M-RAG: Reinforcing Large Language Model Performance through Retrieval-Augmented Generation with Multiple Partitions

Zheng Wang[1], Shu Xian Teo[1], Jieer Ouyang[1], Yongjun Xu[1], Wei Shi[1]
[1]Huawei Technologies, Co., Ltd.
{wangzheng155,teo.shu.xian,ouyang.jieer,xuyongjun6,w.shi}@huawei.com

Abstract

Retrieval-Augmented Generation (RAG) enhances Large Language Models (LLMs) by retrieving relevant memories from an external database. However, existing RAG methods typically organize all memories in a whole database, potentially limiting focus on crucial memories and introducing noise. In this paper, we introduce a multiple partition paradigm for RAG (called M-RAG), where each database partition serves as a basic unit for RAG execution. Based on this paradigm, we propose a novel framework that leverages LLMs with Multi-Agent Reinforcement Learning to optimize different language generation tasks explicitly. Through comprehensive experiments conducted on seven datasets, spanning three language generation tasks and involving three distinct language model architectures, we confirm that M-RAG consistently outperforms various baseline methods, achieving improvements of 11%, 8%, and 12% for text summarization, machine translation, and dialogue generation, respectively.

1 Introduction

Introduced by (Lewis et al., 2020), Retrieval-Augmented Generation (RAG) represents a paradigm within the domain of Large Language Models (LLMs) to augment generative tasks. More specifically, RAG incorporates an initial retrieval step where LLMs query an external database to acquire relevant information before progressing to answer questions or generate text. This process not only guides the subsequent generation step but also guarantees that the responses are firmly anchored in the retrieved information (referred to as memories). Consequently, it enhances LLM performance, and has attracted growing research interests (Gao et al., 2023) in recent years.

While the majority of existing studies (Asai et al., 2023; Cheng et al., 2023b; Ma et al., 2023) adopt a retrieval approach that considers *a database as a whole*, which tends to yield a coarse-grained retrieval. The collective organization of all memories may hinder the focus on crucial memories and introduce noise, particularly due to the inherent challenges of Approximate k-Nearest Neighbor (AKNN) search when applied to large datasets. In this context, we investigate a retrieval approach that aims to search within a partition of the database, corresponding retrieval at a fine-grained level, which is designed to enhance the generation process by targeting specific memories. Moreover, in quite a few vector database systems, database partitions are regarded as fundamental units for analysis. This facilitates the construction and maintenance of index structures (Pan et al., 2023), ensures the protection of user privacy data (stored in specific partitions with access rights) (Xue et al., 2017), and supports distributed architectures (Guo et al., 2022). Therefore, in this work, we propose to take *a partition as a basic entity* in the execution of RAG, which is less explored in current methods.

We discuss our proposal with a motivating experiment illustrated in Figure 1. We investigate various strategies for partitioning a database (elaborated in Section 3.1), and perform RAG with varying the number of partitions for three generation tasks: summarization, translation, and dialogue generation, where we explore all partitions for the retrieval, and the best result (assessed based on a development set) across different partitions is reported. We observe that the optimal performance is typically not achieved through retrieval based on the entire database (#Partitions = 1). This observation inspires us to investigate a novel RAG setting with multiple partitions. To achieve this, the task should address three significant challenges, summarized below. (1) Determining a strategy for partitioning a database and the number of partitions. (2) Developing a method for selecting a suitable partition for a given input query to discover effective memories. (3) Enhancing memory quality,

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1966–1978

M-RAG: Tăng cường hiệu suất mô hình ngôn ngữ lớn thông qua Tạo ra sự tăng cường truy xuất với nhiều phân vùng

Trịnh Vương1 , Thư Hiền Teo1 ,Jieer Âu Dương1 , Yongjun Xu1,Vệ Sĩ1
1Công ty TNHH Công nghệ Huawei
{wangzheng155,teo.shu.xian,ouyang.jieer,xuyongjun6,w.shi}@huawei.com

Tóm tắt

Retrieval-Augmented Generation (RAG) tăng cường các mô hình ngôn ngữ lớn (LLM) bằng cách lấy lại những ký ức có liên quan từ cơ sở dữ liệu bên ngoài. Tuy nhiên, các phương pháp RAG hiện có thường sắp xếp tất cả các ký ức trong một tổng thể cơ sở dữ liệu, có khả năng hạn chế sự tập trung vào những điều quan trọng ký ức và giới thiệu tiếng ồn. Trong bài báo này , chúng tôi giới thiệu một mô hình phân vùng nhiều cho RAG (gọi là M-RAG), trong đó mỗi cơ sở dữ liệu phân vùng đóng vai trò là đơn vị cơ bản cho việc thực thi RAG. Dựa trên mô hình này, chúng tôi đề xuất một khuôn khổ mới tận dụng LLM với Học tăng cường đa tác nhân để tối ưu hóa các tác vụ tạo ngôn ngữ khác nhau một cách rõ ràng . Thông qua các thí nghiệm toàn diện được tiến hành trên bảy tập dữ liệu, trải dài ba nhiệm vụ tạo ngôn ngữ và liên quan đến ba kiến trúc mô hình ngôn ngữ riêng biệt, chúng tôi khẳng định rằng M-RAG luôn vượt trội hơn nhiều phương pháp cơ sở khác nhau, đạt được những cải tiến của 11%, 8% và 12% cho tóm tắt văn bản, dịch máy và tạo hội thoại, tương ứng.

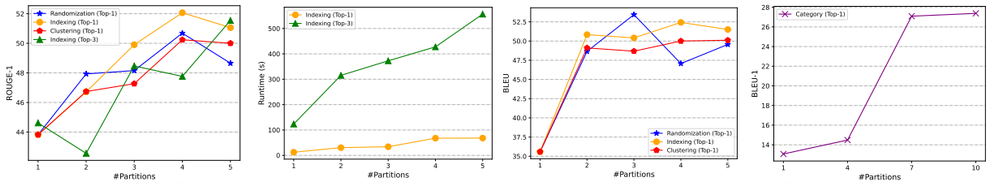
1 Giới thiệu

Được giới thiệu bởi (Lewis và cộng sự, 2020), Thế hệ tăng cường truy xuất (RAG) đại diện cho mô hình trong phạm vi Ngôn ngữ lớn Các mô hình (LLM) để tăng cường các nhiệm vụ tạo ra. Thêm cụ thể, RAG kết hợp một truy xuất ban đầu bước mà LLM truy vấn cơ sở dữ liệu bên ngoài để thu thập thông tin có liên quan trước khi tiến hành trả lời câu hỏi hoặc tạo văn bản. Quá trình này không không chỉ hướng dẫn bước thế hệ tiếp theo mà còn đảm bảo rằng các phản hồi được neo chặt trong thông tin thu được (được gọi là bộ nhớ). Do đó, nó nâng cao hiệu suất LLM, và đã thu hút được sự quan tâm nghiên cứu ngày càng tăng (Gao et al., 2023) trong những năm gần đây.

Trong khi phần lớn các nghiên cứu hiện có (Asai et al., 2023; Cheng và cộng sự, 2023b; Ma và cộng sự, 2023) áp dụng một cách tiếp cận truy xuất coi cơ sở dữ liệu như một tổng thể, có xu hướng tạo ra sự truy xuất thô sơ . Tổ chức tập thể của tất cả các ký ức có thể cản trở sự tập trung vào các ký ức quan trọng và đưa vào tiếng ồn, đặc biệt là do những thách thức vốn có của K-Láng giềng gần nhất (AKNN) tìm kiếm khi áp dụng cho các tập dữ liệu lớn. Trong bối cảnh này, chúng tôi điều tra một phương pháp tiếp cận truy xuất nhằm mục đích tìm kiếm trong một phân vùng của cơ sở dữ liệu, truy xuất tương ứng ở mức độ chi tiết, được thiết kế để tăng cường quá trình tạo ra bằng cách nhắm mục tiêu vào các ký ức cụ thể. Hơn nữa, trong Trong khá nhiều hệ thống cơ sở dữ liệu vector, phân vùng cơ sở dữ liệu được coi là đơn vị cơ bản để phân tích. Điều này tạo điều kiện thuận lợi cho việc xây dựng và bảo trì của các cấu trúc chỉ mục (Pan et al., 2023), đảm bảo bảo vệ dữ liệu riêng tư của người dùng (được lưu trữ trong phân vùng có quyền truy cập) (Xue et al., 2017), và hỗ trợ các kiến trúc phân tán (Guo et al., 2022). Do đó, trong công trình này, chúng tôi đề xuất thực hiện một phân vùng như một thực thể cơ bản trong quá trình thực hiện RAG, ít được khám phá bằng các phương pháp hiện tại.

Chúng tôi thảo luận đề xuất của mình với một thí nghiệm thúc đẩy được minh họa trong Hình 1. Chúng tôi điều tra các chiến lược khác nhau để phân vùng cơ sở dữ liệu (được trình bày chi tiết trong Phần 3.1) và thực hiện RAG bằng cách thay đổi số lượng phân vùng cho ba thế hệ nhiệm vụ: tóm tắt, dịch thuật và đối thoại thế hệ, nơi chúng ta khám phá tất cả các phân vùng cho truy xuất và kết quả tốt nhất (được đánh giá dựa trên bộ phát triển) trên các phân vùng khác nhau được báo cáo. Chúng tôi quan sát thấy hiệu suất tối ưu thường không đạt được thông qua việc truy xuất dựa trên trên toàn bộ cơ sở dữ liệu (#Partitions = 1). Quan sát này truyền cảm hứng cho chúng tôi để điều tra một RAG mới thiết lập với nhiều phân vùng. Để đạt được điều này, nhiệm vụ này phải giải quyết ba thách thức quan trọng, tóm tắt dưới đây. (1) Xác định chiến lược cho phân vùng cơ sở dữ liệu và số lượng phân vùng. (2) Phát triển một phương pháp để lựa chọn một phù hợp phân vùng cho một truy vấn đầu vào nhất định để khám phá những ký ức hiệu quả. (3) Nâng cao chất lượng bộ nhớ,

Biên bản Hội nghị thường niên lần thứ 62 của Hiệp hội Ngôn ngữ học tính toán (Tập 1: Bài báo dài), trang 1966–1978



(a) Summ. (ROUGE-1) (b) Summ. (Runtime) (c) Machine translation (d) Dialogue generation

Figure 1: Comparison with database partitioning strategies for language generation tasks.

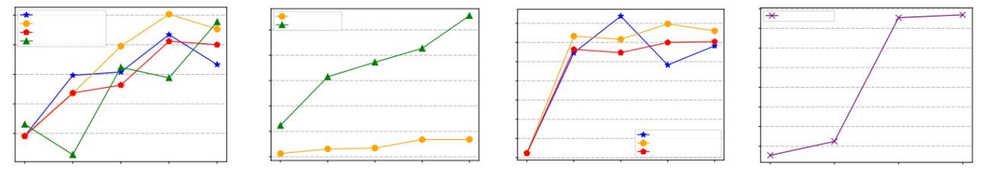
including inherent issues such as hallucination, or irrelevant context, which can impact the grounding of LLM generation.

Building upon the aforementioned discussion, we introduce a new solution called M-RAG, designed to facilitate RAG across multiple partitions of a database. M-RAG addresses all of the three challenges. For (1), we draw insights from the literature on vector database management (Pan et al., 2023; Han et al., 2023) and assess various strategies, namely Randomization (Indyk and Motwani, 1998), Clustering (Jegou et al., 2010), Indexing (Malkov et al., 2014; Malkov and Yashunin, 2018), and Category (Gollapudi et al., 2023), through empirical studies. The effectiveness of these strategies, along with the corresponding number of partitions, is evaluated across different generative tasks on a development set in our experiments. For (2), with multiple partitions at play, we formulate partition selection as a multi-armed bandit problem (Slivkins et al., 2019). In this context, an agent, denoted as Agent-S, iteratively selects one among several partitions. The characteristics of each partition are only partially known at the time of selection, and Agent-S gains a better understanding over time by maximizing cumulative rewards in the environment. To optimize the decision policy, we leverage reinforcement learning with a carefully designed Markov Decision Process (MDP). For (3), after selecting a partition and obtaining memories for generation, we introduce another agent, denoted as Agent-R. This agent generates a pool of candidate memories iteratively through the use of LLMs. Once a candidate is selected, Agent-R evaluates its quality by demonstrating it to generate a hypothesis. The identification of a high-quality hypothesis determined by a specific performance metric, triggers a boosting process, where it signals the exploration and replacement of the previous memory with a superior one, and continues the process. Further, we integrate the efforts of Agent-S and Agent-R through multi-agent reinforcement learning. With a shared objective of enhancing text generation for a given input query, they are jointly optimized through end-to-end training.

Our contributions can be summarized as follows: (1) we propose a multiple partition paradigm for RAG, aiming to facilitate fine-grained retrieval and concentrate on pivotal memories to enhance overall performance. In addition, the utilization of multiple partitions benefits other aspects of RAG, including facilitating the construction and maintenance of indices, protecting user privacy data within specific partitions, and supporting distributed parallel processing across different partitions. (2) We introduce M-RAG, a new solution based on multi-agent reinforcement learning that tackles the three challenges in executing RAG across multiple partitions. We show that the training objective of M-RAG is well aligned with that of text generation tasks. (3) We conduct extensive experiments on *seven* datasets for *three* generation tasks on *three* distinct language model architectures, including a recent Mixture of Experts (MoE) architecture (Jiang et al., 2024). The results demonstrate the effectiveness of M-RAG across diverse RAG baselines. In comparison to the best baseline approach, M-RAG exhibits improvements of 11%, 8%, and 12% for text summarization, machine translation, and dialogue generation tasks, respectively.

2 Related Work

Retrieval-Augmented Generation. We review the literature of Retrieval-Augmented Generation (RAG) in terms of (1) Naive RAG, (2) Advanced RAG, and (3) Modular RAG. For (1), Naive RAG follows a standard process including indexing, retrieval, and generation (Ma et al., 2023). However, its quality faces significant challenges such as low precision, hallucination, and redundancy during the process. For (2), Advanced RAG is further developed to overcome the shortcomings of Naive RAG. Specifically, during the indexing stage, the objective is to enhance the quality of the indexed content by optimizing data embedding (Li et al.,



(a) Summ. (ROUGE-1) (b) Summ. (Runtime) (c) Dịch máy (d) Tạo hội thoại Hình 1: So sánh với các chiến lư ợc phân vùng cơ sở dữ liệu cho các tác vụ tạo ngôn ngữ.

bao gồm các vấn đề cố hữu như ảo giác hoặc bối cảnh không liên quan, có thể ảnh hư ởng đến việc hình thành nền tảng cho thế hệ LLM.

Dựa trên thảo luận đã đề cập ở trên, chúng tôi giới thiệu một giải pháp mới có tên là M-RAG, đư ợc thiết kế để tạo điều kiện thuận lợi cho RAG trên nhiều phân vùng của cơ sở dữ liệu. M-RAG giải quyết tất cả ba thách thức. Đối với (1), chúng tôi rút ra những hiểu biết sâu sắc từ các tài liệu về quản lý cơ sở dữ liệu vectơ (Pan và cộng sự, 2023; Han và cộng sự, 2023) và đánh giá các chiến lư ợc khác nhau, cụ thể là Ngẫu nhiên hóa (Indyk và Mot-wani, 1998), Phân cụm (Jegou và cộng sự, 2010), Lập chỉ mục (Malkov và cộng sự, 2014; Malkov và Yashunin, 2018) và Thể loại (Gollapudi và cộng sự, 2023), thông qua các nghiên cứu thực nghiệm. Hiệu quả của các chiến lư ợc này, cùng với số lư ợng phân vùng tư ơ ng ứng, đư ợc đánh giá trên các tác vụ tạo khác nhau trên một tập phát triển trong các thí nghiệm của chúng tôi.

Đối với (2), với nhiều phân vùng đang hoạt động, chúng tôi xây dựng việc lựa chọn phân vùng như một bài toán máy đánh bạc nhiều tay (Slivkins và cộng sự, 2019). Trong bối cảnh này, một tác nhân, đư ợc ký hiệu là Agent-S, lặp đi lặp lại chọn một trong số nhiều phân vùng. Các đặc điểm của mỗi phân vùng chỉ đư ợc biết một phần tại thời điểm lựa chọn và Agent-S hiểu rõ hơ n theo thời gian bằng cách tối đa hóa phần thư ởng tích lũy trong môi trư ờng. Để tối ư u hóa chính sách quyết định, chúng tôi tận dụng việc học tăng cư ờng với Quy trình quyết định Markov (MDP) đư ợc thiết kế cẩn thận. Đối với (3), sau khi chọn một phân vùng và thu thập bộ nhớ để tạo, chúng tôi giới thiệu một tác nhân khác, đư ợc ký hiệu là Agent-R. Tác nhân này tạo ra một nhóm bộ nhớ ứng viên theo cách lặp đi lặp lại thông qua việc sử dụng LLM. Sau khi chọn đư ợc ứng viên, Agent-R sẽ đánh giá chất lư ợng của ứng viên đó bằng cách chứng minh ứng viên đó để đư a ra giả thuyết. Việc xác định một giả thuyết chất lư ợng cao đư ợc xác định bởi một số liệu hiệu suất cụ thể, kích hoạt một quá trình tăng cư ờng, trong đó nó báo hiệu việc khám phá và thay thế bộ nhớ trư ớc đó bằng một bộ nhớ tốt hơ n và tiếp tục quá trình. Hơ n nữa, chúng tôi tích hợp các nỗ lực của Agent-S và Agent-R thông qua việc học tăng cư ờng đa tác nhân. Với mục tiêu chung là tăng cư ờng việc tạo văn bản

đối với một truy vấn đầu vào nhất định, chúng đư ợc tối ư u hóa chung thông qua quá trình đào tạo đầu cuối.

