Optimizing Multi-Hop Document Retrieval Through Intermediate Representations

Jiaen Lin
Independent Researcher
ljе23@mails.tsinghua.edu.cn

Jingyu Liu
Renmin University of China
liujy1016@ruc.edu.cn

Abstract

Retrieval-augmented generation (RAG) encounters challenges when addressing complex queries, particularly multi-hop questions. While several methods tackle multi-hop queries by iteratively generating internal queries and retrieving external documents, these approaches are computationally expensive. In this paper, we identify a three-stage information processing pattern in LLMs during layer-by-layer reasoning, consisting of extraction, processing, and subsequent extraction steps. This observation suggests that the representations in intermediate layers contain richer information compared to those in other layers. Building on this insight, we propose Layer-wise RAG (L-RAG). Unlike prior methods that focus on generating new internal queries, L-RAG leverages intermediate representations from the middle layers, which capture next-hop information, to retrieve external knowledge. L-RAG achieves performance comparable to multi-step approaches while maintaining inference overhead similar to that of standard RAG. Experimental results show that L-RAG outperforms existing RAG methods on open-domain multi-hop question-answering datasets, including MuSiQue, HotpotQA, and 2WikiMultiHopQA. The code is available in https://anonymous.4open.science/r/L-RAG-ADD5/.

1 Introduction

Large Language Models (LLMs) have made significant advancements in natural language processing, demonstrating exceptional performance across a variety of tasks (Muennighoff et al., 2023; Hendrycks et al., 2021), including complex multi-hop tasks. However, despite their success, LLMs often generate factually incorrect answers as they rely solely on their internal parametric memory, which struggles with tasks requiring external knowledge (Lewis et al., 2020). To address this limitation, Retrieval-Augmented Generation (RAG) frameworks (Gao et al., 2023) have been developed, utilizing semantic similarity metrics to retrieve relevant external documents, which are subsequently integrated into the generation context. Nevertheless, RAG still faces challenges, particularly in handling complex queries that require multi-hop document retrieval.

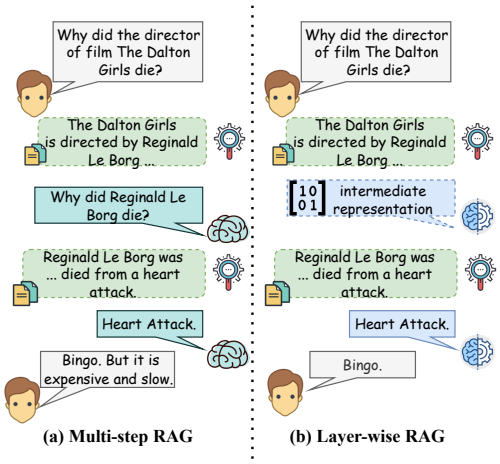


Figure 1: Comparison of Multi-step RAG and Layer-wise RAG. The left panel illustrates Multi-step RAG, which involves multiple rounds of reasoning to generate retrieval queries. In contrast, the right panel shows Layer-wise RAG, which requires only a single round of reasoning to produce intermediate representations for retrieval.

Multi-step methods, such as Self-ask (Press et al., 2023a) and ITER-RETGEN (Shao et al., 2023a), address the challenges of multi-hop reasoning and document retrieval by decomposing complex queries into manageable steps and iteratively refining the retrieval process (Jiang et al., 2023; Shao et al., 2023b; Ma et al., 2023). Over multiple iterations, the response is refined until a final answer is produced. As illustrated in Figure 1, while multi-step methods significantly improve

retrieval and reasoning capabilities, they often result in higher computational overhead due to their iterative nature. Several variants aim to address this issue. Adaptive-RAG (Jeong et al., 2024) simplify the reasoning process for less complex queries. However, for more complex queries, multiple full inference iterations are still required. Efficient RAG (Zhuang et al., 2024) get rid of relying on LLM calls at each step, using the smaller model to generate the special token for information processing. Despite these advancements, such methods still tend to address the problem in generating human understandable query, overlooking the next-hop information contained in intermediate representations during reasoning. There is a question we try to find out that **how can we leverage the information contain to achieve performance comparable to multi-step approaches while maintaining a low inference overhead similar to vanilla RAG?**

LLMs demonstrate the ability to extract entities and their relationships during inference (Geva et al., 2023), including implicit reasoning for bridge entities in multi-hop tasks (Biran et al., 2024; Wang et al., 2024; Brinkmann et al., 2024). Moreover, studies suggest that intermediate representations within LLMs often contain richer information than the final layer (Skean et al., 2024). Through an analysis of LLM weight matrices using Singular Value Decomposition (SVD), we identify the computational regime of LLMs: initial layers primarily perform raw information extraction, middle layers engage in contextual processing, and final layers focus on extracting information for token generation. This analysis suggests that intermediate layer representations, rather than those in the final layer, are more suitable for document retrieval.

In light of this, we introduce **Layer-wise Retrieval-Augmented Generation (L-RAG)**, which leverages intermediate representations from the middle layers to retrieve external information. As illustrated in Figure 2, we train a contriever-based representations retriever to effectively utilize these intermediate representations for retrieving higher-hop documents.

