exceptional performance across various tasks, including question answering (QA) (Yang et al., 2018; Kwiatkowski et al., 2019), owing to their ability to capture diverse knowledge through billions of parameters. However, even the most advanced LLMs often suffer from hallucinations (Chen et al., 2023) and factual inaccuracies due to their reliance on parametric memory. Additionally, it is impractical for these models to memorize all of the ever-evolving knowledge. To address these challenges, retrieval-augmented generation (RAG) (Borgeaud et al., 2022; Izacard et al., 2023; Shi et al., 2023) have garnered increasing attention. These models retrieve passages relevant to the query from external corpora and incorporate them as context to the LLMs, enabling the generation of more reliable answers. By integrating retrieved information, retrieval-augmented LLMs maintain both the accuracy and timeliness of their knowledge. Early studies on RAG primarily focused on single-hop queries (Lazaridou et al., 2022; Ram et al., 2023), where answers can typically be found within a single document. However, these methods often fall short when handling complex QA tasks, such as long-form QA and multi-hop QA, which require aggregating information from multiple sources. Unlike single-hop QA, these queries necessitate connecting and synthesizing information across multiple documents and cannot be solved by a single retrieval-and-response step. For instance, the query "*Is Microsoft Office 2019 available in a greater number of languages than Microsoft Office 2013?*" requires three reasoning steps: first, retrieving information about the languages supported by "*Office 2019*"; second, retrieving similar information for "*Office 2013*"; and finally, comparing the two sets of data to produce an answer.

To address this issue, Adaptive RAG has been proposed. It adaptively selects appropriate retrieval questions and timing based on the difficulty of the

[1] https://anonymous.4open.science/r/Amber-B203/

user query to flexibly capture more valuable knowledge for answering open-domain QA tasks, achieving a balance between effectiveness and efficiency. However, these methods still have several problems. ***First***, each retrieval operates independently and lacks a summarizing memory of previous retrieval fragments, which may cause the outputs to reflect only limited knowledge from specific retrieval steps while neglecting the integration and interaction of retrieved information from different steps. ***Second***, when the LLM uses these retrieved fragments for reasoning, it does not actively evaluate the validity of the information. Consequently, without the ability to determine when to proactively stop retrieval based on known information or update the queries that need to be retrieved, it may lead to inefficiencies or the retrieval of irrelevant information. ***Third***, the effective parts within the retrieved text segments are very few, and excessive redundant information introduces noise, which can obscure important details and negatively impact the model's performance.

To this end, we propose **A**daptive **m**emory-**b**ased optimization for **e**nhanced **R**AG (**Amber**). Amber comprises three core components: Agent-based Memory Updater (AMU), Adaptive Information Collector (AIC), and Multi-granular Content Filter (MCF). These components work in unison to automatically integrate and update retrieved information as the LLM's memory, dynamically adjust the queries based on known information, and employ multi-granular content filtering during retrieval to retain useful information and reduce noise, thereby achieving outstanding performance. ***Firstly***, to address the issue in which each retrieval operates independently and lacks a summarizing memory of previous retrieval fragments, AMU employs a multi-agent collaborative approach. By coordinating various agents, AMU optimizes the LLM's current memory. This process ensures that the knowledge structure is continuously refined and enriched, effectively integrating all valuable information from previous retrieval steps. ***Secondly***, AIC utilizes the real-time memory generated by AMU to update the queries that need to be retrieved and decides when to stop retrieval. By automatically adjusting the retrieval process based on the accumulated knowledge, AIC ensures that subsequent retrievals are more targeted and efficient, effectively addressing the challenge of insufficient knowledge accumulation and avoiding unnecessary retrievals. ***Lastly***, we fine-tune the LLM to function as MCF to reduce noise during retrieval. MCF includes two levels of filtering capabilities. Firstly, it assesses the validity of the entire retrieved text segment and the query, determining whether the information is relevant and useful. Secondly, from the valid retrieved segments, it filters out irrelevant content and retains essential information. This approach effectively reduces redundant information and highlights crucial details, thereby enhancing the overall performance of the model.

In summary, our contributions are as follows.

- We propose the Agent-based Memory Updater, which uses a multi-agent approach to integrate information and form memory from previous retrievals, optimizing the LLM's memory.
- We develop the Adaptive Information Collector, which updates retrieval queries and decides when to stop retrieval, making the process more targeted and efficient.
- We introduce the Multi-granular Content Filter to reduce noise by filtering irrelevant content at multiple levels, enhancing model performance.
- Extensive experiments validate the effectiveness of Amber, showing significant improvements over existing methods in open-domain QA.

## 2 Related Work

**Open-domain QA** Open-domain Question Answering (OpenQA) (Voorhees et al., 1999) seeks to provide answers to questions expressed in natural language that are not restricted to a specify domain. Modern methods for Open QA tasks typically adopt the Retriever-and-Reader framework (Chen, 2017; Wang et al., 2024a). With the advancement of open-domain QA, multi-hop QA (Joshi et al., 2017; Yang et al., 2018) and long-form (Stelmakh et al., 2022; Lyu et al., 2024) have gradually emerged. This more complex QA further necessitates the system to extensively collect and contextualize from the multiple documents (typically through iterative processes) in order to address more intricate queries. In particular, (Khot et al., 2022; Khattab et al., 2022) suggested breaking down multi-hop queries into simpler single-hop queries, iteratively utilizing LLMs and the retriever to address these sub-queries, and then combining their results to construct a comprehensive answer. Unlike the decomposition-based method, other recent studies, such as (Yao et al., 2022) and (Trivedi et al., 2022), explored a technique that creates a logical sequence of reasoning steps with document