Những đóng góp của chúng tôi có thể đư ợc tóm tắt như sau: (1) chúng tôi đề xuất một mô hình phân vùng nhiều cho RAG, nhằm tạo điều kiện thuận lợi cho việc truy xuất chi tiết và tập trung vào các bộ nhớ quan trọng để nâng cao hiệu suất tổng thể. Ngoài ra, việc sử dụng nhiều phân vùng có lợi cho các khía cạnh khác của RAG, bao gồm tạo điều kiện thuận lợi cho việc xây dựng và duy trì các chỉ mục, bảo vệ dữ liệu riêng tư của ngư ời dùng trong các phân vùng cụ thể và hỗ trợ xử lý song song phân tán trên các phân vùng khác nhau. (2) Chúng tôi giới thiệu M-RAG, một giải pháp mới dựa trên học tăng cư ờng đa tác nhân giải quyết ba thách thức trong việc thực thi RAG trên nhiều phân vùng . Chúng tôi chỉ ra rằng mục tiêu đào tạo của M-RAG phù hợp với mục tiêu của các tác vụ tạo văn bản.

(3) Chúng tôi tiến hành các thí nghiệm mở rộng trên bảy tập dữ liệu cho ba tác vụ tạo trên ba kiến trúc mô hình ngôn ngữ riêng biệt, bao gồm kiến trúc Hỗn hợp chuyên gia (MoE) gần đây (Jiang và cộng sự, 2024). Các kết quả chứng minh hiệu quả của M-RAG trên nhiều đư ờng cơ sở RAG khác nhau. So với phư ơ ng pháp đư ờng cơ sở tốt nhất, M-RAG thể hiện sự cải thiện lần lư ợt là 11%, 8% và 12% cho các tác vụ tóm tắt văn bản, dịch máy và tạo hội thoại.

2 Công trình liên quan

Retrieval-Augmented Generation. Chúng tôi xem xét tài liệu về Retrieval-Augmented Generation (RAG) theo các thuật ngữ (1) Naive RAG, (2) Advanced RAG và (3) Modular RAG. Đối với (1), Naive RAG tuân theo một quy trình chuẩn bao gồm lập chỉ mục, truy xuất và tạo (Ma và cộng sự, 2023). Tuy nhiên, chất lư ợng của nó phải đối mặt với những thách thức đáng kể như độ chính xác thấp, ảo giác và trùng lặp trong quá trình này. Đối với (2), Advanced RAG đư ợc phát triển thêm để khắc phục những thiếu sót của Naive RAG. Cụ thể, trong giai đoạn lập chỉ mục, mục tiêu là nâng cao chất lư ợng nội dung đư ợc lập chỉ mục bằng cách tối ư u hóa nhúng dữ liệu (Li và cộng sự,

2023). During the retrieval stage, the focus is on identifying the appropriate context by calculating the similarity between the query and chunks, where the techniques involve fine-tuning embedding models (Xiao et al., 2023), or learning dynamic embeddings for different context (Karpukhin et al., 2020). During the generation stage, it merges the retrieved context with the query as an input into large language models (LLMs), where it addresses challenges posed by context window limits with re-ranking the most relevant content (Jiang et al., 2023b; Zhuang et al., 2023), or compressing prompts (Litman et al., 2020; Xu et al., 2023). In addition, Self-RAG (Asai et al., 2023) is proposed to identify whether retrieval is necessary, or the retrieved context is relevant, which helps language models to produce meaningful generation (Asai et al., 2023). For (3), Modular RAG diverges from the traditional Naive RAG structure by incorporating external modules to further enhance the performance, including search module (Wang et al., 2023a), memory module (Wang et al., 2022; Cheng et al., 2023b), tuning module (Lin et al., 2023), and task adapter (Cheng et al., 2023a; Dai et al., 2023). Specifically, Selfmem (Cheng et al., 2023b) incorporates a retrieval-enhanced generator to iteratively create a memory pool, it then trains a selector to choose one of the memories from the pool to generate responses. The work (Gao et al., 2023) provides a comprehensive survey of RAG for LLMs. Our work differs from existing RAG studies in two aspects. First, we introduce a multiple partition setting, where each partition serves as a fundamental entity for retrieval, rather than retrieving from the entire database. Second, we introduce an M-RAG framework built upon multi-agent reinforcement learning, which tackles three distinct challenges posed by this novel setting.

Reinforcement Learning for LLMs. Recently, reinforcement learning has seen broad applications across a variety of language-related tasks for Large Language Models (LLMs). This includes tasks such as text summarization (Wu et al., 2021a), machine translation (Kreutzer et al., 2018), dialogue systems (Jaques et al., 2019; Yi et al., 2019), semantic parsing (Lawrence and Riezler, 2018), and review generation (Cho et al., 2018). For example, WebGPT (Nakano et al., 2021) incorporates a reinforcement learning framework to autonomously train the GPT-3 model using a search engine during the text generation process. Further, InstructGPT (Ouyang et al., 2022) collects a dataset containing desired model outputs provided by human labelers. Subsequently, it employs Reinforcement Learning from Human Feedback (RLHF) to fine-tune GPT-3 (Brown et al., 2020). In addition, R3 (Ma et al., 2023) introduces a Rewrite-Retrieve-Read process, where the LLM performance serves as a reinforcement learning incentive for a rewriting module. This approach empowers the rewriter to enhance retrieval queries, consequently improving the reader's performance in downstream tasks. MMQS (Wang et al., 2024) introduces a new multi-modal question suggestion task with a multi-agent version of RLHF. In this work, we propose a novel multi-agent reinforcement learning framework utilizing two agents to collaboratively optimize text generation tasks. To our best knowledge, this is the first of its kind.

Multi-source Knowledge-grounded Dialogue System (MKDS). We review the literature on MKDS (Wu et al., 2021b, 2022), and highlight differences with our M-RAG regarding (1) datasets, (2) solutions, and (3) tasks. For (1), MKDS uses multi-source heterogeneous data (plain text, tables, knowledge graphs), each contributing uniquely to dialogue generation. M-RAG uses a single-source homogeneous dataset, initially vectorized and indexed for RAG retrieval. We explore partitioning strategies to create multiple homogeneous partitions for effective retrieval. For (2), MKDS employs an encoder-decoder framework with varied attention weights for different knowledge sources, trained with a small dialogue model like MSKE-Dialog (59.14M parameters) (Wu et al., 2021b). M-RAG uses a Retrieval-then-Generation approach with two RL agents (Agent-S and Agent-R) focusing on retrieval and generation, respectively. For (3), M-RAG leverages LLMs for diverse language generation tasks, including text summarization, machine translation, and dialogue generation, unlike MKDS's specific focus on dialogue generation (Wu et al., 2021b, 2022).

3 Methodology

A task involving M-RAG can be formulated below. Given a database $\mathbb{D} = \{(x_i, y_i)\}_{i=1}^{|\mathbb{D}|}$ for a language generation task (e.g., summarization), where each pair (x, y) represents a document and its corresponding summary stored in $\mathbb{D}$. The M-RAG initiates the process by partitioning $\mathbb{D}$ into multiple partitions. This can be achieved through meth-

2023). Trong giai đoạn truy xuất, trọng tâm là
xác định bối cảnh thích hợp bằng cách tính toán
sự giống nhau giữa truy vấn và các khối, trong đó
các kỹ thuật bao gồm tinh chỉnh các mô hình nhúng (Xiao và
cộng sự, 2023) hoặc học các nhúng động cho các bối cảnh
khác nhau (Karpukhin và cộng sự,
2020). Trong giai đoạn thế hệ, nó hợp nhất
bối cảnh đư ợc lấy lại với truy vấn như một đầu vào
vào các mô hình ngôn ngữ lớn (LLM), nơ i nó giải quyết các
thách thức do giới hạn cửa sổ ngữ cảnh đặt ra
với việc xếp hạng lại nội dung có liên quan nhất (Jiang
et al., 2023b; Zhuang et al., 2023), hoặc nén
gợi ý (Litman và cộng sự, 2020; Xu và cộng sự, 2023). TRONG
ngoài ra, Self-RAG (Asai et al., 2023) đư ợc đề xuất
để xác định xem việc truy xuất có cần thiết hay không, hoặc
ngữ cảnh đư ợc lấy lại có liên quan, giúp ích cho ngôn ngữ
các mô hình để tạo ra thế hệ có ý nghĩa (Asai
et al., 2023). Đối với (3), RAG mô-đun phân kỳ từ
cấu trúc Naive RAG truyền thống bằng cách kết hợp các mô-đun
bên ngoài để nâng cao hiệu suất hơ n nữa, bao gồm mô-đun tìm
kiếm (Wang et al.,
2023a), mô-đun bộ nhớ (Wang et al., 2022; Cheng
và cộng sự, 2023b), mô-đun điều chỉnh (Lin và cộng sự, 2023),
và bộ điều hợp nhiệm vụ (Cheng et al., 2023a; Dai et al.,
2023). Cụ thể là Selfmem (Cheng và cộng sự, 2023b)
kết hợp một trình tạo tăng cư ờng truy xuất để tạo ra một
nhóm bộ nhớ, sau đó nó đào tạo một
bộ chọn để chọn một trong những ký ức từ
nhóm để tạo ra phản hồi. Công trình (Gao et al.,
2023) cung cấp một cuộc khảo sát toàn diện về RAG
cho LLM. Công việc của chúng tôi khác với RAG hiện tại
nghiên cứu theo hai khía cạnh. Đầu tiên, chúng tôi giới thiệu
một thiết lập phân vùng nhiều, trong đó mỗi phân vùng phục vụ
như một thực thể cơ bản để truy xuất, thay vì
lấy từ toàn bộ cơ sở dữ liệu. Thứ hai, chúng tôi
giới thiệu một khuôn khổ M-RAG đư ợc xây dựng dựa trên việc
học tăng cư ờng đa tác nhân, giải quyết ba
những thách thức rõ rệt đặt ra bởi bối cảnh mới lạ này.

Học tăng cư ờng cho LLM. Gần đây,
học tăng cư ờng đã thấy ứng dụng rộng rãi
trên nhiều nhiệm vụ liên quan đến ngôn ngữ dành cho Large
Mô hình ngôn ngữ (LLM). Điều này bao gồm các nhiệm vụ
chẳng hạn như tóm tắt văn bản (Wu et al., 2021a),
dịch máy (Kreutzer và cộng sự, 2018), hệ thống đối
thoại (Jaques và cộng sự, 2019; Yi và cộng sự, 2019),
phân tích ngữ nghĩa (Lawrence và Riezler, 2018),
và tạo ra đánh giá (Cho et al., 2018). Ví dụ, WebGPT (Nakano
et al., 2021) kết hợp một khuôn khổ học tăng cư ờng để tự
động đào tạo mô hình GPT-3 bằng cách sử dụng tìm kiếm
động cơ trong quá trình tạo văn bản. Hơ n nữa,
InstructGPT (Ouyang et al., 2022) thu thập một tập dữ liệu
chứa các đầu ra mô hình mong muốn do ngư ời dán nhãn cung cấp.
Sau đó, nó sử dụng Học tăng cư ờng từ phản hồi của con ngư ời
(RLHF) để
tinh chỉnh GPT-3 (Brown và cộng sự, 2020). Ngoài ra,
R3 (Ma et al., 2023) giới thiệu quy trình Viết lại-Lấy lại
-Đọc, trong đó hiệu suất LLM phục vụ
như một động lực học tập tăng cư ờng cho một mô-đun
viết lại. Cách tiếp cận này trao quyền cho ngư ời viết lại
để tăng cư ờng các truy vấn tìm kiếm, do đó cải
thiện hiệu suất của ngư ời đọc trong các tác vụ tiếp theo.
MMQS (Wang et al., 2024) giới thiệu một nhiệm vụ gợi ý câu
hỏi đa phư ơ ng thức mới với nhiều tác nhân
phiên bản RLHF. Trong tác phẩm này, chúng tôi đề xuất một
khuôn khổ học tăng cư ờng đa tác nhân sử dụng hai tác nhân
để tối ư u hóa văn bản một cách hợp tác
nhiệm vụ thế hệ. Theo hiểu biết tốt nhất của chúng tôi, đây là
đầu tiên thuộc loại này.

Đối thoại dựa trên kiến thức đa nguồn
Hệ thống (MKDS). Chúng tôi xem xét tài liệu về
MKDS (Wu et al., 2021b, 2022) và làm nổi bật
sự khác biệt với M-RAG của chúng tôi liên quan đến (1) tập dữ liệu,
(2) giải pháp và (3) nhiệm vụ. Đối với (1), MKDS sử dụng
dữ liệu không đồng nhất nhiều nguồn (văn bản thuần túy, bảng,
đồ thị kiến thức), mỗi đồ thị đều đóng góp một cách độc đáo vào
thế hệ đối thoại. M-RAG sử dụng một nguồn duy nhất
tập dữ liệu đồng nhất, ban đầu đư ợc vector hóa và lập chỉ
mục để truy xuất RAG. Chúng tôi khám phá phân vùng
chiến lư ợc để tạo ra nhiều phân vùng đồng nhất để truy xuất
hiệu quả. Đối với (2), MKDS sử dụng một khuôn khổ mã hóa-
giải mã với nhiều
trọng số chú ý cho các nguồn kiến thức khác nhau,
đư ợc đào tạo bằng mô hình đối thoại nhỏ như MSKE-Dialog
(59,14 triệu tham số) (Wu et al., 2021b).
M-RAG sử dụng phư ơ ng pháp Truy xuất rồi Tạo
với hai tác nhân RL (Tác nhân-S và Tác nhân-R) tập trung vào
việc truy xuất và tạo ra tư ơ ng ứng. Đối với
(3), M-RAG tận dụng LLM cho ngôn ngữ đa dạng
nhiệm vụ tạo ra, bao gồm tóm tắt văn bản, dịch
máy và tạo hội thoại, không giống như
Tập trung cụ thể của MKDS vào việc tạo ra đối thoại (Wu
và cộng sự, 2021b, 2022).

3 Phư ơ ng pháp

Một nhiệm vụ liên quan đến M-RAG có thể đư ợc xây dựng như sau.
Cho cơ sở dữ liệu D = {(xi , yi)} $_{\text{tôi=1}}^{|\mathbb{D}|}$ cho một ngôn ngữ
nhiệm vụ tạo ra (ví dụ, tóm tắt), trong đó mỗi
cặp (x, y) biểu diễn một tài liệu và bản tóm tắt tư ơ ng ứng
của nó đư ợc lư u trữ trong D. M-RAG khởi tạo quy trình bằng
cách phân vùng D thành nhiều
phân vùng. Điều này có thể đạt đư ợc thông qua phư ơ ng pháp

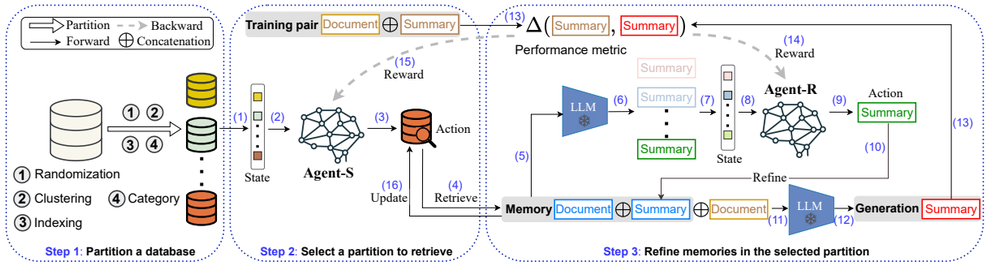


Figure 2: Illustration of M-RAG training in a summarization task: The M-RAG initiates training with multiple partitions (Section 3.1), it then selects a partition to perform retrieval via Agent-S (Section 3.2), and refines the memories within the selected partition via Agent-R (Section 3.3). Both agents are collaboratively trained to enhance generation capabilities through multi-agent reinforcement learning (Section 3.4). For inference, it includes elements (1), (2), (3), (4), (11), and (12).

ods like clustering or by leveraging inherent category labels in the data. The resulting partitions are denoted as $\mathbb{D} = \{D_m\}_{m=1}^{|M|}$, where each D_m $(1 \leq m \leq M)$ supports an independent RAG process (Section 3.1). The M-RAG framework comprises both training and inference processes, as outlined in Algorithm 1. For training, Agent-S learns to select a specific D_m for an input text pair (Section 3.2). Subsequently, Agent-R refines the retrieved memories, represented as $(\tilde{x}, \tilde{y}) \in D_m$, within the selected partition D_m (Section 3.3). Finally, the two agents are collaboratively trained with multi-agent reinforcement learning (see Section 3.4). Figure 2 illustrates the training process of M-RAG. For inference, the refined $\mathbb{D}$ is utilized to support a LLM in generating hypotheses, where a D_m is selected by the trained Agent-S.

