

标题

INFERENCE SCALING FOR LONG-CONTEXT RETRIEVAL AUGMENTED GENERATION

Zhenrui Yue^{*†1}, Honglei Zhuang^{*2}, Aijun Bai², Kai Hui², Rolf Jagerman², Hansi Zeng^{†3}
Zhen Qin², Dong Wang¹, Xuanhui Wang², Michael Bendersky²

¹University of Illinois Urbana-Champaign, ²Google DeepMind, ³UMass Amherst
zhenrui3@illinois.edu, hlz@google.com

<https://openreview.net/forum?id=FSjlrOm1vz>

摘要

The scaling of inference computation has unlocked the potential of long-context large language models (LLMs) across diverse settings. For knowledge-intensive tasks, the increased compute is often allocated to incorporate more external knowledge. However, without effectively utilizing such knowledge, solely expanding context does not always enhance performance. In this work, we investigate inference scaling for retrieval augmented generation (RAG), exploring the combination of multiple strategies beyond simply increasing the quantity of knowledge, including in-context learning and iterative prompting. These strategies provide additional flexibility to scale test-time computation (e.g., by increasing retrieved documents or generation steps), thereby enhancing LLMs' ability to effectively acquire and utilize contextual information. We address two key questions: (1) How does RAG performance benefit from the scaling of inference computation when optimally configured? (2) Can we predict the optimal test-time compute allocation for a given budget by modeling the relationship between RAG performance and inference parameters? Our observations reveal that increasing inference computation leads to nearly linear gains in RAG performance when optimally allocated, a relationship we describe as the inference scaling laws for RAG. Building on this, we further develop the computation allocation model to estimate RAG performance across different inference configurations. The model predicts optimal inference parameters under various computation constraints, which align closely with the experimental results. By applying these optimal configurations, we demonstrate that scaling inference compute on long-context LLMs achieves up to 58.9% gains on benchmark datasets compared to standard RAG.

背景与动机

论文提出的背景

长上下文大型语言模型在各种下游任务中的性能表现不断提高，且能够处理很长的上下文（如 Gemini 1.5 Pro 支持 2M token），用于处理更长的输入序列和更多外部知识，而在 RAG 系统中将检索到的文档数量或大小增加到某个阈值前可以持续提高性能。

问题领域

文章聚焦于“检索增强生成”（Retrieval Augmented Generation, RAG）在长上下文场景下的推理扩展，研究了在资源有限的场景下如何最优地分配计算预算（而不是单纯增加知识量）以提升模型性能。

前人方法的缺点

现有对 RAG 的推理规模化仅仅关注增加检索文档的数量或长度，大多都忽视了知识的利用效率，但当文档数超过某个阈值（如 top-10 文档）后，性能会出现瓶颈期甚至下降，原因在于未经筛选的大量文档容易引入噪声，且长上下文 LLM 对超长输入的关键信息定位能力比较有限。

作者的切入角度与解决目标

检索的计算成本通过比 LLM 推理要低得多，作者提出两种新的 RAG 策略——**DRAG**（Demonstration-based RAG）和**IterDRAG**（Iterative Demonstration-based RAG），通过上下文学习和迭代提示分解复杂查询并交叉回答，期望解决以下目标：

- 建立 RAG 性能、不同规模和推理计算分配之间的关系
- 进行推理计算分配方面的 RAG 性能建模，预测最佳的推理参数集，以最大限度地提高不同 RAG 任务的性能