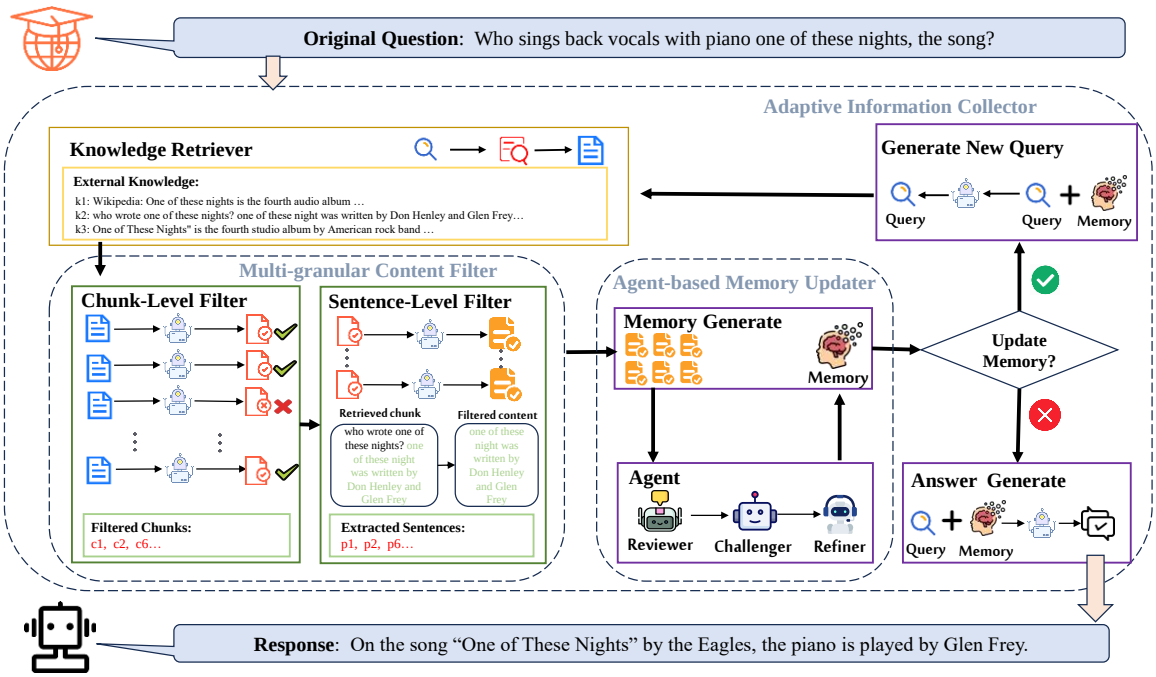



Figure 1: **Illustration of the Amber framework.** Amber is an adaptive Retrieval-Augmented Generation (RAG) approach incorporating three key components: the Adaptive Information Collector (AIC), the Multi-Granular Content Filter (MCF), and the Agent-Based Memory Updater (AMU). The MCF filters chunks irrelevant to the query and extracts the most useful sentences. Subsequently, the AMU updates the generated memory notes. Finally, the AIC evaluates the quality of the memory and determines whether further iterations are necessary.

retrieval. Additionally, (Jiang et al., 2023) proposed a method that involves iteratively fetching new documents when the tokens in the generated sentences exhibit low confidence, and (Jeong et al., 2024) proposed a retrieval strategy based on the complexity of the questions. However, The methods mentioned above neglect both the quality of retrieved documents and the generation of memory. Therefore, it is essential to propose a method aimed at enhanceing the quality of both retrieval and memory generation.

**Retrieval-Augmented Generation.** RAG has become essential for enhancing the response quality of large language models (LLMs). Early strategies (Izacard et al., 2023) relied on single-time retrieval, inputting the retrieved passages directly into LLMs to generate answers. However, these methods often fell short in complex tasks like multi-hop and long-form question answering, failing to capture sufficient information. To address these limitations, multi-time retrieval (Trivedi et al., 2022; Borgeaud et al., 2022) was explored, though it risked incorporating irrelevant data, leading to poor-quality responses. This led to the development of Adaptive RAG (ARAG), which dynamically adjusts retrieval strategies based on real-time feedback, determining optimal retrieval times and content. Key innovations include Flare (Jiang et al., 2023), which triggers new retrievals for low-confidence tokens, and Self-RAG (Asai et al., 2023), which uses self-reflective markers to assess content quality. These adaptive approaches enhance retrieval relevancy and accuracy, while agent-based models like ReAct (Yao et al., 2022) further augment RAG's flexibility and intelligence. Nevertheless, we argue that the above methods overlook the quality issues in both retrieval and agent generation, resulting in inaccurate answers.

## 3 Methods

In this section, we define the task and present an overview of our proposed method, Amber (illustrated in Figure 1). Following this, we provide a detailed explanation of each individual component.

### 3.1 Problem Formulation

RAG aims to enhance the generation quality of LLMs by integrating relevant information from an external document corpus $D = \{d_1, d_2, \ldots, d_n\}$. Given a user input $x$ or a query $q$, the core of RAG involves using a retriever $R$ to identify and select a subset of pertinent documents from $D$. The LLM then leverages both the original input and these retrieved documents to produce an improved output. Generally, this process seeks to achieve an output

$y$ based on the input and retrieved context.

### 3.2 Amber Overview

Figure 1 presents the architecture of Amber, an adaptive memory updating iterative RAG method. It consists of three main components: an Agent-based Memory Updater, an Adaptive Information Collector, and a Multi-granular Content Filter.

Given a query $q$, we initialize the LLM's memory $M_0$ as an empty set. Acting as the primary scheduler, the **Adaptive Information Collector** initiates the iterative loop. At each iteration $t$, we retrieve the top $k$ text chunks $C_t = \{c_1^t, c_2^t, \ldots, c_k^t\}$ from the corpus $D$ based on $q$. The **Multi-granular Content Filter** then assesses the relevance of each chunk $c_i^t$ to $q$ and filters out irrelevant content, retaining the most pertinent sub-paragraphs to form the refined document set $P_t$. Subsequently, the **Agent-based Memory Updater** employs three agents—the Reviewer, Challenger, and Refiner—to collaboratively integrate the new references $P_t$ with the previous memory $M_{t-1}$, producing the updated memory $M_t$. These agents ensure that the memory is optimized for answering $q$. The adaptive information collector then evaluates whether the LLM can adequately respond to $q$ using the current memory $M_t$. If not, it generates a new retrieval query $q_{t+1}$ based on $M_t$ and $q$ and proceeds to the next iteration. If the LLM can answer $q$ satisfactorily, the adaptive information collector terminates the loop. After the iterative process concludes, we use the final memory $M_T$ as context to generate the answer $\alpha \in A$ through the LLM's zero-shot in-context learning (ICL).