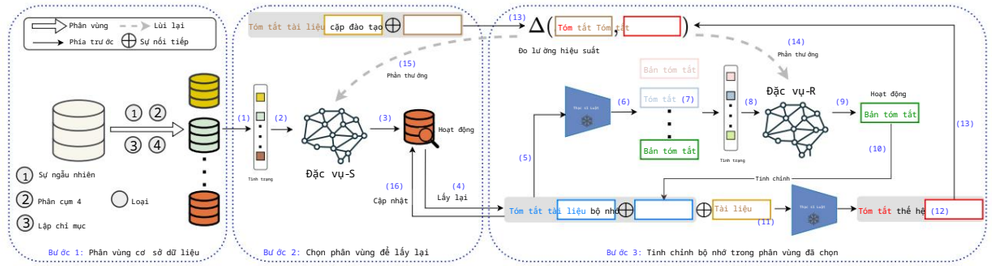
3.1 Discussion on Partitioning a Database

As M-RAG relies on multiple partitions for RAG operations, we investigate various strategies to partition an external database (typically the training corpus). The results of these strategies are then validated through empirical studies. We review the literature, including recent vector database surveys (Pan et al., 2023; Han et al., 2023), and identify the following strategies: namely (1) Randomization (Indyk and Motwani, 1998), (2) Clustering (Jegou et al., 2010), (3) Indexing (Malkov et al., 2014; Malkov and Yashunin, 2018) and (4) Category (Gollapudi et al., 2023). Specifically, for (1), it targets the utilization of probability amplification techniques, such as locality-sensitive hashing (LSH), to hash similar items (data vectors) into the same bucket with a high probability. For (2), it involves clustering data vectors using K-means, where this clustering concept is widely applied in Inverted File Index (IVF) for tasks like Approximate k-Nearest Neighbor (AKNN) search. For (3), navigable graph indexes, such as HNSW (Malkov and Yashunin, 2018) or NSW (Malkov et al., 2014), are designed to facilitate easy traversal of different regions within a vector database. To achieve effective partitions, we employ graph partitioning with spectral clustering on a navigable graph. For (4), it involves assigning data vectors to partitions based on their respective categories. For example, in the DailyDialog dataset (Li et al., 2017), which includes 7 emotion categories (e.g., joy, anger) and 10 topic categories (e.g., work, health), vectors are partitioned according to their category labels. We note that a single vector may be assigned to multiple partitions, due to the characteristics of the dataset, where a dialogue spans multiple categories.

In Figure 1, we perform experiments on a development set, manipulating the number of partitions wrt the 4 strategies across three language generation tasks (summarization, translation, and dialogue generation). The results demonstrate the effectiveness of the strategies, and we conclude the selected strategies with the number of partitions as follows. We choose Indexing (4 partitions), Randomization (3 partitions), and Category (10 partitions) for the summarization, translation, and dialogue generation tasks, respectively. In addition, as shown in Figure 1 (a) and (b), we observe that both Top-1 and Top-3 retrieval methods exhibit comparable performance. For enhanced efficiency, we default to Top-1 retrieval in the rest of the paper.

3.2 Agent-S: Selecting a Database Partition

During the training process of an Agent-S to select a partition from $\mathbb{D}$, the environment is naturally modeled as a bandit setting. In this context, when



Hình 2: Minh họa về đào tạo M-RAG trong tác vụ tóm tắt: M-RAG bắt đầu đào tạo với nhiều phân vùng (Phần 3.1), sau đó chọn một phân vùng để thực hiện truy xuất thông qua Agent-S (Phần 3.2) và tinh chỉnh bộ nhớ trong phân vùng đã chọn thông qua Agent-R (Phần 3.3). Cả hai tác nhân đều đư ợc đào tạo cộng tác để tăng cư ờng khả năng tạo thông qua học tăng cư ờng đa tác nhân (Phần 3.4). Đối với suy luận, nó bao gồm các yếu tố (1), (2), (3), (4), (11) và (12).

ods như phân cụm hoặc bằng cách tận dụng các nhãn danh mục vốn có trong dữ liệu. Các phân vùng kết quả đư ợc ký hiệu là D = {Dm} (1 ≤ m ≤ M) hỗ trợ quy trình RAG độc lập (Mục 3.1). Khung M-RAG bao gồm cả quy trình đào tạo và suy luận, như đư ợc nêu trong Thuật toán 1. Đối với đào tạo, Agent-S học cách chọn một Dm cụ thể cho một cặp văn bản đầu vào (Mục 3.2). Sau đó, Agent-R tinh chỉnh các ký ức đã truy xuất, đư ợc biểu diễn là (˜x, y˜) Dm, trong phân vùng đã chọn Dm (Mục 3.3). Cuối cùng, hai tác nhân đư ợc đào tạo cộng tác với học tăng cư ờng đa tác nhân (xem Mục 3.4). Hình 2 minh họa quy trình đào tạo của M-RAG. Đối với suy luận, D tinh chỉnh đư ợc sử dụng để hỗ trợ LLM trong việc tạo ra các giả thuyết, trong đó Dm đư ợc Agent-S đã đào tạo chọn.

3.1 Thảo luận về Phân vùng Cơ sở dữ liệu

Vì M-RAG dựa vào nhiều phân vùng cho các hoạt động RAG, chúng tôi nghiên cứu nhiều chiến lư ợc khác nhau để phân vùng cơ sở dữ liệu bên ngoài (thư ờng là ngữ liệu đào tạo). Sau đó, kết quả của các chiến lư ợc này đư ợc xác thực thông qua các nghiên cứu thực nghiệm. Chúng tôi xem xét tài liệu, bao gồm các khảo sát cơ sở dữ liệu vectơ gần đây (Pan và cộng sự, 2023; Han và cộng sự, 2023) và xác định các chiến lư ợc sau: cụ thể là (1) Ngẫu nhiên hóa (Indyk và Motwani, 1998), (2) Phân cụm (Jegou và cộng sự, 2010), (3) Lập chỉ mục (Malkov và cộng sự, 2014; Malkov và Yashunin, 2018) và (4) Thể loại (Gollapudi và cộng sự, 2023). Cụ thể, đối với (1), nó nhắm mục tiêu vào việc sử dụng các kỹ thuật khuếch đại xác suất, chẳng hạn như băm nhạy cảm với vị trí (LSH), để băm các mục tư ơng tự (vectơ dữ liệu) vào cùng một nhóm với xác suất cao. Đối với (2), nó liên quan đến việc nhóm các vectơ dữ liệu bằng cách sử dụng K-means, trong đó khái niệm nhóm này đư ợc áp dụng rộng rãi trong

Chỉ mục tệp đảo ngư ợc (IVF) cho các tác vụ như tìm kiếm k-Nearest Neighbor (AKNN) xấp xỉ. Đối với (3), các chỉ mục đồ thị có thể điều hư ớng, chẳng hạn như HNSW (Malkov và Yashunin, 2018) hoặc NSW (Malkov và cộng sự, 2014), đư ợc thiết kế để tạo điều kiện dễ dàng duyệt qua các vùng khác nhau trong cơ sở dữ liệu vectơ . Để đạt đư ợc các phân vùng hiệu quả, chúng tôi sử dụng phân vùng đồ thị với cụm phổ trên đồ thị có thể điều hư ớng. Đối với (4), nó liên quan đến việc gán các vectơ dữ liệu cho các phân vùng dựa trên các danh mục tư ơ ng ứng của chúng. Ví dụ, trong tập dữ liệu DailyDialog (Li và cộng sự, 2017), bao gồm 7 danh mục cảm xúc (ví dụ: vui vẻ, tức giận) và 10 danh mục chủ đề (ví dụ: công việc, sức khỏe), các vectơ đư ợc phân vùng theo nhãn danh mục Chúng tôi lư u ý rằng một vectơ duy nhất có thể đư ợc gán cho nhiều phân vùng, do đặc điểm của tập dữ liệu, trong đó hộp thoại trải dài trên nhiều danh mục.

Trong Hình 1, chúng tôi thực hiện các thí nghiệm trên một tập phát triển, thao tác số lư ợng phân vùng liên quan đến 4 chiến lư ợc trên ba tác vụ tạo ngôn ngữ (tóm tắt, dịch và tạo đối thoại). Kết quả chứng minh tính hiệu quả của các chiến lư ợc và chúng tôi kết luận các chiến lư ợc đã chọn với số lư ợng phân vùng như sau. Chúng tôi chọn Lập chỉ mục (4 phân vùng), Ngẫu nhiên hóa (3 phân vùng) và Thể loại (10 phân vùng) cho các tác vụ tóm tắt, dịch và tạo đối thoại. Ngoài ra , như thể hiện trong Hình 1 (a) và (b), chúng tôi quan sát thấy cả phư ơ ng pháp truy xuất Top-1 và Top-3 đều thể hiện hiệu suất tư ơ ng đư ơ ng. Để nâng cao hiệu quả, chúng tôi mặc định sử dụng truy xuất Top-1 trong phần còn lại của bài báo.

3.2 Agent-S: Chọn phân vùng cơ sở dữ liệu Trong quá

trình đào tạo của Agent-S để chọn phân vùng từ D, môi trư ờng đư ợc mô hình hóa tự nhiên như một thiết lập bandit. Trong bối cảnh này, khi

a random partition is selected, the language model generates a response for the query with feedback (typically based on a specific performance metric), and concludes the episode. The selection process can be formulated as a Markov Decision Process (MDP), involving states, actions, and rewards.

States. Given a training pair (x, y) and a set of database partitions $\mathbb{D} = \{D_m\}_{m=1}^{|M|}$, the state $s^{(S)}$ is defined by assessing the semantic relevance, typically quantified by measures such as cosine similarity $\text{sim}(\cdot, \cdot)$, between the input (x, y) and the stored memories $(\tilde{x}, \tilde{y})$ within each D_m.

$$s^{(S)} = \{\max_{(\tilde{x},\tilde{y})\in D_m} \text{sim}(\sigma(\tilde{x} \oplus \tilde{y}), \sigma(x \oplus y))\}_{m=1}^{|M|}, \tag{1}$$

where $\oplus$ denotes the concatenation operation, and $\sigma(\cdot)$ denotes an embedded model utilized to obtain text representations, such as the CPT-Text (Neelakantan et al., 2022). We consider the Top-1 retrieved memories to construct the state.

Actions. Let $a^{(S)}$ represent an action undertaken by Agent-S. The design of actions corresponds to that of the state $s^{(S)}$. Specifically, the actions are defined as follows:

$$a^{(S)} = m \ (1 \le m \le M), \tag{2}$$

where action $a^{(S)} = m$ means to select the D_m for subsequent the generation task.

Rewards. The reward is denoted by $r^{(S)}$. When the action $a^{(S)}$ involves exploring a partition, the reward cannot be immediately observed, as no response has been received for the query x. However, when the action involves selecting a partition for Agent-R to refine the memories within the partition, the stored response $\tilde{y}$ is updated, and some reward signal can be obtained (for example, by measuring the difference between the results on the original memory and that on the refined memory). Therefore, we make Agent-S and Agent-R are trained with multi-agent reinforcement learning, since they cooperate towards the same objective of learning a policy that produces a response (hypothesis) as similar as possible to the reference y for the x.

3.3 Agent-R: Refining Memories in the Selected Partition

Next, we formulate the task of refining the retrieved memories carried out by Agent-R within a selected partition. To accomplish this, Agent-R explores potential responses denoted by $\hat{y}$ through LLMs for the retrieved $\tilde{x}$, and generates a candidate pool $\mathbb{C} = \{\hat{y}_k \leftarrow \text{LLM}(\tilde{x})\}_{k=1}^{|K|}$ for selection, where K denotes the number of candidates. Upon selecting a candidate, Agent-R evaluates its quality by demonstrating the new memory $(\tilde{x}, \hat{y}_k)$ to generate a hypothesis $h \leftarrow \text{LLM}(x \oplus (\tilde{x}, \hat{y}_k))$. In summary, a high-quality hypothesis h benefits from superior memory, which can be then refined through the produced hypothesis for subsequent selections. Consequently, Agent-R iterates in a boosting process optimized via reinforcement learning, where the states, actions, and rewards are detailed below.

States. The state $s^{(R)}$ is defined to assess the semantic relevance between the produced hypothesis h and the selected $\hat{y}_k$ from the pool $\mathbb{C}$. The rationale is to identify a memory that closely resembles the hypothesis, which aligns with the human intuition that a superior demonstration sample often leads to better generation results, that is

$$s^{(R)} = \{\text{sim}(\sigma(h), \sigma(\hat{y}_k))\}_{k=1}^{|K|}, \tag{3}$$

where $\sigma(\cdot)$ denotes an embedded model, and K governs the constructed state space.

Actions. Let $a^{(R)}$ represent an action taken by Agent-R. The design is consistent with the state $s^{(R)}$, which involves selecting a candidate memory from the pool, that is

$$a^{(R)} = k \ (1 \le k \le K). \tag{4}$$

Rewards. We denote the reward of Agent-R as $r_t^{(R)}$, which corresponds to the transition from the current state $\mathbf{s}_t^{(R)}$ to the next state $\mathbf{s}_{t+1}^{(R)}$ after taking action $a_t^{(R)}$. Specifically, when a memory $(\tilde{x}, \hat{y}_k)$ is updated, the hypothesis changes from h to h' accordingly. We remark that the best hypothesis (denoted as h') identified at state $s^{(R)}$ is maintained according to a specific metric $\Delta(\cdot, \cdot)$ (e.g., ROUGE for text summarization, BLEU for machine translation, BLEU and Distinct for dialogue generation), and the reward is defined as:

$$r^{(R)} = \Delta(h', y) - \Delta(h, y), \tag{5}$$

where y denotes the reference result. In this reward definition, we observe that the objective of the Markov Decision Process (MDP), which aims to maximize cumulative rewards, aligns with Agent-R's goal of discovering the best hypothesis among the memories. To illustrate, we consider the process through a sequence of states:

phân vùng ngẫu nhiên đư ợc chọn, mô hình ngôn ngữ tạo ra phản hồi cho truy vấn với phản hồi (thư ờng dựa trên số liệu hiệu suất cụ thể) và kết thúc tập phim. Quá trình lựa chọn có thể đư ợc xây dựng như một Quy trình quyết định Markov (MDP), bao gồm các trạng thái, hành động và phần thư ởng.

Các trạng thái. Cho một cặp đào tạo (x, y) và một tập hợp (S) trạng phân vùng cơ sở dữ liệu D = {Dm} đư ợc xác định bằng cách đánh giá mức độ liên quan về mặt ngữ nghĩa, thư ờng đư ợc định lư ợng bằng các biện pháp như độ tư ơ ng tự cosin sim(·, ·), giữa đầu vào (x, y) và bộ nhớ đư ợc lư u trữ (˜x, y˜) trong mỗi Dm.

$$S_{(S)} = \{\max_{(\tilde{x},\tilde{y})\ Dm} \text{sim}(\sigma(\tilde{x} \quad \tilde{y}), \sigma(x \quad y))\}_{m=1}^{|M|}, \tag{1}$$

trong đó biểu thị hoạt động nối, và σ(·) biểu thị mô hình nhúng đư ợc sử dụng để thu đư ợc biểu diễn văn bản, chẳng hạn như CPT-Text (Nee-lakantan và cộng sự, 2022). Chúng tôi xem xét các ký ức đư ợc truy xuất Top-1 để xây dựng trạng thái.

Hành động. Hãy để một(S) biểu thị một hành động đư ợc thực hiện bởi Agent-S. Thiết kế của các hành động tư ơ ng ứng với (S) của trạng thái Cụ thể, các hành động là ^{S} · đư ợc định nghĩa như sau:

$$^{Một}(S) = m \ (1 \le m \le M), \tag{2}$$

(S) trong đó hành động a = m có nghĩa là chọn Dm cho tác vụ tạo tiếp theo.

Phần thư ởng. Phần thư ởng đư ợc biểu thị bằng $r^{(S)}_{t}$. Khi hành động a (S) liên quan đến việc khám phá một phân vùng, phần thư ởng không thể đư ợc quan sát ngay lập tức, vì không có phản hồi nào đư ợc nhận cho truy vấn x. Tuy nhiên, khi hành động liên quan đến việc chọn một phân vùng cho Agent-R để tinh chỉnh các bộ nhớ trong phân vùng, phản hồi đư ợc lư u trữ y˜ đư ợc cập nhật và có thể thu đư ợc một số tín hiệu phần thư ởng (ví dụ, bằng cách đo sự khác biệt giữa các kết quả trên bộ nhớ gốc và bộ nhớ đã tinh chỉnh). Do đó, chúng tôi thực hiện Agent-S và Agent-R đư ợc đào tạo bằng học tăng cư ờng đa tác nhân, vì chúng hợp tác hư ởng tới cùng một mục tiêu là học một chính sách tạo ra phản hồi (giả thuyết) giống nhất có thể với tham chiếu y cho x.