Our experimental results demonstrate that the proposed method achieves competitive performance while maintaining computational efficiency comparable to vanilla-RAG systems.

Our key contributions include:

- We propose an SVD-based decomposition approach to analyze weight matrices, identifying the three-stage information processing pattern in LLMs, and validate this through the LogitLens method.
- We introduce a novel method that utilizes intermediate representations to retrieve external information, which is then incorporated into the prompt for final generation.
- We rigorously evaluate the proposed method across several open-source LLMs, demonstrating its effectiveness in enhancing inference performance.

2 Related Works

2.1 Retrieval-Augmented Generation

To address factual inaccuracies in language models, Retrieval-Augmented Generation (RAG) integrates external knowledge by providing relevant passages as contextual input, thereby enhancing factual reasoning and enabling real-time utilization of information. However, RAG systems often struggle with complex, multi-hop queries (Yang et al., 2018; Trivedi et al., 2022; Ho et al., 2020), which require iterative synthesis of information.

Recent advancements in RAG have introduced multi-round retrieval strategies, including query rewriting (Ma et al., 2023; Shao et al., 2023b; Jiang et al., 2023) and self-questioning (Press et al., 2023b), aimed at improving performance on complex tasks. Nevertheless, these methods face a significant limitation: they incur increased computational latency and cost due to the repeated invocation of large language models (LLMs) for query generation. To address these challenges, several methodological variants have been proposed. Adaptive-RAG (Jeong et al., 2024) reduces inefficiencies by dynamically assessing query complexity and selectively activating retrieval processes, thus lowering inference overhead for simpler queries while retaining multi-round retrieval for more complex cases. Alternatively, Efficient RAG (Zhuang et al., 2024) uses a smaller model to iteratively generate queries and filter irrelevant information, bypassing the need for repeated LLM calls, thereby improving computational efficiency.

2.2 Reasoning Mechanisms of Transformers

A lines of research has employed empirical approaches to explore the internal workings of transformers (Delétang et al., 2023; Zhang et al., 2024; Fan et al., 2024) and their reasoning mechanisms

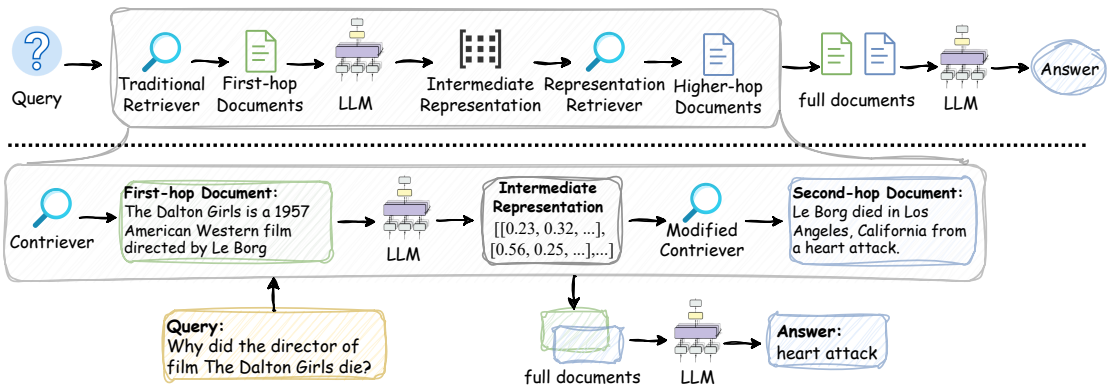


Figure 2: An overview and example of the L-RAG framework, which consists of three components: a traditional retriever, an LLM for generating intermediate representations, and the representation retriever. The area above the dashed line represents the framework, while the area below illustrates the demonstration. Initially, the traditional retriever (e.g., BM25, Contriever) retrieves relevant chunks from the corpus as the first-hop document. The LLM (e.g., LLaMA2-7B) then uses the query and first-hop context to generate an intermediate representation. This intermediate representation is used by the Modified Contriever trained by us to retrieve the higher-hop document. Finally, the LLM generates the final answer using the complete context.

(Haviv et al., 2023; Geva et al., 2023), particularly in the context of multi-step reasoning tasks (Brinkmann et al., 2024; Biran et al., 2024; Stolfo et al., 2023).

Our work is motivated by the findings of Skean et al. (2024), who found that intermediate representations often retain more comprehensive information compared to final layer representations. Drawing from this observation, we utilize Singular Value Decomposition (SVD) to build a novel analytical framework, shifting the focus from hidden representations to the analysis of weight matrices. This approach offers a more generalized and interpretable perspective for understanding model behavior.

3 Analysis of LLM Reasoning Mechanisms

We systematically analyze how LLMs process information across layers, revealing distinct reasoning phases through singular value decomposition and empirical validation.

3.1 Transformer-based Language Models

Transformer-based language models consist of multiple transformer blocks, each comprising an attention layer (*Attn*) and a Multi-Layer Perceptron (MLP), connected via a residual stream (Vaswani et al., 2017).