### 3.3 Agent-based Memory Updater (AMR)

In real-world scenarios, user queries vary significantly in complexity, necessitating the formulation of tailored strategies for each query. Enhanced LLM based on memory provide an effective solution to this challenge. Memory, which represents the information known to the LLM during retrieval, enables the model to determine retrieval strategies effectively. Among these, memory updating is a critical component. It requires the LLM to leverage historical and newly retrieved information to regenerate the latest memory aligned with the query.

Inspired by the concept of multi-agent collaboration, we propose an **Agent-based Memory Updater** framework, which employs a cooperative, multi-agent approach to memory updating. Specifically, AMR consists of three independent agents: the Reviewer, the Challenger, and the Refiner. Through iterative dialogue, these agents reflect upon and optimize the memory. The inputs to AMR include the current memory $m_t$, the retrieved passages $p_t$ for the current query $q_t$, and the original user query $q$. Based on these inputs, the LLM initially generates an updated memory $m_{t+1}$.

**Reviewer.** As the primary evaluator in the AMR framework, the Reviewer examines the proposed memory update $m_{t+1}$ using the current memory $m_t$, retrieved passages $p_t$, and user query $q$. The Reviewer assesses the correctness and relevance of $m_{t+1}$ to the user's intent, identifying strengths and weaknesses. By sharing evaluations with the Challenger and Refiner, the Reviewer facilitates collaborative refinement and coordinates strategies to ensure alignment with collective goals. This evaluation process ensures memory updates are rigorously reviewed, improving the LLM's retained information.

**Challenger.** Acting as the critical analyst, the Challenger builds upon the Reviewer's assessment by examining $m_{t+1}$, identifying potential flaws and overlooked constraints. Through interaction with the Reviewer and Refiner, the Challenger scrutinizes the validity of the memory update, raising probing questions about conflicts with existing knowledge or unmet user requirements. These interactions enable collective strategy adaptation, ensuring $m_{t+1}$ is robust and well-aligned with both the user query and knowledge base.

**Refiner.** As the agent responsible for implementing improvements, the Refiner synthesizes feedback from the Reviewer and Challenger to refine $m_{t+1}$. It translates critiques into concrete modifications, focusing on enhancing accuracy, clarity, and adherence to user query. The Refiner resolves issues identified by other agents, producing a revised $m_{t+1}$ that better satisfies objectives. Through collaboration with the Reviewer and Challenger, the Refiner streamlines the feedback loop and maintains modification records, contributing to an effective refinement cycle.

Through the complementary collaboration of the Reviewer, Challenger, and Refiner within AMR, the proposed method effectively leverages the strengths of each agent. This triadic interaction ensures that memory updates undergo rigorous evaluation, critical examination, and precise refinement. As a result, the updated memory $m_{t+1}$ becomes increasingly accurate, relevant, and aligned with the user's query across multiple iterative cycles.

### 3.4 Adaptive Information Collector (AIC)

We propose the **Adaptive Information Collector** as the primary scheduler to control the entire RAG workflow. The role of AIC is to evaluate whether the information currently available in the memory, generated by the AMU, is sufficient to answer the user query $q$.

Specifically, each iteration of AIC follows three key steps. The process initializes with the user query $q_0 = q$ and an empty memory $m_0 = \emptyset$. Firstly, AIC begins by retrieving the top $k$ text chunks $C_t = \{c_1^t, c_2^t, \ldots, c_k^t\}$ from the corpus $D$ based on the current query $q_t$ using a retrieval mechanism. Next, the query $q$, the retrieved text chunks $C_t$, and the current memory $m_t$ are input into the Agent-based Memory Updater (AMU), which generates an updated memory $m_{t+1} = \text{AMU}(q_t, C_t, m_t)$. Lastly, AIC then evaluates whether the updated memory $m_{t+1}$ contains sufficient information to fully answer the query $q$. If the memory is deemed sufficient, the iterative process terminates, and the latest memory $m_T$, along with the query $q$, are inputted into the LLM using in-context learning to produce the final answer $a$. However, if the updated memory $m_{t+1}$ is insufficient, AIC generates a refined query $q_{t+1} = \text{AIC}(q, q_t, m_{t+1})$. to target the missing information and proceeds to the next iteration. This iterative approach ensures that the final memory $m_T$ is comprehensive and well-aligned with the user's informational needs.

This iterative design allows AIC to dynamically refine queries and memory updates, ensuring that the final memory $m_T$ contains the necessary information to answer the user's query comprehensively.

### 3.5 Multi-granular Content Filter (MCF)

The Adaptive Information Collector, despite leveraging the filtering capabilities of the Agent-based Memory Updater and employing adaptive retrieval to refine the query $q_t$, often retrieves the top $k$ text chunks $C_t = \{c_1^t, c_2^t, \ldots, c_k^t\}$ that still include irrelevant information. These irrelevant parts can be categorized into two levels: chunk-level irrelevance, where an entire chunk $c_i$ may be unrelated to the query $q$, and sentence-level irrelevance, where even within a relevant chunk $c_i$, only a subset of the sentences may be pertinent to the query $q$, while the remainder constitutes noise.

Based on these insights, we used STRINC, CXMI metrics, and GPT-4 (detail see in appendix A) to generate a multi-granular content filter dataset and subsequently fine-tuned a LLM using multi-task learning to create the **Multi-granular Content Filter**, denoted as $F_c$. This content filter operates hierarchically, applying two levels of filtering to each chunk $c_i$.