3.3 Agent-R: Tinh chỉnh bộ nhớ trong phân vùng đã chọn

Tiếp theo, chúng tôi xây dựng nhiệm vụ tinh chỉnh các ký ức đư ợc thu thập bởi Agent-R trong một phân vùng đã chọn. Để thực hiện điều này, Agent-R khám phá các phản hồi tiềm năng đư ợc biểu thị bằng yˆ thông qua LLM cho x˜ đã lấy đư ợc và tạo ra một nhóm ứng viên $C = \{y\hat{}_k \quad \text{LLM}(\tilde{}x)\}_{k=1}^{|K|}$ để lựa chọn, trong đó K biểu thị số lư ợng ứng viên. Khi chọn một ứng viên, Agent-R đánh giá chất lư ợng của ứng viên đó bằng cách chứng minh bộ nhớ mới (˜x, yˆk) để tạo ra giả thuyết h LLM(x (˜x, yˆk)). Tóm lại, một giả thuyết h chất lư ợng cao đư ợc hư ởng lợi từ bộ nhớ vư ợt trội, sau đó có thể đư ợc tinh chỉnh thông qua giả thuyết đư ợc tạo ra cho các lựa chọn tiếp theo. Do đó, Agent-R lặp lại trong một quy trình tăng cư ờng đư ợc tối ư u hóa thông qua học tăng cư ờng, trong đó các trạng thái, hành động và phần thư ởng đư ợc trình bày chi tiết bên dư ới.

trạng thái. Sự liên quan (R) đư ợc định nghĩa để đánh giá các về mặt tâm lý giữa giả thuyết đư ợc tạo ra h và yˆk đư ợc chọn từ nhóm C. Tỷ lệ là để xác định một bộ nhớ gần giống với giả thuyết, phù hợp với trực giác của con ngư ời rằng một mẫu trình diễn vư ợt trội thư ờng dẫn đến kết quả tạo ra tốt hơ n, tức là

$$S_{(R)} = \{\text{sim}(\sigma(h), \sigma(\hat{}yk))\}_{k=1}^{|K|}, \tag{3}$$

trong đó σ(·) biểu thị một mô hình nhúng và K điều khiển không gian trạng thái đư ợc xây dựng.

Hành động. Hãy để một (R) biểu thị một hành động đư ợc thực hiện bởi Agent-R. Thiết kế phù hợp với trạng thái liên quan đến việc chọn một $S^{(Phải)}$, bộ nhớ ứng viên từ nhóm, nghĩa là

$$^{Một}(R) = k \ (1 \le k \le K). \tag{4}$$

Phần thư ởng. Chúng tôi biểu thị phần thư ởng của Agent-R tư ơ ng ứng $r_t^{(Phải)}$, với sự chuyển đổi từ (R) trạng thái hiện tại $s_{t\ (R)}$ đến trạng thái tiếp theo $s_{t+1}^{(R)}$ sau khi uống . Cụ thể, khi một hành động nhớ (˜x, y˜k) xảy ra tại một thời điểm t, đư ợc cập nhật, giả thuyết thay đổi từ h thành h tư ơ ng ứng. Chúng tôi nhận xét rằng giả thuyết tốt nhất (R) đư ợc duy trì) đư ợc xác (đư ợc biểu thị h' định tại trạng thái s_t là h theo số liệu cụ thể (·, ·) (ví dụ: ROUGE để tóm tắt văn bản, BLEU để dịch máy , BLEU và Distinct để tạo hội thoại) và phần thư ởng đư ợc định nghĩa là:

$$r_{(R)} = (h', y) \quad (h, y), \tag{5}$$

trong đó y biểu thị kết quả tham chiếu. Trong định nghĩa phần thư ởng này, chúng ta thấy rằng mục tiêu của Quy trình quyết định Markov (MDP), nhằm mục đích tối đa hóa phần thư ởng tích lũy, phù hợp với mục tiêu của Agent-R là khám phá ra giả thuyết tốt nhất trong số các ký ức. Để minh họa, chúng ta xem xét quy trình thông qua một chuỗi các trạng thái:

Algorithm 1: The M-RAG Framework

Require : a database $\mathbb{D}$; a frozen LLM$(\cdot)$

1 obtain $\mathbb{D} = \{D_m\}_{m=1}^{|M|}$ via a partitioning strategy
2 initialize Ag-S $\pi_\theta(a^{(S)}|s^{(S)})$, Ag-R $\pi_\phi(a^{(R)}|s^{(R)})$
3 **while** *not converged on a validation set* **do**
4 sample a text pair (x, y) from the training set
5 construct $s_1^{(S)}$ with (x, y) on $\mathbb{D}$ by Eq 1
6 **for** $i = 1, 2, ...$ **do**
7 sample $m = a_i^{(S)} \sim \pi_\theta(a|s_i^{(S)})$
8 $r_i^{(S)} \leftarrow 0$
9 $h \leftarrow \text{LLM}(x \oplus (\tilde{x}, \tilde{y}) \in D_m)$
10 construct $s_1^{(R)}$ with h on $\mathbb{C} = \{\hat{y}_k \leftarrow \text{LLM}(\tilde{x})\}_{k=1}^{|K|}$ by Eq 3
11 **for** $j = 1, 2, ...$ **do**
12 sample $k = a_j^{(R)} \sim \pi_\phi(a|s_j^{(R)})$
13 $h' \leftarrow \text{LLM}(x \oplus (\tilde{x}, \hat{y}_k))$
14 **if** $\Delta(h', y) > \Delta(h, y)$ **then**
15 $r_j^{(R)} \leftarrow \Delta(h', y) - \Delta(h, y)$
16 $D_m.\tilde{y} \leftarrow \hat{y}_k, h \leftarrow h'$
17 **else**
18 $r_j^{(R)} \leftarrow 0$
19 construct $s_{j+1}^{(R)}$ with h on a new $\mathbb{C}$
20 $r_i^{(S)} \leftarrow r_i^{(S)} + r_j^{(R)}$
21 construct $s_{i+1}^{(S)}$ by updating $(\tilde{x}, \tilde{y})$ and (x, y)
22 optimize π_θ and π_ϕ via DQN
23 generate final hypotheses via LLM$(\cdot)$ on $\mathbb{D}$ (where the trained Ag-S selects a partition)

$s_1^{(R)}, s_2^{(R)}, ..., s_N^{(R)}$, concluding at $s_N^{(R)}$. The rewards received at these states, except for the termination state, can be denoted as $r_1^{(R)}, r_2^{(R)}, ..., r_{N-1}^{(R)}$. When future rewards are not discounted, we have:

$$\sum_{t=2}^{N} r_{t-1}^{(R)} = \sum_{t=2}^{N} (\Delta(h_t, y) - \Delta(h_{t-1}, y)) = \Delta(h_N, y) - \Delta(h_1, y), \quad (6)$$

where $\Delta(h_N, y)$ corresponds to the highest hypothesis value found throughout the entire iteration, and $\Delta(h_1, y)$ represents an initial value that remains constant. Therefore, maximizing cumulative rewards is equivalent to maximizing the discovered hypothesis value. Finally, the cumulative reward is shared with Agent-S to align with the training objective, that is

$$r^{(S)} = \Delta(h_N, y) - \Delta(h_1, y). \quad (7)$$

3.4 The M-RAG Framework

Policy Learning via DQN. In a MDP, the primary challenge lies in determining an optimal policy that guides an agent to select actions at states, with the aim of maximizing cumulative rewards. Given that the states within our MDPs are continuous, we employ Deep Q-Networks (DQN) with replay memory (Mnih et al., 2013) to learn the policy, denoted as $\pi_\theta(a^{(S)}|s^{(S)})$ for Agent-S (resp. $\pi_\phi(a^{(R)}|s^{(R)})$ for Agent-R). The policy samples an action $a^{(S)}$ (resp. $a^{(R)}$) at a given state $s^{(S)}$ (resp. $s^{(R)}$) via DQN, with parameters denoted by θ (resp. ϕ).

Combining Agent-S and Agent-R. We present the M-RAG framework in Algorithm 1, which combines the functionalities of Agent-S and Agent-R on multiple partitions (line 1). The algorithm comprises two main phases: training and inference. During the training phase (lines 2-22), we randomly sample text pairs from the training set (line 4). For each pair, we generate episodes to iteratively train Agent-S and Agent-R, with the MDPs outlined in (lines 6-21) and (lines 11-20), respectively. Experiences of $(s_t^{(S)}, a_t^{(S)}, r_t^{(S)}, s_{t+1}^{(S)})$ and $(s_t^{(R)}, a_t^{(R)}, r_t^{(R)}, s_{t+1}^{(R)})$ are stored during the iteration, and a minibatch is sampled to optimize the two agents via DQN (line 22).

During the inference phase (line 23), final hypotheses are generated via a LLM based on the refined $\mathbb{D}$, where a partition is selected by the trained Agent-S, and the $\tilde{y}$ and y (unknown during inference) are omitted to construct the state by Eq 1.

Time Complexity. We discuss the complexity of M-RAG compared to a Naive RAG setup introduced in Section 2 in terms of the three steps: (1) indexing, (2) retrieval, and (3) generation as shown in Figure 2. In terms of inference, involving (1) and (2), it is worth noting that the M-RAG exhibits a complexity comparable to that of a Naive RAG setup, with the additional complexity (3) only being involved during training.

For (1), the complexity associated with constructing multiple partitions (e.g., using the HNSW index structure) is represented as $O(M \cdot N \log N)$, where M indicates the number of partitions and N indicates the maximum number of memories within a partition. This approach proves to be faster compared to a Naive RAG setup, which organizes all data within a single index structure with a construction complexity of $O(N' \log N')$, where N' represents the total number of memories in the database.

For (2), the complexity of Agent-S is approximately $O(M \cdot \log N)$, where an AKNN search is performed within each partition, incurring a cost of $O(M \cdot \log N)$ with HNSW. Additionally, sampling actions via Agent-S requires $O(1)$ complexity, owing to its lightweight neural network architecture.

Thuật toán 1: Khung M-RAG

Yêu cầu: một cơ sở dữ liệu D; một LLM đóng băng(·)

1 thu đ ợc D = {Dm} m=1 |M| thông qua một chiến l ợc phân vùng

(S) 2 khởi tạo Ag-S πθ(a |s (S)), Ag-R πφ(a |s (R) (Phải))

3 trong khi không hội tụ trên một tập hợp xác thực

4 lấy mẫu một cặp văn bản (x, y) từ tập huấn luyện

5 xây dựng s 1 (S) với (x, y) trên D theo phư ơ ng trình 1

6 với i = 1, 2, ...

7 làm (S) mẫu m = a (S) ~ πθ(a|s (S) tôi) tôi

8 0

9 h LLM(x (˜x, y˜) Dm)

10 s 1 |K| C = {y˜k (R) với h trên cấu trúc LLM(˜x)} cho j = 1, 2, ... thực hiện (R) mẫu k = ah k=1 bằng phư ơ ng trình 3

11

12 , j πφ(a|s (R) j)

13 LLM(x, (˜x, y˜k)) nếu (h

14 , y) > (h, y) sau đó

15 r (Phải) j (h , y) , (h, y)

16 Dm.y˜ y˜k, h h

17 khác

18 rj (Phải) 0

19 mới s j+1 (R) với h trên một cấu trúc C

20 (S) tôi r + r ij (S) (R)

21 xây dựng s i+1 (S) bằng cách cập nhật (˜x, y˜) và (x,

22 y) tối ư u hóa πθ và πφ thông qua DQN

23 tạo ra các giả thuyết cuối cùng thông qua LLM(·) trên D (nơ i Ag-S đ ợc đào tạo chọn một phân vùng)

các trạng thái trong MDP của chúng tôi là liên tục, chúng tôi sử dụng Mạng Q sâu (DQN) với bộ nhớ phát lại (Mnih et al., 2013) để tìm hiểu chính sách, đ ợc biểu thị là πθ(a (S) |s (S)) cho Agent-S (tư ơ ng ứng là πφ(a (R) |s (R)) cho Agent-R). Chính sách lấy mẫu một hành động a (R)) ở trạng (S) (tư ơ ng thái s nhất định (S) (tư ơ ng ứng với s (R)) thông qua ứng với a DQN, với các tham số đ ợc biểu thị bằng θ (tư ơ ng ứng với φ). Kết hợp Agent-S và Agent-R. Chúng tôi trình bày khuôn khổ M-RAG trong Thuật toán 1, kết hợp các chức năng của Agent-S và Agent-R trên nhiều phân vùng (dòng 1). Thuật toán bao gồm hai giai đoạn chính: đào tạo và suy luận. Trong giai đoạn đào tạo (dòng 2-22), chúng tôi lấy mẫu ngẫu nhiên các cặp văn bản từ bộ đào tạo (dòng 4). Đối với mỗi cặp, chúng tôi tạo các tập để đào tạo lặp lại Agent-S và Agent-R, với các MDP đ ợc nêu trong (dòng 6-21) và (dòng 11-20), tư ơ ng ứng (S)

(S) tích cực. Kinh nghiệm của một t , t (S) ,r t , (S) t+1) và (Phải) (Phải) (S (Rt (Phải) (s tt+1) ngoài (Phải)) đ ợc lư u trữ trong quá trình lặp lại và một lô nhỏ đ ợc lấy mẫu để tối ư u hóa hai tác nhân thông qua DQN (dòng 22).

Trong giai đoạn suy luận (dòng 23), các giả thuyết cuối cùng đ ợc tạo ra thông qua LLM dựa trên D đã tinh chỉnh , trong đó một phân vùng đ ợc chọn bởi Agent-S đã đ ợc đào tạo và y˜ và y (không xác định trong quá trình suy luận) bị bỏ qua để xây dựng trạng thái theo Phư ơ ng trình 1.

Độ phức tạp về thời gian. Chúng tôi thảo luận về độ phức tạp của M-RAG so với thiết lập Naive RAG đ ợc giới thiệu trong Phần 2 theo ba bư ớc: (1) lập chỉ mục , (2) truy xuất và (3) tạo như thể hiện trong Hình 2. Về mặt suy luận, liên quan đến (1) và (2), cần lư u ý rằng M-RAG thể hiện độ phức tạp tư ơ ng đ ơ ng với thiết lập Naive RAG, với độ phức tạp bổ sung (3) chỉ liên quan đến trong quá trình đào tạo.

Đối với (1), độ phức tạp liên quan đến việc xây dựng nhiều phân vùng (ví dụ, sử dụng cấu trúc chỉ mục HNSW) đ ợc biểu thị là O(M ·N log N), trong đó M biểu thị số lư ợng phân vùng và N biểu thị số lư ợng bộ nhớ tối đa trong một

phân vùng. Phư ơ ng pháp này tỏ ra nhanh hơ n so với thiết lập Naive RAG, thiết lập này sắp xếp tất cả dữ liệu trong một cấu trúc chỉ mục duy nhất với độ phức tạp xây dựng là O(N′ log N′), trong đó N′ biểu thị tổng số bộ nhớ trong cơ sở dữ liệu.

Đối với (2), độ phức tạp của Agent-S xấp xỉ là O(M · log N), trong đó tìm kiếm AKNN đ ợc thực hiện trong mỗi phân vùng, gây ra chi phí là O(M · log N) với HNSW. Ngoài ra, các hành động lấy mẫu thông qua Agent-S yêu cầu độ phức tạp O(1) , do kiến trúc mạng nơ -ron nhẹ của nó.

s s (Phải) 1 , 2 (Phải) , ..., N (Phải) , kết thúc tại s N (Phải) . Ở đó-

các phư ơng đ ợc nhận tại các tiểu bang này, ngoại trừ các phư ơng cuối cùng (R) rr 2 , ...,

quốc gia dân tộc, có thể đ ợc ký hiệu là r 1 (R) , N 1 . (Phải)

Khi phần thư ởng trong tư ơ ng lai không đ ợc chiết khấu, chúng ta có:

$$\sum_{t=2}^{N} r_{t-1}^{(\text{Phải})} = \sum_{t=2}^{N} (\Delta(h_t, y) - \Delta(h_{t-1}, y)) = \Delta(h_N, y) - \Delta(h_1, y), \quad (6)$$

trong đó (hN , y) tư ơ ng ứng với giá trị giả thuyết cao nhất đ ợc tìm thấy trong toàn bộ quá trình lặp lại và (h1, y) biểu thị giá trị ban đầu không đổi. Do đó, việc tối đa hóa phần thư ởng tích lũy tư ơ ng đ ơ ng với việc tối đa hóa giá trị giả thuyết đ ợc phát hiện. Cuối cùng, phần thư ởng tích lũy đ ợc chia sẻ với Agent-S để phù hợp với mục tiêu đào tạo, tức là

$$r^{(S)} = \Delta(h_N, y) - \Delta(h_1, y). \quad (7)$$

3.4 Khung M-RAG

Học chính sách thông qua DQN. Trong MDP, thách thức chính nằm ở việc xác định chính sách tối ư u hư ớng dẫn tác nhân lựa chọn hành động ở các trạng thái, với mục đích tối đa hóa phần thư ởng tích lũy. Với điều kiện

In contrast, for the Naive RAG setup, conducting the AKNN search within the entire database costs $O(\log N')$, which is marginally faster than the M-RAG setup.