The attention layer utilizes multiple attention heads, each performing scaled dot-product attention:

$$\text{Attn}(\boldsymbol{X}) = \text{softmax}\left(\frac{\boldsymbol{W}_q\boldsymbol{x} \cdot (\boldsymbol{W}_k\boldsymbol{x})^T}{\sqrt{d_k}}\right)\boldsymbol{W}_v\boldsymbol{x},$$

where d_k represents the dimensionality of the key vectors, $\boldsymbol{x}$ denotes the input vector, and $\boldsymbol{W}_q$, $\boldsymbol{W}_k$, and $\boldsymbol{W}_v$ are the weight matrices for the queries, keys, and values, respectively. The softmax operation generates an attention weight distribution over the input sequence, which indicates the relevance of each token to the current position. The value $\boldsymbol{V} = \boldsymbol{W}_v\boldsymbol{x}$, encapsulating the informational content, is weighted by these attention probabilities, enabling the dynamic aggregation of contextual information across the sequence. Consequently, the output of the attention block is essentially a weighted sum of $\boldsymbol{V}$, with the weights computed based on the interaction between $\boldsymbol{Q}$ and $\boldsymbol{K}$. Therefore, $\boldsymbol{W}_v$ plays a pivotal role in facilitating the flow of information within the attention mechanism, and analyzing it provides valuable insights into the processing and propagation of information within the framework. This analysis is further explored using Singular Value Decomposition (SVD).

3.2 Transformation Divergence

The linear transformation $\boldsymbol{W}_v\boldsymbol{x}$ can be interpreted as a combination of rotation and scaling operations applied to the vector $\boldsymbol{x}$, and these operations are elegantly captured through Singular Value Decomposition (SVD). SVD decomposes a matrix into three simpler matrices, providing a geometric inter-

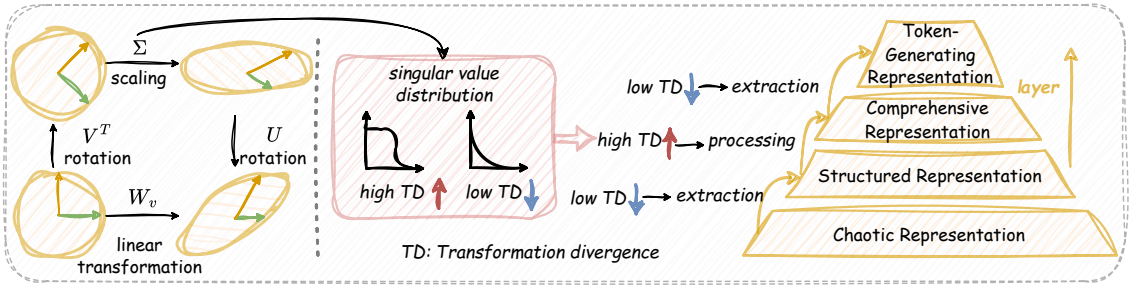


Figure 3: Weight matrix transformation divergence (TD) and information processing in LLM reasoning. The left panel illustrates the SVD, where the weight matrix is decomposed into three operators, including scaling. The singular value distribution represents the scaling magnitude of the weight matrix, which is quantified using TD. The right panel shows how the value of TD reflects the information processing stages in LLM reasoning.

pretation of the transformation represented by the original matrix.

For a matrix $\boldsymbol{W}_v$ of size $m \times n$, SVD expresses it as:

$$\begin{aligned} \boldsymbol{W}_v &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \\ &= \boldsymbol{u}_1\sigma_1\boldsymbol{v}_1^T + \cdots + \boldsymbol{u}_r\sigma_r\boldsymbol{v}_r^T \\ &= \sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^T + \cdots + \sigma_r\boldsymbol{u}_r\boldsymbol{v}_r^T, \end{aligned} \tag{1}$$

where r denotes the rank of $\boldsymbol{W}_v$, and $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ is a diagonal matrix of size $m \times n$ containing the singular values of $\boldsymbol{W}_v$, which are non-negative and sorted in descending order, while matrices $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m]$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$ are orthogonal matrices of sizes $m \times m$ and $n \times n$, respectively, with $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ representing the left and right singular vectors.

This decomposition provides a geometric interpretation of the transformation represented by $\boldsymbol{W}_v$. The matrix $\boldsymbol{V}$ applies a rotation in the input space, while $\boldsymbol{\Sigma}$ scales the input vectors by the singular values. Finally, $\boldsymbol{U}$ performs a rotation in the output space, as illustrated in Figure 3. In essence, the linear transformation can be decomposed into two components: rotation, which maps vectors to other space, and scaling, which stretches or shrinks the roation operations. The transformation directions are defined by the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$.

Definition 3.1 (Transformation Direction) *The transformation direction of a transformer matrix $\boldsymbol{W}_v$ is formally defined as:*

$$\text{Direction}(\boldsymbol{W}_v) = [\boldsymbol{u}_1\boldsymbol{v}_1^T, \ldots, \boldsymbol{u}_r\boldsymbol{v}_r^T]$$

The linear transformation $\boldsymbol{W}_v\boldsymbol{x}$ can be expressed in terms of these transformation directions, since $\boldsymbol{W}_v\boldsymbol{x} = \sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^T\boldsymbol{x} + \cdots + \sigma_r\boldsymbol{u}_r\boldsymbol{v}_r^T\boldsymbol{x}$ according to Equation 1, where σ is the scaling value that emphasizes the magnitude of the transformation along each direction.