At the first level, a chunk-level filtering prompt, formulated as $f_c(prompt_{chunk}, q, p_i)$, determines whether a chunk is relevant to $q$. If $f_c$ returns False, the chunk is directly discarded; otherwise, it progresses to the second level. The chunk-level filter is defined as:

$$f_c(prompt_{chunk}, q, c_i) = \begin{cases} \text{True}, & \text{if } c_i \text{ relevant to } q \\ \text{False}, & \text{if } c_i \text{ not relevant to } q \end{cases} \tag{1}$$

At the second level, a sentence-level evaluation is performed for chunks that pass the initial filter, where $p_i = f_c(prompt_{sentence}, q, c_i)$ assesses each sentence within the chunk to retain the relevant sentences. The output of this stage is a refined set of relevant sentences $P_t = \{p_1, p_2, \ldots, p_m\}$, where $p_i$ are the relevant sentences.

This hierarchical filtering approach significantly reduces noise in the retrieved information by isolating only the relevant content at both chunk and sentence levels. The MCF, ensures that AIC operates with higher precision, improving the quality and relevance of the memory $m_t$ in each iteration and, consequently, the overall performance.

## 4 Experimental Setup

In this section, we present the datasets, models, metrics, and implementation details. More experiment setup can see appendix A and B.

### 4.1 Datasets and Evaluation Metrics

To simulate a realistic scenario, where different queries have varying complexities, we use both the single-hop, multi-hop and long-form QA datasets simultaneously, in the unified experimental setting.

**Single-hop QA** For simpler queries, we use three benchmark single-hop QA datasets, which consist of queries and their associated documents containing answers, namely **1) SQuAD v1.1** (Rajpurkar, 2016), **2) Natural Questions** (Kwiatkowski et al., 2019) and **3) TriviaQA** (Joshi et al., 2017).

**Multi-hop QA** To consider more complex query scenarios, we use two benchmark multi-hop QA datasets, which require sequential reasoning

Table 1: **Performance comparison of Amber with baseline models.** The bold and underlined values indicate the best and second-best results across all models. Overall, Amber consistently achieves superior performance across all datasets, demonstrating its effectiveness in answering questions.

| | Methods | single-hop QA | | | | | | multi-hop QA | | | | Long-form QA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | SQuAD | | Natural Questions | | TriviaQA | | 2WikiMQA | | HotpotQA | | ASQA | |
| | | acc | f1 | acc | f1 | acc | f1 | acc | f1 | acc | f1 | str-em | str-hit |
| No Retrieval | NoR | 12.6 | 18.41 | 24.0 | 27.49 | 49.8 | 52.69 | 28.4 | 35.6 | 19.8 | 25.17 | 35.5 | 8.9 |
| Single-time RAG | Vanilla (Qwen2-7b) | 32.2 | 27.7 | 36.2 | 24.62 | 60.6 | 49.63 | 36.2 | 39.0 | 37.8 | 37.2 | 43.5 | 18.5 |
| | Vanilla (Llama3-8b) | 30.4 | 36.08 | 33.2 | 38.99 | 58.2 | 60.28 | 22.2 | 26.2 | 34.2 | 42.2 | 38.7 | 13.7 |
| | Vanilla (GPT-3.5) | 34.4 | 37.88 | 35.9 | 38.43 | 63.8 | 63.49 | 35.4 | 38.2 | 38.6 | 44.36 | 47.77 | 21.62 |
| | Self-Refine | 32.1 | 33.04 | 35.8 | 35.17 | 61.2 | 58.91 | 35.9 | 38.6 | 38.2 | 43.8 | 42.1 | 16.6 |
| | Self-Rerank | 31.1 | 35.19 | 34.3 | 39.05 | 60.7 | 59.84 | 34.8 | 32.1 | 35.6 | 42.2 | 35.0 | 11.4 |
| | Chain-of-note | 31.8 | 33.94 | 35.2 | 37.66 | 61.0 | 58.33 | 35.1 | 39.7 | 36.8 | 45.0 | 40.3 | 15.6 |
| Adaptive RAG | ReAct | 33.6 | 34.85 | 35.4 | 38.37 | 60.9 | 59.83 | 34.6 | 37.3 | 37.5 | 46.9 | 32.9 | 8.3 |
| | Self-RAG | 32.7 | 33.84 | 37.9 | 39.17 | 60.3 | 58.94 | 29.8 | 30.8 | 35.3 | 44.4 | 40.9 | 16.5 |
| | FLARE | 32.9 | 35.81 | 36.4 | 38.94 | 61.1 | 57.75 | 38.2 | 42.8 | 37.2 | 47.8 | 34.9 | 9.5 |
| | Adaptive-RAG | 33.0 | 38.3 | 44.6 | 47.3 | 58.2 | 60.7 | 46.4 | 49.75 | 44.4 | 52.56 | 42.1 | 15.8 |
| | Adaptive-Note | 29.0 | 33.61 | 40.0 | 45.38 | 59.6 | 59.72 | 39.4 | 39.1 | 39.0 | 46.6 | 43.7 | 17.7 |
| Ours | Amber (Qwen2-7b) | **36.8** | 38.43 | **47.8** | 49.84 | 65.8 | 62.77 | **56.0** | **52.73** | **52.6** | 51.13 | 49.7 | 25.2 |
| | Amber (Llama3-8b) | 34.6 | **39.37** | 44.2 | 50.49 | 63.6 | 62.79 | 43.8 | 43.5 | 45.8 | **53.72** | 44.7 | 18.8 |
| | Amber (GPT-3.5) | 35.8 | 39.06 | 47.4 | **52.01** | **66.8** | **66.08** | 46.7 | 45.95 | 47.4 | 53.55 | **51.3** | **26.3** |

over multiple documents, namely **1) 2WikiMultiHopQA (2WikiMQA)** (Ho et al., 2020) and **2) HotpotQA** (Yang et al., 2018). For both single-hop QA and multi-hop QA, we report the accuracy (**acc**) and F1-score (**f1**) as evaluation metrics, where acc measures if the predicted answer contains the ground-truth, and f1 measures the number of overlapping words between the predicted answer and the ground-truth.

**Long-form QA** We select an English dataset **ASQA** (Stelmakh et al., 2022). Specially, we use the ASQA dataset with 948 queries recompiled by ALCE (Gao et al., 2023) and apply ALCE's official evaluation metrics, involving String Exact Match (**str-em**) and String Hit Rate (**str-hit**).