For (3), the complexity of Agent-R is roughly $O(C \cdot E^2)$, where E tokens are generated via a LLM based on the transformer attention mechanism, and C represents the number of its MDP iterations. This component predominantly influences the overall training complexity. In contrast, for a Naive RAG setup, it runs only once during the inference procedure to produce the generation outcomes, with a complexity of approximately $O(E^2)$.

4 Experiments

4.1 Experimental Setup

Datasets. By following (Cheng et al., 2023b), we conduct experiments on seven datasets for three generation tasks: (1) text summarization (XSum Narayan et al., 2018 and BigPatent Sharma et al., 2019), (2) machine translation (JRC-Acquis Steinberger et al., 2006 with Es→En, En→Es, De→En, and En→De), and (3) dialogue generation (DailyDialog Li et al., 2017). Specifically, XSum comprises single-document summaries for highly abstractive articles sourced from BBC news. BigPatent comprises 1.3 million records of U.S. patent documents accompanied by human-written abstractive summaries. JRC-Acquis serves as a collection of parallel legislative texts of European Union Law, commonly employed as a benchmark in machine translation tasks. DailyDialog comprises multi-turn dialogues centered around daily life topics. The detailed statistics for these datasets are available in (Cheng et al., 2023b).

Baselines. We carefully review the literature including a recent survey paper (Gao et al., 2023), and identify the following RAGs, namely Naive RAG (Ma et al., 2023), Self-RAG (Asai et al., 2023), and Selfmem (Cheng et al., 2023b), which correspond to three kinds of RAG techniques as described in Section 2. In addition, we incorporate the RAGs into three typical language model architectures, namely Mixtral 8×7B (Jiang et al., 2024), Llama 2 13B (Touvron et al., 2023), Phi-2 2.7B (Abdin et al., 2023), Gemma 7B (Mesnard et al., 2024), and Mistral 7B (Jiang et al., 2023a) for the evaluation.

Evaluation Metrics. We evaluate the effectiveness of M-RAG in terms of the three generation tasks by following (Cheng et al., 2023b). (1) For summarization, ROUGE (R-1/2/L) (Lin, 2004) is used. (2) For machine translation, BLEU (Post, 2018) is used. (3) For dialogue generation, BLEU (B-1/2) and Distinct (D-1/2) (Li et al., 2016, 2021) are used. Overall, a higher evaluation metric (i.e., ROUGE, BLEU, Distinct) indicates a better result. We remark that all results are statistically significant, as confirmed by a t-test with $p < 0.05$.

Implementation Details. We implement M-RAG and adapt other baselines using Python 3.7 and LlamaIndex. The database partitioning strategies for Randomization [1] and Indexing [2] utilize existing libraries. The Agent-S (resp. Agent-R) is instantiated through a two-layered feedforward neural network. The first layer consists of 25 neurons using the tanh activation function, and the second layer comprises M (resp. K) neurons corresponding to the action space with a linear activation function. The hyperparameters M and K are empirically set to 4 and 3, respectively. Some of the built-in RL codes can be found in the GitHub repositories referenced in (Wang et al., 2023b, 2021). During training, we randomly sample 10% of text pairs from the training set, while the remaining data is utilized for constructing the database with multiple partitions. The MDP iterations are determined by performance evaluation on a validation set. Evaluation metrics, such as ROUGE, BLEU, and Distinct, are obtained from (Cheng et al., 2023b). The language models with 4-bit quantization, including Mixtral 8×7B, Llama 2 13B, Phi-2 2.7B, Gemma 7B, and Mistral 7B, are available for download via the link [3]. To boost training efficiency, we cache the QA pairs generated by the LLMs during training.

4.2 Experimental Results

(1) Effectiveness evaluation (partitioning strategies). We conduct experiments to evaluate various partitioning strategies across text summarization (XSum), machine translation (Es→En), and dialogue generation (DailyDialog) tasks with Mixtral 8 × 7B. The best results, based on a development set across different partitions, are reported. As shown in Figure 1, we observe that retrieval based on the entire database generally fails to achieve optimal performance. Moreover, the performance slightly decreases as the number of partitions increases. This is attributed to the AKNN search, where a smaller partition size recalls more similar

[1] https://pypi.org/project/graph-partition/
[2] https://pypi.org/project/LocalitySensitiveHashing/
[3] https://huggingface.co/TheBloke

Ngư ợc lại, đối với thiết lập Naive RAG, việc thực hiện tìm kiếm AKNN trong toàn bộ cơ sở dữ liệu có chi phí là O(log N′), nhanh hơ n một chút so với thiết lập M-RAG.

Đối với (3), độ phức tạp của Agent-R là gần đúng O(C · E2), trong đó E mã thông báo đư ợc tạo ra thông qua LLM dựa trên cơ chế chú ý của máy biến áp nism, và C biểu diễn số lần lặp MDP của nó . Thành phần này chủ yếu ảnh hư ởng đến độ phức tạp của quá trình đào tạo tổng thể. Ngư ợc lại, đối với thiết lập Naive RAG, nó chỉ chạy một lần trong quy trình suy luận để tạo ra kết quả tạo ra, với độ phức tạp xấp xỉ O(E2).

4 Thí nghiệm

4.1 Thiết lập thử nghiệm

Bộ dữ liệu. Bằng cách theo dõi (Cheng et al., 2023b), chúng tôi tiến hành các thí nghiệm trên bảy bộ dữ liệu cho ba tác vụ tạo: (1) tóm tắt văn bản (XSum Narayan et al., 2018 và BigPatent Sharma et al., 2019), (2) dịch máy (JRC-Acquis Steinberger et al., 2006 với Es En, En Es, De En và En De), và (3) tạo hội thoại (DailyDialog Li et al., 2017). Cụ thể, XSum bao gồm các bản tóm tắt tài liệu đơ n lẻ cho các bài báo có tính trừu tư ợng cao có nguồn gốc từ tin tức BBC. BigPatent bao gồm 1,3 triệu bản ghi tài liệu bằng sáng chế của Hoa Kỳ kèm theo các bản tóm tắt trừu tư ợng do con ngư ời viết. JRC-Acquis đóng vai trò là một bộ sư u tập các văn bản lập pháp song song của Luật Liên minh Châu Â u, thư ờng đư ợc sử dụng làm chuẩn mực trong các tác vụ dịch máy. DailyDia -log bao gồm các cuộc đối thoại nhiều lư ợt tập trung vào các chủ đề cuộc sống hàng ngày. Số liệu thống kê chi tiết cho các tập dữ liệu này có sẵn trong (Cheng et al., 2023b).

Đư ờng cơ sở. Chúng tôi xem xét cẩn thận các tài liệu bao gồm một bài báo khảo sát gần đây (Gao et al., 2023) và xác định các RAG sau đây, cụ thể là Naive RAG (Ma et al., 2023), Self-RAG (Asai et al., 2023) và Selfmem (Cheng et al., 2023b), tư ơ ng ứng với ba loại kỹ thuật RAG như đư ợc mô tả trong Phần 2. Ngoài ra, chúng tôi kết hợp các RAG vào ba kiến trúc mô hình ngôn ngữ điển hình, cụ thể là Mixtral 8×7B (Jiang et al., 2024), Llama 2 13B (Touvron et al., 2023), Phi-2 2.7B (Abdin et al., 2023), Gemma 7B (Mesnard et al., 2024) và Mistral 7B (Jiang et al., 2023a) để đánh giá.

Đánh giá số liệu. Chúng tôi đánh giá hiệu quả của M-RAG về mặt ba nhiệm vụ thế hệ bằng cách sau (Cheng et al., 2023b). (1) Tóm lại rization, ROUGE (R-1/2/L) (Lin, 2004) đư ợc sử dụng. (2) Đối với dịch máy, BLEU (Post, 2018) đư ợc sử dụng. (3) Đối với tạo hội thoại, BLEU (B-1/2) và Distinct (D-1/2) (Li et al., 2016, 2021) đư ợc sử dụng. Nhìn chung, một số liệu đánh giá cao hơ n (tức là ROUGE, BLEU, Distinct) chỉ ra kết quả tốt hơ n. Chúng tôi lư u ý rằng tất cả các kết quả đều có ý nghĩa thống kê, như đư ợc xác nhận bởi kiểm định t với p < 0,05.

Chi tiết triển khai. Chúng tôi triển khai M-RAG và điều chỉnh các đư ờng cơ sở khác bằng Python 3.7 và Lla- maIndex. Các chiến lư ợc phân vùng cơ sở dữ liệu cho Randomization và Indexing sử dụng các thư viện hiện có. [1] [2] Agent-S (tư ơ ng ứng với Agent-R) đư ợc khởi tạo thông qua mạng nơ -ron truyền thẳng hai lớp . Lớp đầu tiên bao gồm 25 nơ -ron sử dụng hàm kích hoạt tanh, và lớp thứ hai bao gồm M (tư ơ ng ứng với K) nơ -ron tư ơ ng ứng với không gian hành động với hàm kích hoạt tuyến tính.

Các siêu tham số M và K đư ợc thiết lập theo kinh nghiệm lần lư ợt là 4 và 3. Một số mã RL tích hợp có thể đư ợc tìm thấy trong kho lư u trữ GitHub đư ợc tham chiếu trong (Wang et al., 2023b, 2021). Trong quá trình đào tạo, chúng tôi lấy mẫu ngẫu nhiên 10% cặp văn bản từ tập đào tạo, trong khi dữ liệu còn lại đư ợc sử dụng để xây dựng cơ sở dữ liệu với nhiều phân vùng. Các lần lặp MDP đư ợc xác định bằng cách đánh giá hiệu suất trên một tập xác thực. Các số liệu đánh giá, chẳng hạn như ROUGE, BLEU và Distinct, đư ợc lấy từ (Cheng et al., 2023b). Các mô hình ngôn ngữ với lư ợng tử hóa 4 bit, bao gồm Mixtral 8×7B, Llama 2 13B, Phi-2 2.7B, Gemma 7B và Mistral 7B, có sẵn để tải xuống qua 3 liên kết

· Để tăng hiệu quả đào tạo, chúng tôi lư u trữ đệm các cặp QA do LLM tạo ra trong quá trình đào tạo.

4.2 Kết quả thử nghiệm (1) Đánh giá

hiệu quả (chiến lư ợc phân vùng). Chúng tôi tiến hành các thử nghiệm để đánh giá các chiến lư ợc phân vùng khác nhau trên các tác vụ tóm tắt văn bản (XSum), dịch máy (Es En) và tạo hội thoại (DailyDialog) với Mixtral 8 × 7B. Kết quả tốt nhất, dựa trên một tập phát triển trên các phân vùng khác nhau, đư ợc báo cáo. Như thể hiện trong Hình 1, chúng tôi quan sát thấy rằng việc truy xuất dựa trên toàn bộ cơ sở dữ liệu thư ờng không đạt đư ợc hiệu suất tối ư u. Hơ n nữa, hiệu suất giảm nhẹ khi số lư ợng phân vùng tăng lên. Điều này đư ợc quy cho tìm kiếm AKNN, trong đó kích thư ớc phân vùng nhỏ hơ n sẽ nhớ lại nhiều nội dung tư ơ ng tự hơ n

[1] https://pypi.org/project/graph-partition/
[2] https://pypi.org/project/LocalitySensitiveHashing/
[3] https://huggingface.co/TheBloke

Table 1: Text summarization.

LLM	RAG	XSum			BigPatent		
		R-1	R-2	R-L	R-1	R-2	R-L
Mixtral 8 × 7B	None	25.40	6.39	18.30	47.41	16.63	25.14
Mixtral 8 × 7B	Naive	43.82	22.07	37.44	60.11	38.33	43.44
Mixtral 8 × 7B	Selfmem	44.67	22.38	37.86	64.12	39.21	46.21
Mixtral 8 × 7B	Self-RAG	44.01	22.26	37.51	63.59	38.65	45.25
Mixtral 8 × 7B	M-RAG	**48.13**	**24.66**	**39.43**	**71.34**	**42.24**	**47.22**
Llama 2 13B	M-RAG	37.18	18.02	26.44	60.31	37.33	33.47
Phi-2 2.7B	M-RAG	30.70	11.57	26.20	31.25	14.72	18.98

Table 2: Machine translation.

LLM	RAG	Es→En		En→Es		De→En		En→De	
		Dev	Test	Dev	Test	Dev	Test	Dev	Test
Mixtral 8 × 7B	None	34.34	34.81	32.60	28.32	43.75	44.09	43.78	42.24
Mixtral 8 × 7B	Naive	36.64	36.22	33.18	30.70	47.84	46.77	45.83	44.23
Mixtral 8 × 7B	Selfmem	37.65	37.11	34.12	31.86	48.08	47.31	51.38	49.81
Mixtral 8 × 7B	Self-RAG	37.17	36.82	33.80	31.61	47.99	47.27	50.10	48.75
Mixtral 8 × 7B	M-RAG	**39.11**	**39.98**	**35.18**	**32.70**	**49.16**	**48.15**	**53.76**	**50.75**
Llama 2 13B	M-RAG	30.41	30.03	26.40	22.03	41.10	42.22	45.98	42.58
Phi-2 2.7B	M-RAG	22.83	24.22	17.64	16.60	34.21	34.71	40.01	37.08

Table 3: Dialogue generation.

LLM	RAG	DailyDialog			
		B-1	B-2	D-1	D-2
Mix. 8 × 7B	None	15.52	7.05	61.49	89.51
Mix. 8 × 7B	Naive	37.44	29.16	89.42	92.55
Mix. 8 × 7B	Selfmem	38.16	29.92	89.23	95.23
Mix. 8 × 7B	Self-RAG	37.76	29.79	88.24	95.34
Mix. 8 × 7B	M-RAG	**42.61**	**32.97**	88.82	95.74
Llama 2 13B	M-RAG	31.29	17.63	63.19	88.20
Phi-2 2.7B	M-RAG	7.71	3.93	44.21	82.86
Mix. 8 × 7B	M-RAG(D)	39.14	30.98	**93.14**	**98.34**

memories, which may not align well with the LLM preferences and impede the focus on crucial memories. Additionally, we observe that the RAG with Top-1 retrieval exhibits faster runtime compared to the Top-3 due to a shorter input length for the LLM, while maintaining comparable performance.

(2) Effectiveness evaluation (text summarization). We compare the performance of the M-RAG against alternative RAG methods on three distinct language models: Mixtral 8×7B, Llama 2 13B, and Phi-2 2.7B. The corresponding results are outlined in Table 1. We observe consistent improvement in language models when utilizing the RAG framework (e.g., Naive) compared to models without RAG (e.g., None). In addition, the recent MoE architecture Mistral 8 × 7B generally outperforms the typical Llama 2 13B in the summarization task. Specifically, when considering Mistral 8 × 7B as a base model, the performance of M-RAG outperforms that of other baseline models on both datasets. For example, it achieves better results than the best baseline model Selfmem, by 8% and 11% in terms of R-1 on XSum and BigPatent, respectively.

(3) Effectiveness evaluation (machine translation). We further conduct experiments to evaluate the performance of M-RAG for machine translation, and the results are reported in Table 2. We observe that a consistent improvement in the performance of translation tasks with M-RAG across four datasets and three architectures. Notably, it surpasses the Selfmem by 8% in the Es→En translation task.

(4) Effectiveness evaluation (dialogue generation). As shown in Table 3, M-RAG further enhances the language model performance for dialogue generation tasks. It outperforms the Selfmem by 12% in terms of B-1. Notably, we can also use the Distinct score as the performance metric for optimizing the two agents, denoted by M-RAG(D), and it results in a more diverse dialogue.

(5) Effectiveness evaluation (results on 7B LLMs). We increase the number of evaluated LLMs, e.g., comparing 7B models (Gemma 7B and Mistral 7B) to show more results. This comparison aims to assess the performance of M-RAG across the three generation tasks, against the best baseline method Selfmem. The results are presented in Table 4. In general, M-RAG consistently outperforms Selfmem on the 7B models.

(6) Ablation study. To evaluate the effectiveness of the two agents in M-RAG, we conduct an ablation study on XSum. We remove Agent-S and utilize

Bảng 1: Tóm tắt văn bản.

Thạc sĩ Luật	VẢI	Tổng hợp				Bằng sáng chế lớn	
		R-1	R-2	RL	R-1	R-2	RL
Mixtral 8 × 7B Không có		25,40	6,39	18,30	47,41	16,63	25,14
Mixtral 8 × 7B Ngây thơ		43,82	22,07	37,44	60,11	38,33	43,44
Hỗn hợp 8 × 7B Selfmem		44,67	22,38	37,86	64,12	39,21	46,21
Mixtral 8 × 7B Tự-RAG		44.01	22.26	37.51	63.59	38.65	45.25
Mixtral 8 × 7B M-RAG		48,13	24,66	39,43	71,34	42,24	47,22
Lạc đà không bướu 2 13B	M-RAG	37,18	18,02	26,44	60,31	37,33	33,47
Phi-2 2,7B	M-RAG	30,70	11,57	26,20	31,25	14,72	18,98

Bảng 2: Dịch máy.