Scaling operations are mathematically characterized by the singular values σ, which represent the relative magnitudes of the orthogonal transformation directions in the matrix decomposition. When the singular values are similar, it indicates that the transformation processes information across multiple directions. In contrast, large discrepancies between singular values suggest a more selective extraction of information. Therefore, singular values can be used to determine whether the transformation focuses on information extraction or processes information across different directions.

Definition 3.2 (Transformation Divergence) *The transformation divergence of any non-zero matrix (transformation) $\boldsymbol{W}_v \in \mathbb{R}^{m\times n}$ is defined as:*

$$\text{TD}(\boldsymbol{W}_v) = -\sum_{i=1}^{r} \frac{\sigma_i}{\sum_{j=1}^{r}\sigma_j} \log \frac{\sigma_i}{\sum_{j=1}^{r}\sigma_j}, \tag{2}$$

where $r = \min(m, n)$, and $\sigma_1, \sigma_2, \ldots, \sigma_r$ are the singular values of matrix $\boldsymbol{W}_v$.

Transformation divergence, as defined above, quantifies the extent of information processing and extraction within a linear transformation represented by a matrix. It focuses on relative magnitudes rather than absolute ones, specifically measuring the disparity in the scaling of different directions. A higher transformation divergence, such as for the vector $(5, 4, 3)^T$, with a divergence value of 1.08, indicates a transformation that processes information across multiple directions. In this case, the non-zero values suggest that all directions will be retained, though some directions are weighted more heavily than others. In contrast, a lower

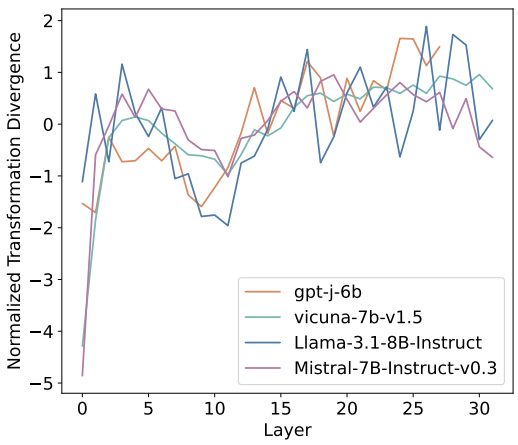


Figure 4: Transformation Divergence $\mathrm{TD}(\boldsymbol{W}_v)$ of the weight matrix across various models.

transformation divergence signifies a more concentrated focus on a particular direction. For instance, for the vector $(1, 0, 0)^T$, the transformation divergence indicates that the transformation primarily extracts information from a single dominant direction, discarding information from other directions. This metric allows for a layer-wise analysis of the model's behavior, as illustrated in Figure 3.

3.3 Layer-wise Dynamics of Transformation Divergence

To show how different layers work in LLM using transformation divergence, we applied Equation 3.2 to quantify this metric across diverse open-source LLMs with varying architectures. These models exhibit fundamental structural differences, rather than differing in parametric scale, as detailed in Appendix B.

As illustrated in Figure 4, transformation divergence $\mathrm{TD}(\boldsymbol{W}_v)$ exhibits a consistent three-phase progression across all analyzed models: an initial high-value phase, followed by a period of progressive decrease, and a final increase. As discussed in Section 3.2, low $\mathrm{TD}(\boldsymbol{W}_v)$ corresponds to a concentrated extraction of information along principal component directions, while high $\mathrm{TD}(\boldsymbol{W}_v)$ indicates broad manipulation across multiple transformation directions. Based on this, we hypothesize that LLMs undergo three distinct reasoning operations, leading to four representational states.

These three operations reflect fundamental information processing mechanisms within transformer architectures. During the initial high $\mathrm{TD}(\boldsymbol{W}_v)$ phase, LLMs extract information from disorganized data. The subsequent low $\mathrm{TD}(\boldsymbol{W}_v)$ phase signifies the processing of this information. Finally, the resurgent high $\mathrm{TD}(\boldsymbol{W}_v)$ phase aligns the representation with the token prediction requirements which is also proved in Fan et al. (2024) and Haviv et al. (2023). Based on these operations, we can categorize the representations during inference into four types:

- **Chaotic Representation**: The initial representation containing raw input features.
- **Structured Representation**: The structured information extracted from the chaotic representation.
- **Comprehensive Representation**: A multi-dimensional embedding that integrates structured information, contextual dependencies, and domain-specific knowledge for a holistic understanding.
- **Token-Generating Representation**: The final output specialization aligning for token prediction.

Consequently, the structured and comprehensive representations may be particularly effective for retrieving relevant next-hop documents.

To validate this hypothesis, we evaluate the relationship between transformation divergence and information processing in LLMs by computing the recall rate for documents indirectly related to the query using different layer representations in the L-RAG framework, as described in Section 4. These results are then compared with the transformation divergence patterns. As shown in Appendix D, the best recall performance is observed at the layer during or after the low transformation divergence period, supporting the hypothesis outlined above.