### 4.2 Baseline&LLMs

We extensively compare three types of baselines: 1) No Retrieval (**NoR**), which directly feeds queries into LLMs to output answers without any retrieval process; 2) Single-time RAG (**STRAG**), which retrieves knowledge in a one-time setting to answer the original queries; 3) Adaptive RAG (**ARAG**), which leverages an adaptive forward exploration strategy to retrieve knowledge to enhance answer quality. For STRAG, we select Vanilla RAG, Chain-of-note (Yu et al., 2023), Self-Refine, and Self-Rerank are simplified from Self-RAG (Asai et al., 2023). For ARAG, we include five recent famous methods for comparison - FLARE (Jiang et al., 2023), Self-RAG, ReAct (Yao et al., 2022), Adaptive-RAG (Jeong et al., 2024) and Adaptive-Note (Wang et al., 2024b). Additionally, we conduct experiments on multiple LLMs, including Qwen2-7b (Yang et al., 2024), Llama3-8b (Touvron et al., 2023a) and GPT-3.5 (OpenAI gpt-3.5-turbo-instruct). We default to using Llama3-8b as the Multi-granular Content Filter LLM, detail experiment setting about multi-filter content see appendix A. Unless otherwise specified, Llama3-8b was employed as the default model.

## 5 Results and Analysis

In this section, we evaluate our proposed framework, Amber, on six real-world datasets and compare it against several baselines, including No retrieval, single-time and adaptive RAG methods.

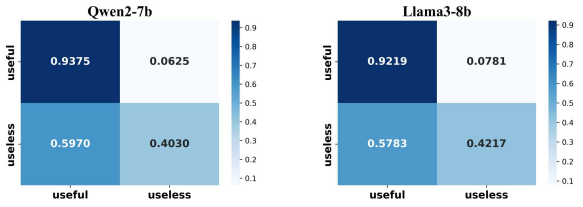
### 5.1 Main Results.

We implemented the Amber on six datasets. The comparison with baseline models is summarized in Table 1. Key observations are as follows:

**Amber vs. Single-time RAG.** Results show that our method surpassed all STRAG for all six QA datasets. Meanwhile, it is noteworthy that our method outperformed Vanilla by over 30% on the Natural Questions, 2WikiMQA, and HotpotQA datasets. Even on the remaining three datasets, it still achieved an approximate 10% improvement. These achievements highlight its superiority and effectiveness. An intuitive explanation is that STRAG heavily depends on the quality of one-time retrieval, whereas our method can adaptively explore more knowledge in the corpus and filter useless chunks and irrelevant sentence in chunk. Therefore, it is able to demonstrate that our method preserves more and more effective knowledge.

**Amber vs. Adaptive RAG.** In Table 1, we conduct an in-depth comparison of our approach with several existing ARAG models,including FLARE, Self-RAG, ReAct, Adaptive-RAG and Adaptive-Note. Our method consistently outperforms baselines in single-hop, multi-hop, and long-form QA tasks, particularly in accuracy. Even compared to the state-of-the-art ARAG method, it improves by over 10%, demonstrating its superiority, effectiveness, and robustness. We provide an in-depth analysis of the baseline limitations and the factors contributing to our success. **First**, ReAct and Flare relies on LLMs' internal knowledge to guide retrieval decisions, but its inherent overconfidence (Zhou et al., 2023) may hinder retrieval efficacy by neglecting existing knowledge. In contrast, our method employs a greedy strategy to first gather information extensively, followed by a careful assessment of whether to incorporate new, useful knowledge into the existing framework. This process optimizes knowledge extraction and significantly enhances response accuracy. **Second**, Self-RAG faces challenges in training effective models for complex tasks due to numerous classifications such as labeled inputs, retrieved paragraphs, and output categorizations. Unlike this approach, our Multi-granular Content filter training strategy is relatively simple, yet it maximizes the utilization of valuable information through multiple iterations and agent-based memory. **Third**, The Adaptive-RAG method adapts retrieval strategies based on query complexity, and Adaptive-Note generates new memory in each iteration until memory growth stabilizes. However, both of them neglect passage quality, which affects answer accuracy. Instead, our method focuses on the importance of retrieving relevant paragraphs, aiming to minimize the impact of irrelevant information on the LLM's decision-making when answering questions.

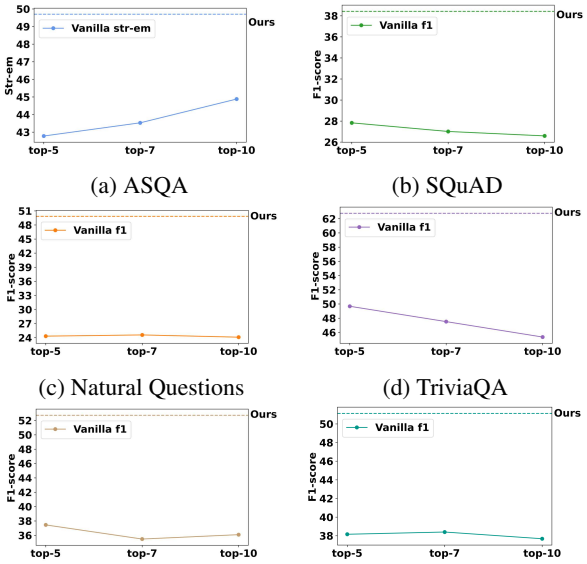
## 5.2 Classifier Performance




(a) Qwen2-7b (b) Llama3-8b

Figure 2: **Confusion matrix for fine-tuned LLMs. Our Fine-Tuned LLMs serve as excellent classifiers.**

To understand the performance of the proposed classifier, we analyze its effectiveness across two LLM models. As shown in figure 2, whether in Llama3-8b or Qwen2-7b, our Amber classifier, achieves over 90% accuracy in classifying useful retrieved passages. Furthermore, it successfully excludes more than 40% of negative retrieved knowledge, significantly improving the quality of the knowledge and eliminating irrelevant information.