Thạc sĩ Luật	VẢI	Es En		En Es		Đê Â n		En De	
		Kiểm thử phát triển		Kiểm thử phát triển		Kiểm thử phát triển		Kiểm thử phát triển	
Mixtral 8 × 7B Không có		34,34	34,81	32,60	28,32	43,75	44,09	43,78	42,24
Hỗn hợp 8 × 7B Ngây thơ		36,64	36,22	33,18	30,70	47,84	46,77	45,83	44,23
Hỗn hợp 8 × 7B Selfmem		37,65	37,11	34,12	31,86	48,08	47,31	51,38	49,81
Mixtral 8 × 7B Tự-RAG		37,17	36,82	33,80	31,61	47,99	47,27	50,10	48,75
Hỗn hợp 8 × 7B M-RAG		39,11	39,98	35,18	32,70	49,16	48,15	53,76	50,75
Lạc đà không bướu 2 13B	M-RAG	30,41	30,03	26,40	22,03	41,10	42,22	45,98	42,58
Phi-2 2,7B	M-RAG	22,83	24,22	17,64	16,60	34,21	34,71	40,01	37,08

Bảng 3: Tạo hội thoại.

Thạc sĩ Luật	VẢI	Đối Thoại Hàng Ngày			
		B-1	B-2	D-1	D-2
Trộn. 8 × 7B Không có		15,52	7,05	61,49	89,51
Hỗn hợp. 8 × 7B Ngây thơ		37,44	29,16	89,42	92,55
Trộn. 8 × 7B Selfmem		38,16	29,92	89,23	95,23
Hỗn hợp. 8 × 7B Self-RAG		37,76	29,79	88,24	95,34
Hỗn hợp. 8 × 7B M-RAG		42,61	32,97	88,82	95,74
Lạc đà không bướu 2 13B M-RAG		31,29	17,63	63,19	88,20
Hỗn hợp	M-RAG	7.71	3.93	44.21	82.86
Phi-2 2.7B. 8 × 7B M-RAG(D)		39,14	30,98	93,14	98,34

những ký ức có thể không phù hợp với LLM sở thích và cản trở sự tập trung vào những ký ức quan trọng . Ngoài ra, chúng tôi quan sát thấy rằng RAG với Truy xuất Top-1 cho thấy thời gian chạy nhanh hơn so với vào Top-3 do độ dài đầu vào ngắn hơn cho LLM, trong khi vẫn duy trì hiệu suất tương đương. (2) Đánh giá hiệu quả (tóm tắt văn bản). Chúng tôi so sánh hiệu suất của M-RAG chống lại các phương pháp RAG thay thế trên ba phương pháp riêng biệt mô hình ngôn ngữ: Mixtral 8×7B, Llama 2 13B và Phi-2 2.7B. Các kết quả tương ứng được nêu ra trong Bảng 1. Chúng tôi quan sát thấy sự cải thiện nhất quán trong các mô hình ngôn ngữ khi sử dụng khuôn khổ RAG (ví dụ: Naive) so với các mô hình không có RAG (ví dụ: Không có). Ngoài ra, Bộ Giáo dục gần đây kiến trúc Mistral 8 × 7B nói chung là vượt trội Llama 2 13B điển hình trong nhiệm vụ tóm tắt. Cụ thể, khi xem xét Mistral 8 × 7B như một mô hình cơ sở, hiệu suất của M-RAG vượt trội của các mô hình cơ sở khác trên cả hai tập dữ liệu. Đối với Ví dụ, nó đạt được kết quả tốt hơn so với tốt nhất mô hình cơ sở Selfmem, tăng 8% và 11% về mặt của R-1 trên XSum và BigPatent. (3) Đánh giá hiệu quả (dịch máy). Chúng tôi tiếp tục tiến hành các thí nghiệm để đánh giá hiệu suất của M-RAG cho dịch máy, và kết quả được báo cáo trong Bảng 2. Chúng tôi quan sát rằng một sự cải thiện nhất quán trong hiệu suất của các nhiệm vụ dịch thuật với M-RAG trên bốn tập dữ liệu và ba kiến trúc. Đáng chú ý là nó vượt qua Tự nhớ 8% trong nhiệm vụ dịch Es En. (4) Đánh giá hiệu quả (tạo đối thoại). Như thể hiện trong Bảng 3, M-RAG tăng cường hơn nữa hiệu suất mô hình ngôn ngữ cho các tác vụ tạo đối thoại. Nó vượt trội hơn Selfmem 12% về mặt B-1. Đáng chú ý, chúng ta cũng có thể sử dụng điểm số Di-tinct làm thước đo hiệu suất để tối ưu hóa hai tác nhân, được biểu thị bằng M-RAG(D), và nó dẫn đến trong một cuộc đối thoại đa dạng hơn. (5) Đánh giá hiệu quả (kết quả trên 7B LLM). Chúng tôi tăng số lượng được đánh giá LLM, ví dụ, so sánh các mô hình 7B (Gemma 7B và Mistral 7B) để hiển thị thêm kết quả. So sánh này nhằm mục đích đánh giá hiệu suất của M-RAG trên ba nhiệm vụ thế hệ, so với đường cơ sở tốt nhất phương pháp Selfmem. Các kết quả được trình bày trong Bảng 4. Nhìn chung, M-RAG luôn vượt trội hơn Selfmem trên các mô hình 7B. (6) Nghiên cứu cắt bỏ. Để đánh giá hiệu quả của hai tác nhân trong M-RAG, chúng tôi tiến hành cắt bỏ nghiên cứu về XSum. Chúng tôi loại bỏ Agent-S và sử dụng

Table 4: Comparing M-RAG on various 7B LLMs.

LLM	RAG	Summarization			Translation (Es→En)	Dialogue	
		R-1	R-2	R-L	BLEU	B-1	B-2
Gemma 7B	Selfmem	31.38	9.97	25.07	24.61	15.56	7.91
Gemma 7B	M-RAG	**33.81**	**12.93**	**27.82**	**26.92**	**18.15**	**9.95**
Mistral 7B	Selfmem	35.40	12.68	27.06	26.26	18.28	10.05
Mistral 7B	M-RAG	**37.47**	**13.24**	**30.49**	**32.65**	**24.52**	**11.53**

Table 5: Ablation study.

Components	R-1	R-2	R-L
M-RAG	**48.13**	**24.66**	**39.43**
w/o Agent-S (single DB)	44.20	22.72	37.40
w/o Agent-R (greedy)	45.75	23.21	38.28
w/o Agent-S and Agent-R	43.82	22.07	37.44

Table 6: Impacts of the number of M in Agent-S.

M	1	2	3	4	5
R-1	44.20	44.53	46.27	48.13	47.21
Index constr. (s)	299	278	257	246	227
Retrieval (s)	0.61	1.09	1.54	2.19	2.59
Generation (s)	83.59	84.88	82.81	82.89	86.64

the entire database for RAG; we replace Agent-R with a greedy rule to select a candidate memory from the pool according to Equation 3; and we remove both agents, which degrades to the Naive RAG. The results are presented in Table 5, demonstrating that both agents contribute to performance improvement. Specifically, removing Agent-S results in a significant decline in R-1 from 48.13 to 44.20. This underscores the role of the multiple partition setting in enhancing overall performance. Moreover, removing Agent-R leads to a reduction in R-1 from 48.13 to 45.75. This decline is attributed to the effectiveness of Agent-R in learning memory selection dynamically, as opposed to relying on a fixed rule for decision-making.

(7) Parameter study (Agent-S state space M). We study the effect of parameter M, which controls the state space of Agent-S and corresponds to the number of partitions. In Table 6, we observe that setting $M = 4$ yields the best effectiveness while maintaining reasonable runtime in terms of index construction, retrieval, and generation. This is consistent with empirical studies illustrated in Figure 1 (a). When $M = 1$, it reduces to a single database for RAG. As M increases, index construction accelerates on smaller partitions, while retrieval time sightly increases due to the additional time required for constructing states by querying each partition. As expected, the retrieval time is much smaller than the language generation time.

(8) Parameter study (Agent-R state space K).

Table 7: Impacts of the number of K in Agent-R.

K	1	2	3	4	5
R-1	45.81	46.54	48.13	48.18	48.25
Pool gen. (s)	76	191	267	290	359

We study the effect of parameter K in Agent-R, representing the state space of Agent-R, to choose one memory from a candidate pool with a size of K. In Table 7, we observe a performance improvement as K increases from 1 to 3, and then remains stable. Particularly, when $K = 1$, M-RAG exhibits the worst performance, possibly due to the limited exploration of potential memories for generating improved hypotheses. We choose the setting of $K = 3$, as it demonstrates effective performance, and runs reasonably fast for generating the pool.

5 Conclusion and Limitations

In this paper, we propose a multiple partition paradigm for RAG, which aims to refine retrieval processes and emphasize pivotal memories to improve overall performance. Additionally, we introduce M-RAG, a novel framework with multi-agent reinforcement learning, which addresses key challenges inherent in executing RAG across multiple partitions. The training objective of M-RAG is well aligned with that of text generation tasks, showcasing its potential to enhance system performance explicitly. Through extensive experiments conducted on seven datasets for three language generation tasks, we validate the effectiveness of M-RAG.

For limitations, we conduct experiments with quantized versions of language models due to computational constraints. However, the observed effectiveness gains are expected to remain consistent across different model sizes and should not significantly impact the overall trends of various RAG methods. Further, although the parameters of the LLMs remain fixed and only the parameters of Agent-S and Agent-R are trained, the training efficiency is limited, as indicated by the training time complexity discussed in Section 3.4. This is due to the necessity of querying the LLMs during the training process. In future work, we intend to explore solutions to overcome these limitations.

Bảng 4: So sánh M-RAG trên nhiều LLM 7B khác nhau.

Thạc sĩ Luật	VẢI	Tóm tắt Bản dịch (Es En)		Đối thoại
		R-1 R-2 RL	Màu xanh	B-1B-2
Gemma 7B Selfmem 31,38 9,97 25,07			24,61	15,56 7,91
Gemma 7B M-RAG 33,81 12,93 27,82 Mistral 7B			26,92	18,15 9,95
Selfmem 35,40 12,68 27,06			26,26	18.28 10.05
Mistral 7B M-RAG 37,47 13,24 30,49			32,65	24,52 11,53

Bảng 5: Nghiên cứu cắt bỏ.

Linh kiện M-	R-1 R-2 RL
RAG	48,13 24,66 39,43
không có Agent-S (DB đơ n)	44,20 22,72 37,40
không có Agent-R (tham lam)	45,75 23,21 38,28
không có Agent-S và Agent-R	43,82 22,07 37,44

Bảng 6: Tác động của số lư ợng M trong Agent-S.

Tôi	1	2	3	4	5
R-1	44,20	44,53	46,27	48,13	47,21
Chỉ số xây dựng (s)	299	278	257	246	227
Truy xuất (s)	0,61	1,09	1,54	2,19	2,59
Thế hệ (s)	83,59	84,88	82,81	82,89	86,64

toàn bộ cơ sở dữ liệu cho RAG; chúng tôi thay thế Agent-R với một quy tắc tham lam để chọn một bộ nhớ ứng viên từ hồ bơ i theo Phư ơ ng trình 3; và chúng tôi loại bỏ cả hai tác nhân, làm suy thoái thành Naive RAG. Các kết quả đư ợc trình bày trong Bảng 5, chứng minh rằng cả hai tác nhân đều góp phần vào hiệu suất cải thiện. Cụ thể, việc loại bỏ Agent-S dẫn đến sự suy giảm đáng kể trong R-1 từ 48,13 xuống 44.20. Điều này nhấn mạnh vai trò của nhiều thiết lập phân vùng để nâng cao hiệu suất tổng thể. Hơ n nữa, việc loại bỏ Agent-R dẫn đến việc giảm trong R-1 từ 48,13 xuống 45,75. Sự suy giảm này là do hiệu quả của Agent-R trong việc học lựa chọn bộ nhớ một cách năng động, trái ngư ợc với việc dựa vào một quy tắc cố định để ra quyết định.

(7) Nghiên cứu tham số (Không gian trạng thái Agent-S M). Chúng tôi nghiên cứu tác động của tham số M, tham số này điều khiển không gian trạng thái của Agent-S và tư ơ ng ứng với số lư ợng phân vùng. Trong Bảng 6, chúng tôi quan sát thiết lập M = 4 mang lại hiệu quả tốt nhất trong khi vẫn duy trì thời gian chạy hợp lý về mặt xây dựng chỉ mục, truy xuất và tạo ra. Điều này phù hợp với các nghiên cứu thực nghiệm đư ợc minh họa trong Hình 1 (a). Khi M = 1, nó giảm xuống còn một cơ sở dữ liệu cho RAG. Khi M tăng, việc xây dựng chỉ mục tăng tốc trên các phân vùng nhỏ hơ n, trong khi thời gian truy xuất tăng lên một chút do bổ sung thời gian cần thiết để xây dựng trạng thái bằng cách truy vấn mỗi phân vùng. Như mong đợi, thời gian truy xuất là nhỏ hơ n nhiều so với thời gian tạo ra ngôn ngữ.

(8) Nghiên cứu tham số (Không gian trạng thái K của tác nhân-R).

Bảng 7: Tác động của số lư ợng K trong Agent-R.

K	1	2	3	4	5
R-1	45,81	46,54	48,13	48,18	48,25
Thế hệ hồ bơ i. (các)	76	191	267	290	359

Chúng tôi nghiên cứu tác động của tham số K trong Agent-R, đại diện cho không gian trạng thái của Agent-R, để lựa chọn một bộ nhớ từ một nhóm ứng viên có kích thư ớc là K. Trong Bảng 7, chúng ta quan sát thấy sự cải thiện hiệu suất khi K tăng từ 1 đến 3, và sau đó duy trì ổn định. Đặc biệt, khi K = 1, M-RAG thể hiện hiệu suất tệ nhất, có thể là do hạn chế khám phá những ký ức tiềm năng để tạo ra cải thiện các giả thuyết. Chúng tôi chọn bối cảnh của K = 3, vì nó thể hiện hiệu suất hiệu quả, và chạy khá nhanh để tạo nhóm.

5 Kết luận và hạn chế

Trong bài báo này, chúng tôi đề xuất một phân vùng đa mô hình cho RAG, nhằm mục đích tinh chỉnh việc truy xuất quá trình và nhấn mạnh các ký ức quan trọng để cải thiện hiệu suất tổng thể. Ngoài ra, chúng tôi giới thiệu M -RAG, một khuôn khổ mới với nhiều tác nhân học tăng cư ờng, giải quyết những thách thức chính vốn có trong việc thực hiện RAG trên nhiều phân vùng. Mục tiêu đào tạo của M-RAG là tốt phù hợp với các tác vụ tạo văn bản, thể hiện rõ tiềm năng của nó trong việc nâng cao hiệu suất hệ thống . Thông qua các thí nghiệm mở rộng đư ợc tiến hành trên bảy tập dữ liệu cho ba thế hệ ngôn ngữ nhiệm vụ, chúng tôi xác nhận tính hiệu quả của M-RAG.

Để hạn chế, chúng tôi tiến hành các thí nghiệm với các phiên bản lư ợng tử của các mô hình ngôn ngữ do các ràng buộc tính toán. Tuy nhiên, các mức tăng hiệu quả quan sát đư ợc dự kiến sẽ vẫn nhất quán trên các kích thư ớc mô hình khác nhau và không nên tác động đáng kể đến xu hư ớng chung của nhiều Phư ơ ng pháp RAG. Hơ n nữa, mặc dù các tham số của LLM vẫn cố định và chỉ có các tham số của Agent-S và Agent-R đư ợc đào tạo, việc đào tạo hiệu quả bị hạn chế, như đư ợc chỉ ra bởi chư ơ ng trình đào tạo độ phức tạp về thời gian đư ợc thảo luận trong Phần 3.4. Đây là do nhu cầu phải truy vấn các LLM trong quá trình đào tạo. Trong công việc tư ơ ng lai, chúng tôi dự định tìm ra giải pháp để khắc phục những hạn chế này.

References

Marah Abdin, Jyoti Aneja, ebastien Bubeck, and Caio Cesar Teodoro Mendes et al. 2023. Phi-2: The surprising power of small language models. https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models.

Nathan Anderson, Caleb Wilson, and Stephen D. Richardson. 2022. Lingua: Addressing scenarios for live interpretation and automatic dubbing. In *AMTA*, pages 202–209.

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

V. Blagojevi. 2023. Enhancing rag pipelines in haystack: Introducing diversityranker and lostinthemiddler-anker. *https://towardsdatascience.com/enhancing-rag-pipelines-in-haystack-45f14e2bc9f5*.

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. *NeurIPS*, 33:1877–1901.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. 2023. Walking down the memory maze: Beyond context limit through interactive reading. *CoRR*, abs/2310.05029.

Daixuan Cheng, Shaohan Huang, Junyu Bi, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Furu Wei, Weiwei Deng, and Qi Zhang. 2023a. UPRISE: universal prompt retrieval for improving zero-shot evaluation. In *EMNLP*, pages 12318–12337.

Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao, and Rui Yan. 2023b. Lift yourself up: Retrieval-augmented text generation with self memory. *NeurIPS*.

Woon Sang Cho, Pengchuan Zhang, Yizhe Zhang, Xiujun Li, Michel Galley, Chris Brockett, Mengdi Wang, and Jianfeng Gao. 2018. Towards coherent and cohesive long-form text generation. *CoRR*.

Zhuyun Dai, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. 2023. Promptagator: Few-shot dense retrieval from 8 examples. In *ICLR*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *CoRR*, abs/2312.10997.

Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, et al. 2023. Filtered-diskann: Graph algorithms for approximate nearest neighbor search with filters. In *WWW*, pages 3406–3416.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O. K. Li. 2018. Search engine guided neural machine translation. In *AAAI*, pages 5133–5140. AAAI Press.

Rentong Guo, Xiaofan Luan, Long Xiang, Xiao Yan, Xiaomeng Yi, Jigao Luo, Qianya Cheng, Weizhi Xu, Jiarui Luo, Frank Liu, et al. 2022. Manu: a cloud native vector database management system. *PVLDB*, 15(12):3548–3561.

Yikun Han, Chunjiang Liu, and Pengfei Wang. 2023. A comprehensive survey on vector database: Storage and retrieval technique, challenge. *CoRR*.

Nabil Hossain, Marjan Ghazvininejad, and Luke Zettlemoyer. 2020. Simple and effective retrieve-edit-rerank text generation. In *ACL*, pages 2532–2538.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, pages 604–613.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. 2019. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *CoRR*.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *TPAMI*, 33(1):117–128.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, and et al. 2023a. Mistral 7b. *CoRR*, abs/2310.06825.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, and Arthur Mensch et al. 2024. Mixtral of experts. *CoRR*, abs/2401.04088.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023b. Llmlingua: Compressing prompts for accelerated inference of large language models. In *EMNLP*, pages 13358–13376.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Julia Kreutzer, Shahram Khadivi, Evgeny Matusov, and Stefan Riezler. 2018. Can neural machine translation be improved with user feedback? *CoRR*.

Carolin Lawrence and Stefan Riezler. 2018. Improving a neural semantic parser by counterfactual learning from human bandit feedback. *CoRR*.

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. *NeurIPS*, 33:9459–9474.

Tài liệu tham khảo

Marah Abdin, Jyoti Aneja, ebastien Bubeck và Caio Cesar Teodoro Mendes cùng cộng sự. 2023. Phi-2: Sức mạnh đáng kinh ngạc của các mô hình ngôn ngữ nhỏ. https://www.microsoft.com/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models .

Nathan Anderson, Caleb Wilson và Stephen D. Richardson. 2022. Lingua: Xử lý các tình huống cho phiên dịch trực tiếp và lồng tiếng tự động. Trong AMTA, trang 202–209.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil và Hannaneh Hajishirzi. 2023. Tự vấn: Học cách thu thập, sáng tạo và phê bình thông qua quá trình tự phản ánh. CoRR, tuyệt đối/2310.11511.

V. Blagojevi. 2023. Cải thiện đư ờng ống rác thải ở Haystack: Giới thiệu diversityranker và lostinthemiddler-anker. https://towardsdatascience.com/enhancing- rag-pipelines-in-haystack-45f14e2bc9f5.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Các mô hình ngôn ngữ là những ngư ời học ít lần . NeurIPS, 33:1877–1901.

Howard Chen, Ramakanth Pasunuru, Jason Weston và Asli Celikyilmaz. 2023. Đi bộ xuống mê cung trí nhớ: Vư ợt qua giới hạn ngữ cảnh thông qua đọc tư ơ ng tác. CoRR, abs/2310.05029.

Daixuan Cheng, Shaohan Huang, Junyu Bi, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Furu Wei, Weiwei Deng và Qi Zhang. 2023a. UPRISE: truy xuất nhanh chóng toàn diện để cải thiện việc đánh giá không bắn đư ợc . Trong EMNLP, trang 12318–12337.

Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao và Rui Yan. 2023b. Nâng cao bản thân: Tạo văn bản tăng cư ờng truy xuất bằng khả năng tự ghi nhớ. Thần kinhIPS.

Woon Sang Cho, Pengchuan Zhang, Yizhe Zhang, Xiu- jun Li, Michel Galley, Chris Brockett, Mengdi Wang và Jianfeng Gao. 2018. Hư ớng tới việc tạo ra văn bản dạng dài mạch lạc và gắn kết . CoRR.

Zhuyun Dai, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall và Ming-Wei Chang. 2023. Promptagator: Thu thập dữ liệu dày đặc với ít ảnh chụp từ 8 ví dụ. Trong ICLR.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang và Haofen Wang. 2023. Thế hệ tăng cư ờng truy xuất cho các mô hình ngôn ngữ lớn: Một cuộc khảo sát. CoRR, abs/2312.10997.

Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, và những ngư ời khác. 2023. Filtered-diskann: Thuật toán đồ thị để tìm kiếm lân cận gần nhất bằng bộ lọc. Trong WWW, trang 3406–3416.

Jiatao Gu, Yong Wang, Kyunghyun Cho và Victor OK Li. 2018. Dịch máy thần kinh hư ớng dẫn công cụ tìm kiếm. Trong AAAI, trang 5133–5140. AAAI Press.

Rentong Guo, Xiaofan Luan, Long Xiang, Xiao Yan, Xiaomeng Yi, Jigao Luo, Qianya Cheng, Weizhi Xu, Jiarui Luo, Frank Liu, và những ngư ời khác. 2022. Manu: hệ thống quản lý cơ sở dữ liệu vector gốc trên nền tảng đám mây. PVLDB, 15(12):3548–3561.

Yikun Han, Chunjiang Liu và Pengfei Wang. 2023. Một khảo sát toàn diện về cơ sở dữ liệu vectơ : Kỹ thuật lư u trữ và truy xuất, thách thức. CoRR.

Nabil Hossain, Marjan Ghazvininejad và Luke Zettle- moyer. 2020. Tạo văn bản lấy lại-sửa-xếp hạng lại đơ n giản và hiệu quả. Trong ACL, trang 2532–2538.

Piotr Indyk và Rajeev Motwani. 1998. Những ngư ời hàng xóm gần nhất gần đúng: hư ớng tới việc loại bỏ lời nguyền của tính đa chiều. Trong STOC, trang 604–613.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu và Rosalind Picard. 2019. Way off- chính sách lô học tăng cư ờng sâu về sở thích ngầm của con ngư ời trong đối thoại. CoRR.

Herve Jegou, Matthijs Douze và Cordelia Schmid. 2010. Lư ợng tử hóa sản phẩm cho tìm kiếm hàng xóm gần nhất. TPAMI, 33(1):117–128.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Men-sch, Chris Bamford và cộng sự. 2023a. Mistral 7b. CoRR, tuyệt đối/2310.06825.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, và Arthur Mensch và cộng sự. 2024. Sự kết hợp của các chuyên gia. CoRR, tuyệt đối/2401.04088.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang và Lili Qiu. 2023b. Llmlingua: Nén các lời nhắc để tăng tốc suy luận của các mô hình ngôn ngữ lớn . Trong EMNLP, trang 13358–13376.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen và Wen-tau Yih. 2020. Truy xuất đoạn văn dày đặc để trả lời câu hỏi miền mở. Trong EMNLP (1), trang 6769–6781.

Julia Kreutzer, Shahram Khadivi, Evgeny Matusov và Stefan Riezler. 2018. Liệu dịch máy thần kinh có thể đư ợc cải thiện bằng phản hồi của ngư ời dùng không? CoRR.

Carolin Lawrence và Stefan Riezler. 2018. Cải thiện trình phân tích ngữ nghĩa thần kinh bằng cách học phản chứng từ phản hồi của con ngư ời. CoRR.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Hein-rich Küttler, Mike Lewis, Wen-tau Yih, Tim Rock-täschel, et al. 2020. Tạo thế hệ tăng cư ờng truy xuất cho các tác vụ nlp chuyên sâu về kiến thức. NeurIPS, 33:9459–9474.

Jinpeng Li, Yingce Xia, Rui Yan, Hongda Sun, Dongyan Zhao, and Tie-Yan Liu. 2021. Stylized dialogue generation with multi-pass dual learning. In *NeurIPS*, pages 28470–28481.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *HLT-NAACL*, pages 110–119.

Xinze Li, Zhenghao Liu, Chenyan Xiong, Shi Yu, Yu Gu, Zhiyuan Liu, and Ge Yu. 2023. Structure-aware language model pretraining improves dense retrieval on structured data. In *ACL (Findings)*, pages 11560–11574.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. In *IJCNLP(1)*, pages 986–995.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Scott Yih. 2023. RA-DIT: retrieval-augmented dual instruction tuning. *CoRR*, abs/2310.01352.

Ron Litman, Oron Anschel, Shahar Tsiper, Roee Litman, Shai Mazor, and R. Manmatha. 2020. SCATTER: selective context attentional scene text recognizer. In *CVPR*, pages 11959–11969.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *EMNLP*, pages 5303–5315.

Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *TPAMI*, 42(4):824–836.

Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. 2014. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45:61–68.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, and et al. 2024. Gemma: Open models based on gemini research and technology. *CoRR*, abs/2403.08295.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *CoRR*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, and Jeff Wu et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *CoRR*, abs/2112.09332.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *EMNLP*, pages 1797–1807.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. 2022. Text and code embeddings by contrastive pre-training. *CoRR*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *NeurIPS*, 35:27730–27744.

James Jie Pan, Jianguo Wang, and Guoliang Li. 2023. Survey of vector database management systems. *CoRR*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *WMT*, pages 186–191.

Eva Sharma, Chen Li, and Lu Wang. 2019. BIGPATENT: A large-scale dataset for abstractive and coherent summarization. In *ACL (1)*, pages 2204–2213.

Aleksandrs Slivkins et al. 2019. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Dániel Varga. 2006. The jrc-acquis: A multilingual aligned parallel corpus with 20+ languages. In *LREC*, pages 2142–2147.

Hugo Touvron, Louis Martin, Kevin Stone, and Peter Albert et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Shuohang Wang, Yichong Xu, Yuwei Fang, Yang Liu, Siqi Sun, Ruochen Xu, Chenguang Zhu, and Michael Zeng. 2022. Training data is more valuable than you think: A simple and effective method by retrieving from training data. In *ACL*, pages 3170–3179.

Xintao Wang, Qianwen Yang, Yongting Qiu, Jiaqing Liang, Qianyu He, Zhouhong Gu, Yanghua Xiao, and Wei Wang. 2023a. Knowledgpt: Enhancing large language models with retrieval and storage access on knowledge bases. *CoRR*, abs/2308.11761.

Zheng Wang, Bingzheng Gan, and Wei Shi. 2024. Multimodal query suggestion with multi-agent reinforcement learning from human feedback. In *WWW*, pages 1374–1385.

Zheng Wang, Cheng Long, Gao Cong, and Christian S. Jensen. 2023b. Collectively simplifying trajectories in a database: A query accuracy driven approach. *CoRR*, abs/2311.11204.

Jinpeng Li, Yingce Xia, Rui Yan, Hongda Sun, Dongyan Zhao và Tie-Yan Liu. 2021. Tạo đối thoại cách điệu với phư ơ ng pháp học tập kép nhiều lư ợt. Trong NeurIPS, trang 28470–28481.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao và Bill Dolan. 2016. Một hàm mục tiêu thúc đẩy sự đa dạng cho các mô hình hội thoại thần kinh. Trong HLT-NAACL, trang 110–119.

Xinze Li, Zhenghao Liu, Chenyan Xiong, Shi Yu, Yu Gu, Zhiyuan Liu và Ge Yu. 2023. Việc đào tạo trư ớc mô hình ngôn ngữ nhận biết cấu trúc giúp cải thiện khả năng truy xuất dày đặc trên dữ liệu có cấu trúc. Trong ACL (Kết quả), trang 11560–11574.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao và Shuzi Niu. 2017. Dailydialog: Tập dữ liệu hội thoại nhiều lư ợt đư ợc gắn nhãn thủ công. Trong IJCNLP(1), trang 986–995.

Chin-Yew Lin. 2004. ROUGE: Một gói để tự động đánh giá tóm tắt. Trong Text Summarization Branches Out, trang 74–81.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Wei-jia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer và Scott Yih. 2023. RA-DIT: điều chỉnh hư ớng dẫn kép tăng cư ờng truy xuất. CoRR, abs/2310.01352.

Ron Litman, Oron Anschel, Shahar Tsiper, Roee Lit-man, Shai Mazor và R. Manmatha. 2020. SCAT-TER: công cụ nhận dạng văn bản cảnh chú ý ngữ cảnh có chọn lọc . Trong CVPR, trang 11959–11969.

Xinbei Ma, Yeyun Gong, Peng Cheng He, Hai Zhao và Nan Duan. 2023. Viết lại truy vấn cho các mô hình ngôn ngữ lớn đư ợc tăng cư ờng truy xuất. EMNLP, trang 5303–5315.

Yu A Malkov và Dmitry A Yashunin. 2018. Tìm kiếm lân cận gần nhất hiệu quả và mạnh mẽ bằng cách sử dụng đồ thị thế giới nhỏ có thể điều hư ớng theo thứ bậc. TPAMI, 42(4):824–836.

Yury Malkov, Alexander Ponomarenko, Andrey Logvi- nov, và Vladimir Krylov. 2014. Thuật toán láng giềng gần nhất xấp xỉ dựa trên đồ thị thế giới nhỏ có thể điều hư ớng. Hệ thống thông tin, 45:61–68.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju và những ngư ời khác. 2024. Gemma: Các mô hình mở dựa trên nghiên cứu và công nghệ của gemini. CoRR, tuyệt đối/2403.08295.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra và Martin Riedmiller. 2013. Chơ i atari với học tăng cư ờng sâu. CoRR.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji và Jeff Wu cùng cộng sự. 2021. Webgpt: Trả lời câu hỏi bằng trình duyệt với phản hồi của con ngư ời. CoRR, abs/2112.09332.

Shashi Narayan, Shay B. Cohen và Mirella Lapata. 2018. Đừng cho tôi biết chi tiết, chỉ cần tóm tắt! mạng nơ - ron tích chập nhận biết chủ đề để tóm tắt cực đoan. Trong EMNLP, trang 1797–1807.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford , Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, và những ngư ời khác. 2022. Nhúng văn bản và mã bằng cách đào tạo trư ớc tư ơ ng phản. CoRR.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, và những ngư ời khác. 2022. Đào tạo các mô hình ngôn ngữ để tuân theo hư ớng dẫn với phản hồi của con ngư ời. NeurIPS, 35:27730–27744 .

James Jie Pan, Jianguo Wang và Guoliang Li. 2023. Khảo sát các hệ thống quản lý cơ sở dữ liệu vector. Đồng hồ đo

Matt Post. 2018. Lời kêu gọi làm rõ trong việc báo cáo điểm BLEU. Trong WMT, trang 186–191.

Eva Sharma, Chen Li và Lu Wang. 2019. BIG-PATENT: Một tập dữ liệu quy mô lớn để tóm tắt trừu tư ợng và mạch lạc. Trong ACL (1), trang 2204–2213 .

Aleksandrs Slivkins và cộng sự. 2019. Giới thiệu về máy đánh bạc nhiều tay. Nền tảng và xu hư ớng® trong Học máy, 12(1-2):1–286.

Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis và Dániel Varga. 2006. jrc-acquis: Kho ngữ liệu song song đư ợc căn chỉnh đa ngôn ngữ với hơ n 20 ngôn ngữ. Trong LREC, trang 2142–2147.

Hugo Touvron, Louis Martin, Kevin Stone và Peter Albert và cộng sự. 2023. Llama 2: Nền tảng mở và các mô hình trò chuyện đư ợc tinh chỉnh. CoRR, abs/2307.09288.

Shuohang Wang, Yichong Xu, Yuwei Fang, Yang Liu, Siqi Sun, Ruochen Xu, Chenguang Zhu và Michael Zeng. 2022. Dữ liệu đào tạo có giá trị hơ n bạn nghĩ: Một phư ơ ng pháp đơ n giản và hiệu quả bằng cách lấy từ dữ liệu đào tạo. Trong ACL, trang 3170–3179.

Xintao Wang, Qianwen Yang, Yongting Qiu, Jiaqing Liang, Qianyu He, Zhouhong Gu, Yanghua Xiao và Wei Wang. 2023a. Knowledgpt: Tăng cư ờng các mô hình ngôn ngữ lớn với khả năng truy xuất và lư u trữ trên cơ sở tri thức. CoRR, abs/2308.11761.

Zheng Wang, Bingzheng Gan và Wei Shi. 2024. Gợi ý truy vấn đa phư ơ ng thức với học tăng cư ờng đa tác nhân từ phản hồi của con ngư ời. Trong WWW, trang 1374–1385.