3.4 LogitLens Across LLMs

To validate the transformation divergence pattern, we apply the LogitLens technique to track the evolution of information across transformer layers. Our hypothesis suggests that, during layer-by-layer reasoning, LLMs extract and process information progressively. This process likely generates intermediate information, which would result in intermediate answers emerging earlier than the final answers.

We analyze individual hidden states using the LogitLens technique (Haviv et al., 2023), where

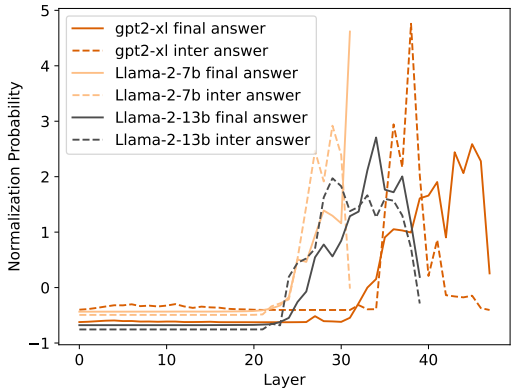


Figure 5: Logit Lens analysis conducted at multiple layers within different LLMs. The dashed lines represent the probability of intermediate answers, while the solid lines indicate the probability of final answers.

activations are transformed into logits for each vocabulary token by multiplying them with the output embedding matrix, as detailed in Appendix A.

As shown in Figure 5, intermediate answers emerge earlier than final answers across various LLMs. This suggests that the information leading to the intermediate answer is processed earlier in the reasoning process. Therefore, identifying the layer with the most relevant information for the intermediate answer is crucial for retrieving the next-hop document, rather than relying on the final layer.

4 Method

We present Layer-wise Retrieval-Augmented Generation (L-RAG), an innovative framework that enhances retrieval-augmented LLMs by leveraging intermediate representations to improve the efficiency of multi-hop document retrieval, particularly at higher hops. In Section 3.3, we discuss how LLMs extract valuable information from chaotic representations, process this information into comprehensive representations, and ultimately derive token-generation information for producing token-generating representations. Based on this understanding, we proposed an overview of the L-RAG framework as shown in Figure 2. This section outlines the pipeline structure in Section 4.1, followed by the detailed implementation procedures in Section 4.2.

4.1 Overview of L-RAG

L-RAG consists of three main components. Initially, the traditional retriever (such as BM25 or Contriever) retrieves the first-hop documents directly related to the query. These documents are then combined with the query to form an input instruction. LLM perform single token-level reasoning to generate an intermediate representation. Representation retriever, designed to effectively utilize the nuanced information embedded in these intermediate representations, then retrieves next-hop documents that are more contextually relevant.

Figure 2 illustrates the L-RAG framework, detailing the sequential steps from query input to final answer generation. By integrating these components, L-RAG aims to improve the overall retrieval efficiency and answer quality in retrieval-augmented LLMs.

4.2 Implementation Details

In the L-RAG framework, only the representation retriever requires training. We utilize off-the-shelf traditional retrievers and LLMs. For representation retriever, we add a MLP layer in front of the original Contriever to align LLM intermediate representations with the Contriever embedding space. The relevance score between an intermediate representation and a document is calculated as the dot product of their respective representations after processing through the Contriever. Formally, given an intermediate representation r and a document d, both are encoded independently using the same model f. Additionally, the intermediate representation r is passed through an MLP g. The relevance score $s(r, d)$ is then defined as:

$$s(r, d) = f(g(r))^T \cdot f(d) \tag{3}$$

For training, we employ contrastive InfoNCE loss to enable the model to distinguish between documents by comparing positive pairs (related documents) and negative pairs (unrelated documents) using intermediate representations. Formally, given a batch of intermediate representations $R = [r_1, r_2, \ldots, r_n]$ and a pool of associated positive documents $D = [d_1, d_2, \ldots, d_m]$, the contrastive loss for one representation and its corresponding positive document is defined as:

$$\hat{\mathcal{L}}(r_i, d_i^+) = \frac{\exp(s(r_i, d^+)/\tau)}{\sum_{d \in D} \exp(s(r_i, d)/\tau)}, \tag{4}$$

where s represents the similarity score, as defined in Equation 3, and τ is the temperature parameter.

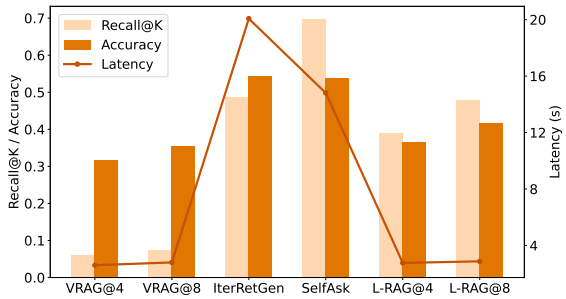


Figure 6: The recall rate and accuracy of different methods in the 2WikiMQA dataset. VRAG stands for Vanilla RAG.