(a) ASQA (b) SQuAD

(c) Natural Questions (d) TriviaQA

(e) 2WikiMQA (f) HotpotQA

Figure 3: **Results of the in-depth comparison under a fair top-$k$.**

## 5.3 In-depth comparison under a fair top-k

Under the same raw top-k setting, ARAG methods generally retrieve more passages compared to single-step methods. Unfortunately, while we can specify the top-k value for each step, the inherent retrieval uncertainly in ARAG prevents us from controlling the total number of retrieved passages. To ensure a fairer performance comparsion, we address the discrepancy by calculating the average number of unique passages retrieved per sample across all adaptive steps, which we term the fair top-k. Figure 3 illustrates the overall performance of Vanilla RAG under this fair-top-k setting. It is evident that as the number of retrieved passages increases, the Vanilla method shows little to no significant improvement. These findings further emphasize the superiority of our method.

## 5.4 Ablation Study

To analyze the contributions of components in the proposed Amber method, particularly the fine-tuning of the Multi-granular Content Filter and the Agent-based Memory Updater in Adaptive information Collector, we conducted an ablation study

| LLM | Qwen2-7b | | LLama3-8b | | GPT3.5 | |
|---|---|---|---|---|---|---|
| metrix | acc | f1 | acc | f1 | acc | f1 |
| Amber | **56.0** | **52.73** | **43.8** | **43.5** | **46.7** | **45.95** |
| **Multi-granular Content Filter** | | | | | | |
| w/o CF | 52.2 | 49.78 | 41.5 | 40.51 | 43.5 | 42.37 |
| w/o SF | 55.1 | 50.67 | 42.8 | 42.17 | 45.2 | 43.84 |
| w/o ALL | 51.6 | 48.39 | 40.9 | 40.12 | 42.4 | 41.65 |
| **Agent-Based Memory Updater** | | | | | | |
| w/o AM | 53.5 | 50.97 | 42.7 | 41.63 | 44.3 | 42.85 |

Table 2: **Ablation study on the 2WikiMQA dataset.**

| top-$k$ | 2WikiMQA | | HotpotQA | | ASQA | |
|---|---|---|---|---|---|---|
| | (acc) | (f1) | (acc) | (f1) | (str-em) | (str-hit) |
| **Vanilla** | | | | | | |
| top-3 | 36.3 | 36.82 | 35.9 | 37.8 | 42.5 | 17.5 |
| top-5 | **37.0** | **37.45** | 37.6 | 38.16 | 42.78 | 18.14 |
| top-7 | 35.6 | 35.48 | **39.8** | **38.4** | **43.53** | **17.93** |
| **Ours** | | | | | | |
| top-3 | 53.2 | 40.3 | 50.9 | 49.4 | 48.3 | 23.5 |
| top-5 | **56.0** | **42.73** | 52.6 | **51.13** | 49.7 | 25.2 |
| top-7 | 55.2 | 41.7 | **52.8** | 51.02 | **50.1** | **25.6** |

Table 3: **Amber with different top-$k$ with Qwen2-7b**.

| max-iter | 2WikiMQA | | HotpotQA | | ASQA | |
|---|---|---|---|---|---|---|
| | (acc) | (f1) | (acc) | (f1) | (str-em) | (str-hit) |
| 1 | 53.2 | 38.95 | 50.4 | 49.73 | 45.3 | 20.9 |
| 2 | 55.3 | 40.72 | 51.8 | 50.08 | 47.4 | 23.2 |
| 3 | **56.0** | **42.73** | **52.6** | **51.13** | **49.7** | **25.2** |

Table 4: **Amber with varying max iterations in AIC.**

on the 2WikiMQA dataset. The above components comprise three key components:

- **Chunk-Level Filter (CF):** Input: Query and retrieved chunk. Output: useful / useless.
- **Sentence-Level Filter (SF):** Input: query and filtered chunk. Output: filtered sentences.
- **Agent-Based Memory Updater (AMU):** whether to use agent-based method for memory.

**Effect of the Multi-granular Content Filter.** To evaluate the contribution of each Multi-granular Content Filter component, we systematically removed one type of content. Additionally, removing both two types (*w/o ALL*) effectively disables the content filter stage. The results, presented in Table 2, show that the model achieves its best performance when both two filter strategy are used together. Conversely, removing any single type of data leads to a noticeable decline in performance, highlighting the importance of each component in enhancing the framework's overall effectiveness. Interestingly, the performance when Sentence-Level Filter (*w/o SF*) are excluded remains higher than when Chunk-Level Filter (*w/o CF*) are omitted. This indicates that, The Chunk-Level Filter plays a more significant role in our approach, effectively filtering out irrelevant chunk information to a large extent.

**Impact of Agent-based Memory Updater.** To examine the role of Agent-based Memory Updater(*w/o AMU*) in Amber, we conducted an ablation experiment by removing the AMU module. As shown in Table 2, removing this module significantly decreases performance. This highlights the agent's critical role in memory generation. The agent ensures the creation of more efficient memory, thereby enabling the LLM to provide more accurate responses.

The ablation study highlights the importance of each content filter component and the agent-based memory module. Multi-granular Content Filter and Agent-based Memory Updater significantly enhance the performance of the Amber framework.

### 5.5 Parameter Analysis

**Impact of top-k.** In Table 3, we present a comparison between our method and Vanilla RAG across different top-k settings. The results demonstrate that our approach consistently outperforms Vanilla RAG under the same top-k conditions, highlighting its robustness in achieving reliable improvements regardless of the number of retrieved passages. Additionally, in most cases, the performance of our system improves as the number of retrieved passages (top-k) per step increases.

**Impacts of max iterations** As shown In Table 4, performance on these complex QA datasets improves with increasing iterations, peaking at 3. We therefore recommend setting the max iterations to 3 for optimal performance.

## 6 Conclusion

In this paper, we propose a novel RAG method, Amber, based on memory-adaptive updates. Our approach introduces a collaborative multi-agent memory updating mechanism, combined with an adaptive retrieval feedback iteration and a multi-granular filtering strategy. This design enables efficient information gathering and adaptive updates, significantly improving answer accuracy while reducing hallucinations. We validated Amber and its core components across several open-domain QA datasets. Extensive experiments prove the superiority and effectiveness of Amber.