Zheng Wang, Cheng Long, Gao Cong và Christian S. Jensen. 2023b. Đơ n giản hóa các quỹ đạo trong cơ sở dữ liệu: Một phư ơ ng pháp tiếp cận dựa trên độ chính xác của truy vấn. CoRR, tuyệt đối/2311.11204.

Zheng Wang, Cheng Long, Gao Cong, and Qianru Zhang. 2021. Error-bounded online trajectory simplification with multi-agent reinforcement learning. In *KDD*, pages 1758–1768.

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. 2021a. Recursively summarizing books with human feedback. *CoRR*.

Sixing Wu, Ying Li, Minghui Wang, Dawei Zhang, Yang Zhou, and Zhonghai Wu. 2021b. More is better: Enhancing open-domain dialogue generation via multi-source heterogeneous knowledge. In *EMNLP*, pages 2286–2300.

Sixing Wu, Ying Li, Dawei Zhang, and Zhonghai Wu. 2022. KSAM: infusing multi-source knowledge into dialogue generation via knowledge source aware multi-head decoding. In *ACL (Findings)*, pages 353–363.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Xingrun Xing. 2023. Lm-cocktail: Resilient tuning of language models via model merging. *CoRR*, abs/2311.13534.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. RECOMP: improving retrieval-augmented lms with compression and selective augmentation. *CoRR*, abs/2310.04408.

Wenzhuo Xue, Hui Li, Yanguo Peng, Jiangtao Cui, and Yu Shi. 2017. Secure k nearest neighbors query for high-dimensional vectors in outsourced environments. *IEEE TBD*, 4(4):586–599.

Sanghyun Yi, Rahul Goel, Chandra Khatri, Alessandra Cervone, Tagyoung Chung, Behnam Hedayatnia, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tur. 2019. Towards coherent and engaging spoken dialog response generation using automatic conversation evaluators. *CoRR*.

Shengyao Zhuang, Bing Liu, Bevan Koopman, and Guido Zuccon. 2023. Open-source large language models are strong zero-shot query likelihood models for document ranking. In *EMNLP (Findings)*, pages 8807–8817.

A Appendix

A.1 Other Evaluation Metrics for Machine Translation

We utilize BLEURT [4] (with the checkpoint of BLEURT-20) and COMET [5] (with wmt22-comet-da to obtain features) to evaluate the performance of machine translation, and then compare M-RAG with the best baseline method, Selfmem, on the Mixtral 8 × 7B. The results are reported in Table 8.

[4]https://huggingface.co/spaces/evaluate-metric/bleurt
[5]https://huggingface.co/spaces/evaluate-metric/comet

Overall, we observe that M-RAG consistently outperforms Selfmem across diverse translation datasets, as evidenced by various evaluation metrics.

A.2 Further Discussion

Q1. Why applying RAG for summarization or translation?

Employing RAG for summarization or translation is based on two key factors: (1) We believe that the two tasks effectively capture the essence of text generation facilitated by LLMs; (2) the widespread adoption of summarization and translation tasks in retrieval-augmented literature (Cheng et al., 2023b; Gu et al., 2018; Hossain et al., 2020) provides a standardized and comparable testbed for benchmarking our method. Here, certain text pairs are stored within an external database, such as (document, summary) pairs for summarization or (context, response) pairs for dialogue generation. These pairs are retrieved from the database and serve as demonstration examples to guide a LLM in conducting text generations. The underlying rationale of this paradigm is that better demonstrations typically prompt better generation outcomes.

Q2. Why applying such partitioning, what intuition behind that, instead of improving the quality of retrieval or introduce more dimensions in the scoring function to account for categories/partitions?

We recognize that database partitioning plays a crucial role in efficiently managing a database. However, this aspect has been relatively underexplored in the context of RAG, despite the necessity of accessing an external database to obtain essential information for LLM generation. To address this gap, we investigate a multiple partition paradigm for executing RAG. The rationale behind this approach is intuitive: with various attributes associated with the data in a database, queries should ideally be matched with their corresponding attributed data, thereby filtering out noise data.

We discuss our choice of employing partitioning for RAG instead of two alternative approaches: (1) improving the quality of retrieval or (2) introduce more dimensions in the scoring function to account for categories/partitions.

For (1), improving retrieval quality typically emphasizes the effectiveness of AKNN search, often measured using metrics such as recall. However, this focus is not entirely aligned with the primary objective of RAG, which is to generate a good re-

Zheng Wang, Cheng Long, Gao Cong và Qianru Zhang. 2021. Đơn giản hóa quỹ đạo trực tuyến có giới hạn lỗi với học tăng cường đa tác nhân. Trong KDD, trang 1758–1768.

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike và Paul Christiano. 2021a. Tóm tắt sách theo cách đệ quy với con người phản hồi. CoRR.

Sixing Wu, Ying Li, Minghui Wang, Dawei Zhang, Yang Zhou và Zhonghai Wu. 2021b. Nhiều hơn là tốt hơn : Tăng cường thế hệ đối thoại miền mở thông qua kiến thức không đồng nhất đa nguồn. Trong EMNLP, trang 2286–2300.

Sixing Wu, Ying Li, Dawei Zhang và Zhonghai Wu. 2022. KSAM: truyền tải kiến thức đa nguồn vào tạo ra đối thoại thông qua nhận thức nguồn kiến thức giải mã nhiều đầu. Trong ACL (Phát hiện), trang 353–363.

Shitao Xiao, Zheng Liu, Peitian Zhang và Xin-grun Xing. 2023. Lm-cocktail: Điều chỉnh kiên cư ờng của các mô hình ngôn ngữ thông qua việc hợp nhất mô hình. CoRR, tuyệt đối/2311.13534.

Fangyuan Xu, Weijia Shi và Eunsol Choi. 2023. RECOMP: cải thiện các lms tăng cư ờng khả năng truy xuất với nén và tăng cư ờng chọn lọc. CoRR, cơ bản/2310.04408.

Wenzhuo Xue, Hui Li, Yanguo Peng, Jiangtao Cui, và Yu Shi. 2017. Truy vấn k láng giềng gần nhất an toàn đối với các vectơ có chiều cao trong môi trư ờng gia công ngoài . IEEE TBD, 4(4):586–599.

Sanghyun Yi, Rahul Goel, Chandra Khatri, Alessandra Cervone, Tagyoung Chung, Behnam Hedayatnia, Anu Venkatesh, Raefer Gabriel và Dilek Hakkani- Tur. 2019. Hư ớng tới lời nói mạch lạc và hấp dẫn tạo phản hồi hội thoại bằng cách sử dụng trình đánh giá hội thoại tự động. CoRR.

Shengyao Zhuang, Bing Liu, Bevan Koopman, và Guido Zuccon. 2023. Ngôn ngữ lớn mã nguồn mở các mô hình là các mô hình khả năng truy vấn zero-shot mạnh để xếp hạng tài liệu. Trong EMNLP (Phát hiện), các trang 8807–8817.

Phụ lục

A.1 Các số liệu đánh giá khác cho máy Bản dịch

Chúng tôi sử dụng BLEURT [4] (với trạm kiểm soát của BLEURT-20) và COMET da để có đư ợc [5] (với wmt22-sao chổi-các tính năng) để đánh giá hiệu suất của bản dịch máy, và sau đó so sánh M-RAG với phư ơng pháp cơ sở tốt nhất, Selfmem, trên Mixtral 8 × 7B. Kết quả đư ợc báo cáo trong Bảng 8.

[4]https://huggingface.co/spaces/evaluate-metric/bleurt
[5]https://huggingface.co/spaces/evaluate-metric/comet

Nhìn chung, chúng tôi nhận thấy rằng M-RAG luôn vư ợt trội hơn Selfmem trên nhiều tập dữ liệu dịch thuật khác nhau, như đư ợc chứng minh bằng nhiều số liệu đánh giá khác nhau.

A.2 Thảo luận thêm

Câu hỏi 1. Tại sao áp dụng RAG để tóm tắt hoặc dịch?

Việc sử dụng RAG để tóm tắt hoặc dịch thuật dựa trên hai yếu tố chính: (1) Chúng tôi tin rằng hai nhiệm vụ này nắm bắt hiệu quả bản chất của văn bản thế hệ đư ợc tạo điều kiện thuận lợi bởi LLM; (2) sự lan rộng việc áp dụng các nhiệm vụ tóm tắt và dịch thuật trong tài liệu đư ợc tăng cư ờng khả năng truy xuất (Cheng et al., 2023b; Gu và cộng sự, 2018; Hossain và cộng sự, 2020) cung cấp một nền tảng thử nghiệm chuẩn hóa và có thể so sánh đư ợc để đánh giá chuẩn mực phư ơng pháp của chúng tôi. Ở đây, một số cặp văn bản nhất định là đư ợc lư u trữ trong cơ sở dữ liệu bên ngoài, chẳng hạn như cặp (tài liệu, tóm tắt) để tóm tắt hoặc cặp (bối cảnh , phản hồi) để tạo hội thoại. Những các cặp đư ợc lấy từ cơ sở dữ liệu và phục vụ như ví dụ minh họa để hư ớng dẫn LLM trong việc tạo ra văn bản. Cơ sở lý luận cơ bản của mô hình này là các cuộc biểu tình tốt hơn thư ờng dẫn đến kết quả thế hệ tốt hơn.

Câu hỏi 2. Tại sao lại áp dụng phân vùng như vậy, trực giác đằng sau đó là gì, thay vì cải thiện chất lư ợng truy xuất hay đư a thêm nhiều chiều hơn vào hàm tính điểm để tính đến các danh mục/phân vùng?

Chúng tôi nhận ra rằng phân vùng cơ sở dữ liệu đóng vai trò một vai trò quan trọng trong việc quản lý cơ sở dữ liệu hiệu quả. Tuy nhiên, khía cạnh này vẫn chư a đư ợc khám phá đầy đủ trong bối cảnh của RAG, mặc dù có sự cần thiết của việc truy cập vào cơ sở dữ liệu bên ngoài để có đư ợc thông tin cần thiết thông tin cho thế hệ LLM. Để giải quyết vấn đề này khoảng cách, chúng tôi điều tra một mô hình phân vùng đa để thực hiện RAG. Cơ sở lý luận đằng sau cách tiếp cận này là trực quan: với các thuộc tính khác nhau đư ợc liên kết với dữ liệu trong cơ sở dữ liệu, các truy vấn lý tư ởng nhất nên đư ợc khớp với các thuộc tính tư ơng ứng của chúng dữ liệu, do đó lọc bỏ dữ liệu nhiễu.

Chúng tôi thảo luận về sự lựa chọn của chúng tôi trong việc sử dụng phân vùng cho RAG thay vì hai cách tiếp cận thay thế: (1) cải thiện chất lư ợng truy xuất hoặc (2) giới thiệu nhiều chiều hơn trong chức năng tính điểm để tính đến dành cho các danh mục/phân vùng.

Đối với (1), việc cải thiện chất lư ợng truy xuất thư ờng nhấn mạnh vào hiệu quả của tìm kiếm AKNN, thư ờng đư ợc đo bằng các số liệu như thu hồi. Tuy nhiên, sự tập trung này không hoàn toàn phù hợp với mục tiêu chính mục tiêu của RAG là tạo ra một sự phản hồi tốt

Table 8: Machine translation with BLEURT and COMET.

LLM	RAG	BLEURT				COMET			
		Es→En	En→Es	De→En	En→De	Es→En	En→Es	De→En	En→De
Mixtral 8 × 7B	Selfmem	63.63	53.26	59.93	59.91	75.65	55.28	60.41	52.13
Mixtral 8 × 7B	M-RAG	**71.74**	**63.66**	**66.77**	**70.99**	**82.66**	**80.29**	**67.33**	**85.14**

sponse. In the M-RAG framework, we prioritize the quality of LLM generation as an end-to-end metric explicitly guiding the retrieved information.

For (2), unlike attending to data categories or partitions, we observe that the multiple partition setup offers a cost-effective approach to enhance effectiveness, as confirmed in Figure 1. In this context, no additional computation associated with the LLM is required. Instead, we can keep the LLM frozen, and explore (via Agent-S) or revise (via Agent-R) a relevant memory. This typically leads to improved generation results for the LLM.

Q3. What is the motivation of the Agent-R and the revision of the retrieved memory?

M-RAG involves a Retrieval-then-Generation process employing LLMs, typically containing billions of parameters. Here, the LLM remains frozen while the retrieved memories undergo revision before being fed back into the LLM to enhance results. Common revision operations within the retrieved memory, such as re-ranking content (Blagojevi, 2023), eliminating irrelevant context (Anderson et al., 2022), summarizing key information (Chen et al., 2023), and generating candidates for selection (Cheng et al., 2023b), have been extensively studied in retrieval-augmented literature, as highlighted in the survey paper (Gao et al., 2023). In our work, we conceptualize memory revision as a Markov Decision Process (MDP) and investigate a reinforcement learning solution employing the proposed Agent-R for this operation.

Q4. M-RAG relies on the partitioning strategy. If the partitions are not well-optimized, it could lead to suboptimal retrieval and generation performance?

The performance of M-RAG is preserved through several measures. First, we conduct an empirical study, depicted in Figure 1, to investigate a partitioning strategy that outperforms retrieval from the entire database. This serves as a prerequisite for achieving performance improvements. Additionally, building upon this prerequisite, the challenge shifts to identifying suitable partitions and enhancing data quality within them, tasks that are addressed concurrently by two agents. As illustrated in the ablation study presented in Table 5, performance gains are still attainable even if one agent fails, suggesting that performance improvements can be expected with the M-RAG approach.

Bảng 8: Dịch máy với BLEURT và COMET.

Thạc sĩ Luật	VẢI	BLEURT								SAO CHỔI		
		Es	En	En	Es	De	En	En	De	Es	En	En	Es	De	En	En	De
Hỗn hợp 8 × 7B Selfmem		63,63		53,26		59,93		59,91		75,65		55,28		60,41		52,13	
Hỗn hợp 8 × 7B M-RAG		71,74		63,66		66,77		70,99		82,66		80,29		67,33		85,14	

sponse. Trong khuôn khổ M-RAG , chúng tôi ưu tiên chất lượng của thế hệ LLM như một thước đo đầu cuối hướng dẫn rõ ràng thông tin thu thập được.

Đối với (2), không giống như việc chú ý đến các loại dữ liệu hoặc phân vùng, chúng tôi quan sát thấy rằng nhiều phân vùng thiết lập cung cấp một cách tiếp cận hiệu quả về chi phí để nâng cao hiệu quả, như đã xác nhận trong Hình 1. Trong này bối cảnh, không có tính toán bổ sung nào liên quan đến LLM là bắt buộc. Thay vào đó, chúng ta có thể giữ LLM bị đóng băng và khám phá (thông qua Agent-S) hoặc sửa đổi (thông qua Agent-R) một bộ nhớ có liên quan. Điều này thường dẫn đến kết quả đào tạo LLM được cải thiện.

Câu hỏi 3. Động cơ của Agent-R là gì và việc xem xét lại ký ức đã thu thập được?

M-RAG liên quan đến quá trình Truy xuất rồi Tạo sử dụng LLM, thường chứa hàng tỷ tham số. Ở đây, LLM vẫn bị đóng băng trong khi những ký ức được thu thập sẽ được xem xét lại trước khi đưa trở lại LLM để nâng cao kết quả. Các hoạt động sửa đổi chung trong các truy xuất bộ nhớ, chẳng hạn như xếp hạng lại nội dung (Blagojevi, 2023), loại bỏ bối cảnh không liên quan (Anderson et al., 2022), tóm tắt thông tin chính (Chen et al., 2023) và việc tạo ra các ứng viên để lựa chọn (Cheng et al., 2023b) đã được tiến hành rộng rãi được nghiên cứu trong tài liệu tăng cường truy xuất, như được nêu bật trong bài báo khảo sát (Gao et al., 2023). Trong công việc của chúng tôi, chúng tôi khái niệm hóa việc sửa đổi bộ nhớ như một Quá trình quyết định Markov (MDP) và điều tra một giải pháp học tăng cường sử dụng đề xuất Agent-R cho hoạt động này.

Q4. M-RAG dựa vào chiến lược phân vùng. Nếu các phân vùng không được tối ưu hóa tốt, nó có thể dẫn đến hiệu suất thu hồi và tạo dữ liệu không tối ưu ?

Hiệu suất của M-RAG được bảo toàn thông qua nhiều biện pháp. Đầu tiên, chúng tôi tiến hành một thử nghiệm nghiên cứu, được mô tả trong Hình 1, để điều tra một chiến lược phân vùng vượt trội hơn việc truy xuất từ toàn bộ cơ sở dữ liệu. Điều này đóng vai trò là điều kiện tiên quyết để đạt được cải thiện hiệu suất. Ngoài ra, dựa trên điều kiện tiên quyết này, thách thức chuyển sang xác định các phân vùng phù hợp và nâng cao chất lượng dữ liệu trong đó, các nhiệm vụ đó là được giải quyết đồng thời bởi hai tác nhân. Như minh họa trong nghiên cứu cắt bỏ được trình bày trong Bảng 5, hiệu suất tăng vẫn có thể đạt được ngay cả khi một tác nhân không thành công, cho thấy có thể mong đợi cải thiện hiệu suất với phương pháp M-RAG.