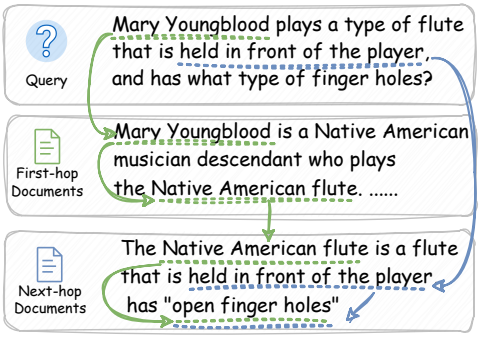


Figure 7: Information Leakage in Multi-Hop Reasoning on HotpotQA. This diagram contrasts standard reasoning pathways (green arrows) with shortcut retrieval patterns (blue arrows), demonstrating how query leverage surface-level positional descriptors (e.g., 'held in front of the player') to directly match target documents while bypassing critical intermediate reasoning about cultural attributes of Native American flutes.

The overall contrastive InfoNCE loss for the set of intermediate representations R is computed as the average loss across all representations $r \in R$. It is formally defined as:

$$\mathcal{L}(R, D) = -\log \sum_{r_i \in R} \sum_{d_i^+} \hat{\mathcal{L}}(r_i, d_i^+), \quad (5)$$

where d_i^+ denotes the positive document for representation r_i in D, and $\hat{\mathcal{L}}(r_i, d_i^+)$ is the contrastive InfoNCE loss defined in Equation 4.

5 Experiment

Method	2Wiki	Musique	Hotpot
No Retrieval	2.29	2.30	2.28
VanillaRAG@4	2.60	2.55	2.64
VanillaRAG@8	2.79	2.67	2.73
SelfAsk	14.80	19.62	19.59
IterRetGen	20.07	21.18	42.36
L-RAG@4	2.76	2.62	2.80
L-RAG@8	2.87	2.86	2.92

Table 1: Task latency per query across different datasets for the Llama3.1-8B model.

5.1 Setup

Datasets. We evaluate our method on three multi-hop question answering (QA) benchmark, which require sequential reasoning across multiple documents: MuSiQue (Trivedi et al., 2022) under CC-BY 4.0 license, HotpotQA (Yang et al., 2018) under a CC BY-SA 4.0 license, and 2WikiMultiHopQA (Ho et al., 2020) under Apache license 2.0 . Specifically, we focus on queries that require two-hop reasoning to arrive at the correct answer, explicitly excluding those solvable through single-hop inference, as detailed in Appendix C. Additionally, we use GPT-4 to identify higher-hop documents that cannot be directly recognized within the dataset.

Baselines. Our baseline models encompass: (1) straightforward methods, specifically **No Retrieval**, **Vanilla RAG**, and **SelfRAG** (Asai et al., 2024); (2) advanced multi-step retrieval strategies, namely **SelfAsk** (Press et al., 2023a) and **IterRetGen** (Shao et al., 2023a). All models are executed on the LLama3.1-8b platform.

Metrics. To evaluate L-RAG, we assess three aspects: retrieval performance, task performance, and task efficiency. For retrieval performance, we report the higher-hop document recall rate. Task performance is measured by accuracy (Acc), in line with standard evaluation protocols. Accuracy is determined by whether the predicted answer contains the ground-truth answer. Regarding efficiency, we report the average time required to answer each query.

5.2 Task Efficiency

We calculate the latency of each query during the execution of those method. As shown in Table 1 and Figure 6, our method requires low latency, comparable to Vanilla-RAG and direct reasoning, while delivering significantly better performance.

Method	2WikiMQA Recall@K	K	Musique Recall@K	K	HotpotQA Recall@K	K
Vanilla RAG@4	0.061	4.0	0.695	4.0	0.543	4.0
Vanilla RAG@8	0.074	8.0	0.762	8.0	0.641	8.0
IterRetGen	0.487	6.0	0.523	6.0	**0.732**	6.0
SelfAsk	**0.697**	27.4	**0.765**	28.1	0.711	28.6
L-RAG@4	0.389	4.0	0.520	4.0	0.449	4.0
L-RAG@8	0.480	8.0	0.764	8.0	0.613	8.0

Table 2: Results on higher-hop document recall rate across three dataset. Baselines are implemented from the source code. Bold fonts denote the best results in the same generator.

5.3 Retrieval Performance

We evaluate the model's performance on next-hop document retrieval using the Recall@K metric across three datasets. As shown in Table 2, L-RAG demonstrates moderate yet acceptable recall scores. Notably, it significantly outperforms Vanilla RAG on the 2WikiMQA dataset and achieves near-best performance on the Musique dataset.

The recall metric for the vanilla RAG method quantifies information leakage in higher-hop document retrieval, reflecting the likelihood of retrieving higher-hop documents using the initial query. As shown in Table 2, 2WikiMQA dataset exhibits the lowest direct recall score (0.061), while other datasets achieve recall rates above 0.5. This significant gap suggests that 2WikiMQA contains less inherent information leakage between reasoning hops compared to the other benchmarks. As Figure 7 shown, while the question structure ostensibly requires intermediate reasoning about the "Native American flute" concept, the presence of the distinctive positional descriptor "held in front of the player" creates an unintended retrieval shortcut. Therefore, 2WikiMQA serves as a more rigorous and reliable benchmark for evaluating multi-hop reasoning, as its structure reduces shortcut solutions through direct query matching, requiring genuine multi-step inference.