**Limitation.** Although Amber has made significant progress in open-domain question answering with RAG, there are still some limitations. First, the framework requires multiple fine-tuning steps to train the Multi-granular Content Filter, which necessitates the collection of a substantial amount of data, as well as considerable computational resources and time. Second, since Amber requires multiple accesses to the LLM, answering each question takes more time compared to the vanilla approach. In future work, we plan to design a more time-efficient and generalizable fine-tuning strategy to improve the quality of open-domain question answering, thereby enhancing the overall effectiveness of Amber.

## References


Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

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.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

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

D Chen. 2017. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. How is chatgpt's behavior changing over time? *arXiv preprint arXiv:2307.09009*.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. *arXiv preprint arXiv:2305.14627*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv preprint arXiv:2212.14024*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

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.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115*.

Yuanjie Lyu, Zhiyu Li, Simin Niu, Feiyu Xiong, Bo Tang, Wenjin Wang, Hao Wu, Huanyong Liu, Tong Xu, and Enhong Chen. 2024. Crud-rag: A comprehensive chinese benchmark for retrieval-augmented generation of large language models. *ACM Transactions on Information Systems*.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. 2021. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*.

P Rajpurkar. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. Asqa: Factoid questions meet long-form answers. *arXiv preprint arXiv:2204.06092*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82.

Hongru Wang, Boyang Xue, Baohang Zhou, Tianhua Zhang, Cunxiang Wang, Guanhua Chen, Huimin Wang, and Kam-fai Wong. 2024a. Self-dc: When to retrieve and when to generate? self divide-and-conquer for compositional unknown questions. *arXiv preprint arXiv:2402.13514*.

Ruobing Wang, Daren Zha, Shi Yu, Qingfei Zhao, Yuxuan Chen, Yixuan Wang, Shuo Wang, Yukun Yan, Zhenghao Liu, Xu Han, et al. 2024b. Retriever-and-memory: Towards adaptive note-enhanced retrieval-augmented generation. *arXiv preprint arXiv:2410.08821*.

Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. 2023. Learning to filter context for retrieval-augmented generation. *arXiv preprint arXiv:2311.08377*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. 2024. Qwen2 technical report. *ArXiv*, abs/2407.10671.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2023. Making retrieval-augmented language models robust to irrelevant context. *arXiv preprint arXiv:2310.01558*.

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu. 2023. Chain-of-note: Enhancing robustness in retrieval-augmented language models. *arXiv preprint arXiv:2311.09210*.

Kaitlyn Zhou, Dan Jurafsky, and Tatsunori Hashimoto. 2023. Navigating the grey area: How expressions of uncertainty and overconfidence affect language models. In *Conference on Empirical Methods in Natural Language Processing*.

## A Multi-granular Content Filter

In Amber, for the Multi-granular content filter, we finetune the LLaMA 3-8B and Qwen 2-7B models using the LLamaFactory framework. Specifically, for the query-reference classification task, we finetune the model into two categories: useful and useless, retaining only the useful ones. For the context filter, we use only the extracted content from the original passages and feed it into the LLMs. Both fine-tuning processes are conducted over 2 epochs, with the per-device training batch size set to 4.

**Chunk-Level Filter** The accuracy of responses generated by LLMs can be significantly compromised by noisy retrieved contexts (Yoran et al., 2023). To mitigate this, we introduce the Chunk-Level Filter module to enhance response accuracy and robustness. This module utilizes LLMs to filter out irrelevant knowledge. Rather than directly querying an LLM to identify noise, we incoporate a Natural Language Inference (NLI) framework (Bowman et al., 2015) for this purpose. Specially, for a query $q$ and retrieved reference $r$, the NLI task evaluates whether the knowledge contains reliable answers, or usefule information aiding the response to the question. This results in a judgment $j$ categorized as useful or useless. The operation of the Chunk-Level Filter can be mathematically represented as :

$$F_\theta(q, r) \rightarrow j \in \{\text{useful}, \text{useless}\} \quad (2)$$

Knowledge is retained if the NLI result is classified as useful, and the reference is discarded when the NLI result is classified as useless. The NLI training dataset is constructed semi-automatically. We provide task instruction, query $q$, along with retrieved reference $r$ as prompt to GPT-4, which then generated a brief explanation $e$ and a classification result $j$. The prompt template is as follows:

> **[Instruction]:** Your task is to solve the NLI problem: given the premise in [Knowledge] and the hypothesis that "The [Knowledge] contains reliable answers aiding the response to [Question]". You should classify the response as useful and useless.
> **[Question]:**{Question}
> **[Knowledge]:**{Knowledge}
> **[Format]:**{Explanation}{NLI result}

**Sentence-Level Filter** Followed by previous work (Wang et al., 2023), we use the STRINC measure for single-hop QA datasets and CXMI for multi-hop datasets. We train the sentence-Level Filter models $M_{slf}$, using context filtered with above two measures. To create training data for $M_{slf}$, for each training sample with query $q$, we concatenate the retrieved passages $P$ and query $q$, then, we apply the filter method $f$ to obtain filtered context $t_{output}$ as output. The $s$ is the sentence of the retrieved passages, and $t_{output}$ can be represented as:

$$t_{\text{output}} = [s_{text} \mid s_{strinc} == 1] \quad (3)$$

$$t_{\text{output}} = [s_{text} \mid s_{cxmi} >= threshold] \quad (4)$$

We train $M_{slf}$ by feeding in query and retrieved passages $P$, and ask it to generate filtered context.

## B Retriever & Corpus

To ensure a fair comparison of all baselines, we align the retriever and corpus across all methods for each dataset. For both single-hop and multi-hop datasets, we employ BM25 (Robertson et al., 1995), implemented in the search tool Elasticsearch, as the foundational retriever. For the external document corpus, we use the Wikipedia corpus preprocessed by (Karpukhin et al., 2020) for single-hop datasets, and the preprocessed corpus by (Trivedi et al., 2022) for multiple-hop datasets. For long-form ASQA dataset, we employ dense retriever GTR-XXL (Ni et al., 2021) and use the corpus provided by ALCE, consisting of the 2018-12-20 Wikipedia snapshot, segmented into 100-word passages.

## C Implementation Details

For computing resources, we utilize NVIDIA 4090 GPUs with 24GB of memory. Additionally, due to the frequent access to the LLM, we employ VLLM as the inference framework. The software stack includes Python 3.10.15, VLLM 0.6.3.post1, PyTorch 2.5.0, and CUDA 12.1.

## D Detailed prompt

We present all the prompts used in our method in Tables A3 and A4. In Table A3, we detail the prompt for the Multi-granular Content Filter. Specifically, at the Memory Initialization stage, query represents the original query q, and refs refers to the retrieved k passages obtained by feeding the original query q into the retriever. At the Iterative Information Collection stage, query still represents the original query q, and note refers to

the content of the optimal memory $M_{opt}$. Additionally, as mentioned in the main text, LLMs tend to ask similar questions if previous ones were not well resolved. To address this, we introduce the already-asked questions list query log to avoid repetition. At the Note-Updating stage, query still refers to q, while refs represents new retrieved k passages based on the updated queries, and note refers to $M_{opt}$. In the Memory Updating phase, query represents the original query q, while best note and new note represent $M_{opt}$ and $M_{cur}$, respectively.

In the Multi-granular Content Filter stage, for the Chunk-Level Filter, External_knowledge refers to the retrieved k passages, from which we filter out useless passages, retaining only the useful ones. Next, for the Sentence-Level Content Filter, context refers to each useful passage. After passing through this filter, we extract important sentences from the passages to generate the answer.

In the Agent-based Memory Update, we assume three roles in the memory generation process: reviewer, challenger, and refiner. The reviewer evaluates the strengths and weaknesses of the note memory based on the query and refs. The challenger, using the reviewer's feedback, provides suggestions to revise and enhance the memory. Finally, the refiner uses both the reviewer's insights and the challenger's suggestions to refine and generate the new memory. In the final memory updating phase, we compare the new memory with the initialized memory to select the best memory.

**Prompt of the Memory Initialization Stage**

**Instruction:**
Based on the provided document content, write a note. The note should integrate all relevant information from the original text that can help answer the specified question and form a coherent paragraph. Please ensure that the note includes all original text information useful for answering the question.

Question to be answered: {query}
Document content: {refs}

Please provide the note you wrote:

**Prompt of the Iterative query rewritten Stage**

**Instruction:**
Task: Based on the notes, propose a new question. The new question will be used to retrieve documents to supplement the notes and help answer the original question. The new question should be concise and include keywords that facilitate retrieval. The new question should avoid duplication with the existing question list.

Original question: {query}
Notes: {note}
Existing question list: {query_log}

Provide your new question,you MUST reply with the new question on the last line, starting with "### New Question".

**Prompt of the Chunk-Level Filter**

**Instruction:**
You are an advanced AI model specialized in understanding the Natural Language Inference (NLI) tasks. Your task is to do the NLI problem. The premise is [External Knowledge]. The hypothesis is "There exist clear and unambiguous answer in the [External Knowledge] that can convincingly and soundly answer the Question." Your response should be in one of (useful,useless).

External Knowledge: {External_Knowledge}
Question: {Question}

Now give me the NLI result, which 1. should be one of (useful,useless). 2.Please strictly following this json format and fill xxx with your answer. 3. Please notice the Escape Character and keep correct format. 4. Please just give me the concise Json response and no ther redundant words. 5. The output should not appear Here is the NLI result, Just strictly follow the format below:
{"NLI result":"xxx"}

**Prompt of the Sentence-Level Filter**

**Instruction:**
You are an AI model specialized in extracting helpful sentences from a given context. Your task is to extract helpful sentences while removing irrelevant or unhelpful ones based on the provided question and context.

Question: {query}
context: {context}

Now provide the extracted helpful sentences, which should include only valid and relevant sentences from the context.

Table 5: All prompts of Memory initialize, query rewritten and Multi-granular Content Filter.

**Prompt of the reviewer in Agent-based memory**

**Instruction:**
Task: Analyze the relationship between the query, retrieved documents, and notes. Identify the strengths and weaknesses of how well the notes align with the query and incorporate the information from the retrieved documents. Highlight areas where the notes effectively cover the query and the references, as well as areas where they could be improved to better address the query or utilize the information from the references.
Question: {query}
retrieved documents: {refs}
note: {note}
Provide an analysis of the notes with a focus on the strengths and weakness:

**Prompt of the challenger in Agent-based memory**

**Instruction:**
Based on the provided reviewer information, provide specific and actionable suggestions to improve the notes. The goal is to ensure the notes comprehensively and accurately address the query while fully utilizing relevant information from the retrieved documents.
Question: {query}
retrieved documents: {refs}
Notes: {note}
reviewer information: {review_info}
Provide detailed suggestions to revise and enhance the notes:

**Prompt of the refiner in Agent-based memory**

**Instruction:**
Refine the provided notes based on the reviewer information and suggestions. The goal is to ensure the notes are improved to better address the query and fully utilize the relevant information from the retrieved documents.
Question: {query}
retrieved documents: {refs}
Notes: {note}
reviewer information: {review_info}
suggestions: suggestions
Provide the refined notes that incorporate the feedback from the reviewer information and suggestions:

**Prompt of the memory updating**

**Instruction:**
Task: Please help me determine which note is better based on the following evaluation criteria:
1. Contains key information directly related to the question.
2. Completeness of Information: Does it cover all relevant aspects and details?
3. Level of Detail: Does it provide enough detail to understand the issue in depth?
4. Practicality: Does the note offer practical help and solutions?
Please make your judgment adhering strictly to the following rules:
- If Note 2 has significant improvements over Note 1 based on the above criteria, return {“status": “True"} directly; otherwise, return {“status": “False"} .
Question: {query}
Provided Note 1: {best_note}
Provided Note 2: {new_note}
Based on the above information, make your judgment without explanation and return the result directly.

Table 6: All prompts of Agent-based Memory Update