5.4 Task Performance

As shown in Table 3, the L-RAG framework demonstrates superior performance across benchmark datasets, achieving the highest accuracy (0.352) on MuSiQue, securing a competitive second-place ranking (0.641) on HotpotQA, and attaining third place (0.418) on the complex 2WikiMQA corpus. While L-RAG may not achieve the best performance across all datasets, its task latency, as illustrated in Figure 6, indicates that its performance is sufficiently strong for practical real-world applications.

Dataset	2Wiki	Musique	Hotpot
No Retrieval	0.275	0.094	0.230
VanillaRAG@4	0.316	0.305	0.590
VanillaRAG@8	0.355	0.297	0.598
SelfAsk	0.539	0.180	0.373
IterRetGen	**0.543**	0.297	**0.668**
L-RAG@4	0.367	0.258	0.555
L-RAG@8	0.418	**0.352**	0.641

Table 3: Task performance (Acc) across different method and three dataset.Bold fonts denote the best results in the same generator.

6 Conclusion

This work has three principal contributions to enhance multi-hop reasoning in Retrieval-Augmented Generation (RAG) systems. First, we identify a three-stage computational pattern in LLM inference through singular value decomposition (SVD) of weight matrices and LogitLens analysis. Second, building on this insight, we propose Layer-wise RAG (L-RAG), a novel framework that leverages intermediate layer representations for document retrieval. By training a Contriever-based retriever to interpret these representations, our method bypasses iterative LLM calls while capturing critical multi-hop reasoning signals. Third, comprehensive evaluations across multiple LLMs demonstrate that L-RAG achieves competitive accuracy with multi-step RAG methods, yet maintains computational efficiency comparable to vanilla RAG. Our findings bridge the gap between reasoning capability and inference overhead, offering a practical solution for knowledge-intensive applications.

Limitations

In this paper, we conduct the first systematic investigation into analyzing the reasoning patterns of LLMs through SVD, leveraging these findings to motivate an enhanced RAG framework. However, given that reasoning pattern characterization requires broader empirical support beyond current validation scope, our experimental analysis currently focuses on: (a) RAG system verification through external knowledge grounding, and (b) internal answer coherence validation within the LLM's parametric knowledge base.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. 2024. Hopping too late: Exploring the limitations of large language models on multi-hop queries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14113–14130, Miami, Florida, USA. Association for Computational Linguistics.

Jannik Brinkmann, Abhay Sheshadri, Victor Levoso, Paul Swoboda, and Christian Bartelt. 2024. A mechanistic analysis of a transformer trained on a symbolic multi-step reasoning task. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4082–4102, Bangkok, Thailand. Association for Computational Linguistics.

Grégoire Delétang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A. Ortega. 2023. Neural networks and the chomsky hierarchy. *Preprint*, arXiv:2207.02098.

Siqi Fan, Xin Jiang, Xiang Li, Xuying Meng, Peng Han, Shuo Shang, Aixin Sun, Yequan Wang, and Zhongyuan Wang. 2024. Not all layers of llms are necessary during inference. *Preprint*, arXiv:2403.02181.

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

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235.

Adi Haviv, Ido Cohen, Jacob Gidron, Roei Schuster, Yoav Goldberg, and Mor Geva. 2023. Understanding transformer memorization recall through idioms. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 248–264, Dubrovnik, Croatia. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050, Mexico City, Mexico. Association for Computational Linguistics.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

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

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023a. Measuring and

narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023b. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

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

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

Oscar Skean, Md Rifat Arefin, and Ravid Shwartz-Ziv. 2024. Does representation matter? exploring intermediate layers in large language models. In *Workshop on Machine Learning and Compression, NeurIPS 2024*.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 2023. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7035–7052, Singapore. Association for Computational Linguistics.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. 2024. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization. *Preprint*, arXiv:2405.15071.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Dylan Zhang, Curt Tigges, Zory Zhang, Stella Biderman, Maxim Raginsky, and Talia Ringer. 2024. Transformer-based models are not yet perfect at learning to emulate structural recursion. *Preprint*, arXiv:2401.12947.

Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2024. EfficientRAG: Efficient retriever for multi-hop question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3392–3411, Miami, Florida, USA. Association for Computational Linguistics.

A Logit Lens

LogitLens is an interpretability framework for analyzing language model hidden states by examining logits (raw prediction scores) and their corresponding probability distributions. Specifically, for a given hidden state $\boldsymbol{h}_l$ at the $\boldsymbol{l}^{th}$ layer, the logits $\boldsymbol{s}_l$ and probabilities $\boldsymbol{p}_l$ over the output vocabulary set V are defined as:

$$\begin{cases} \boldsymbol{s}_l = \boldsymbol{W}_U \boldsymbol{h}_l^i \in \mathbb{R}^{|V|}, \\ \boldsymbol{p}_l = \text{softmax}\left(\boldsymbol{s}_l\right) \end{cases}, \tag{6}$$

where $\boldsymbol{W}_U$ denotes the unembedding matrix, which is the same matrix used in the final layer of the model for prediction.

Our evaluation spans diverse LLM architectures, including `GPT-2-XL`, `Llama-2-7B`, and `Llama-2-13B`. We manually annotated intermediate answers for the 2WikiMultiHopQA dataset to establish ground truth for multi-hop reasoning steps.

B Transformation Divergence in LLMs of Varying Scales

We compute the transformation divergence for various scaling `GPT` models, including `GPT-2` (124M), `GPT-2-XL` (1.5B), and `GPT-J-6B`, as illustrated in Figure 8, as well as for `Llama-2-7B-chat-hf`, `Llama-2-13B-chat-hf`, and `Llama-3.1-8B-Instruct`, as shown in Figure 9.

C Multi-Hop Dataset Example

The manually annotated dataset is derived from 2WikiMultiHopQA, with examples including intermediate answers, Here is an example:

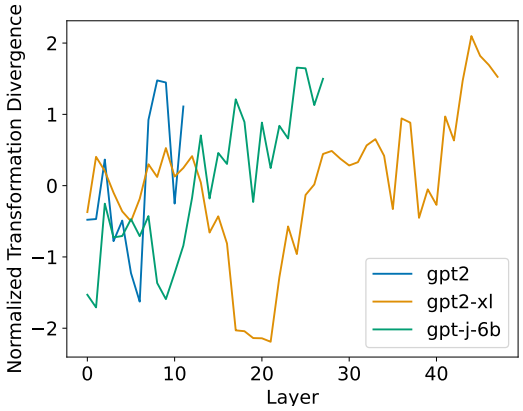


Figure 8: Transformation Divergence in gpt of Varying Scales

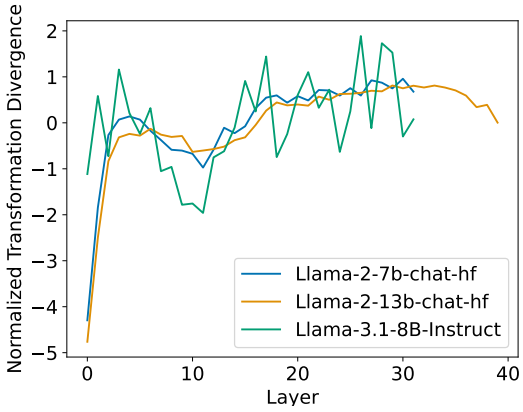


Figure 9: Transformation Divergence in Llama of Varying Scales

Question What is the date of birth of Mina Gerhardsen's father?

Document

1. Mina Gerhardsen (born 14 September 1975) is a Norwegian politician for the Labour Party. She is the daughter of Rune Gerhardsen and Tove Strand, and granddaughter of Einar Gerhardsen. She is married to Eirik Øwre Thorshaug. She led the Oslo branch of Natur og Ungdom from 1993 to 1995, and was deputy leader of the Workers' Youth League in Oslo in 1997. She took the cand.mag. degree at the University of Oslo in 1998, and also has master's degrees in pedagogy from 2000 and human geography from 2003. From 1999 to 2002 she worked part-time as a journalist in Dagsavisenänd Dagbladet:. She then worked in the Norwegian Red Cross from 2002 to 2004, except for a period from 2003 to 2004 as a journalist in Mandag Morgen:. She was hired as a political advisor in the Norwegian Office of the Prime Minister in 2005, when Stoltenberg's Second Cabinet assumed office. In 2009 she was promoted to State Secretary. In 2011 she changed to the Ministry of Culture.
2. Rune Gerhardsen (born 13 June 1946) is a Norwegian politician, representing the Norwegian Labour Party. He is a son of Werna and Einar Gerhardsen, and attended Oslo Cathedral School. He chaired the Workers' Youth League from 1973 to 1975 and chaired the City Government of Oslo from 1992 to 1997. He chaired the Norwegian Skating Association from 1986 to 1990 and 2001 to 2003 and also 2013 to 2017.

Intermediate Answer Rune Gerhardsen

Final Answer 13 June 1946

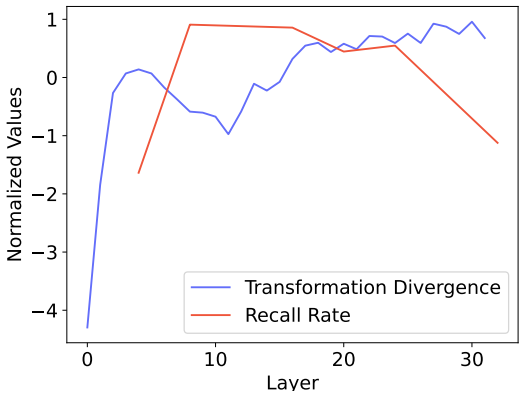


Figure 10: Transformation Divergence across the llama2-7b model and the corresponding representation generated for retriever performance.

D Transformation Divergence and Recall Rate

To evaluate retriever performance with different layer representations, we utilize `meta-llama/Llama-2-7b-chat-hf` to generate representations in layers 4, 8, 16, 20, 24, 32 and compute the recall rate for the documents indirectly related to the query in 2WikiMultiHopQA dataset which result is shown in Figure 10.