方法

文章自己的总结就挺好，下图中包含了文章中提出的两种 RAG 的工作流程及对 IterDRAG 的解释。

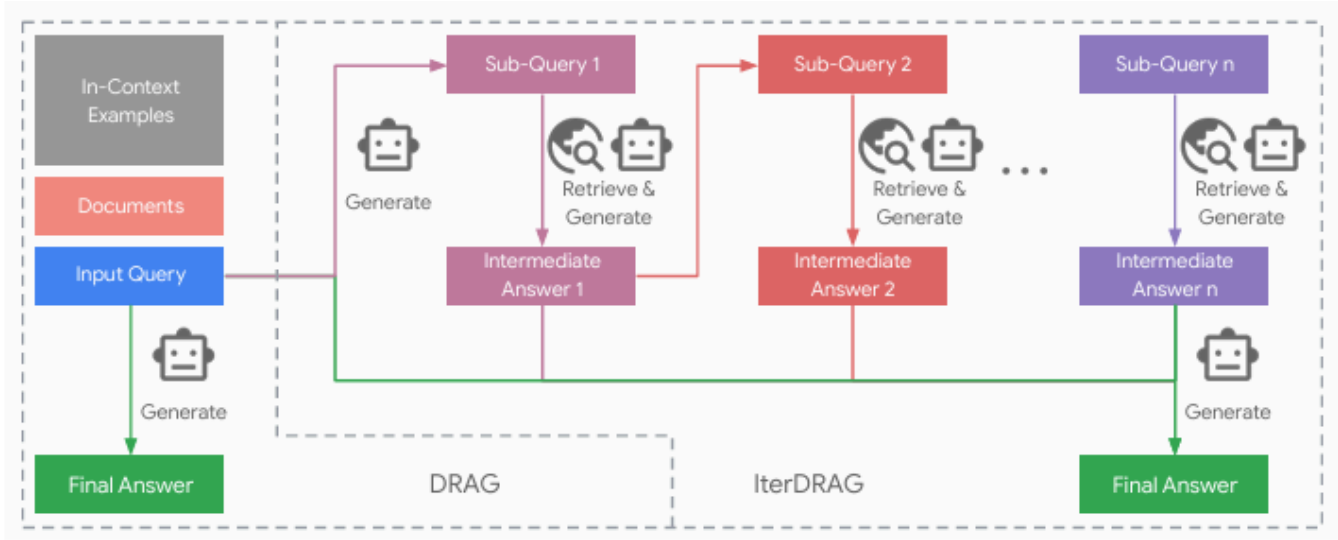


Figure 3: DRAG vs. IterDRAG. IterDRAG breaks down the input query into sub-queries and answer them to improve the accuracy of the final answer. In test-time, IterDRAG scales the computation through multiple inference steps to decompose complex queries and retrieve documents.

DRAG：基于演示的 RAG

在一个推理请求中，DRAG 在 prompt 中按顺序同时加入：

- m 个示例
 - 每个示例包含检索文档、输入及其答案
- k 篇检索到的文档
 - 文章中特别指出此处需要反序排列
 - 排名较高的文档需要更靠近查询
- 待回答的查询

也就是上图中的左侧内容，这样可以让长上下文 LLM 学会在示例中“提取信息 → 生成答案”。

IterDRAG：迭代的 DRAG

Despite access to external knowledge, complex multi-hop queries remain challenging due to the compositionality gap.

为解决上面的问题，IterDRAG 首先将原始查询分解为若干子查询，针对每个子查询交替执行：

1. 检索文档
2. 生成“中间答案”

直至所有子查询解决后，汇总文档、子查询及其中间答案生成最终答案。该过程最多被允许迭代 5 步，有效上下文长度为所有迭代输入之和，正如图右侧所示。

As such, the iterative retrieval and generation strategy helps narrowing the compositionality gap and improves knowledge extraction, thereby enhancing overall RAG performance.

推理规模化规律与计算分配模型

最佳性能定义如下图左，下图右可看到在最优配置下， $P^*(L_{\max})$ 关于 $\log L_{\max}$ 近似线性增长。

For each input query and its ground-truth answer $(x_i, y_i) \in \mathcal{X}$, we can apply the RAG inference strategy f parameterized by θ . We denote the effective input context length to the LLM as $l(x_i; \theta)$ and the obtained prediction as $\hat{y}_i = f(x_i; \theta)$. A metric $P(y_i, \hat{y}_i)$ can then be calculated based on y_i and \hat{y}_i . To understand the relationship between RAG performance and inference computation, we sample a few different inference computation budgets. For each budget L_{\max} , we find the optimal average metric $P^*(L_{\max})$ achievable within this budget by enumerating different $\theta \in \Theta$:

$$P^*(L_{\max}) := \max_{\theta \in \Theta} \left\{ \frac{1}{|\mathcal{X}|} \sum_i P(y_i, f(x_i; \theta)) \mid \forall i, l(x_i; \theta) \leq L_{\max} \right\}. \quad (1)$$

若定义任务特征向量 $i = (i_{\text{doc}}, i_{\text{shot}}, 0)$ ，其中两者分别测量文档和上下文示例的信息量，则有

$$\sigma^{-1}(P(\theta)) \approx (a + b \odot i)^\top \log(\theta) + c$$

其中参数 a, b, c 仅依赖于模型，可一次性估计，用于预测不同任务和预算下的最优 θ 。

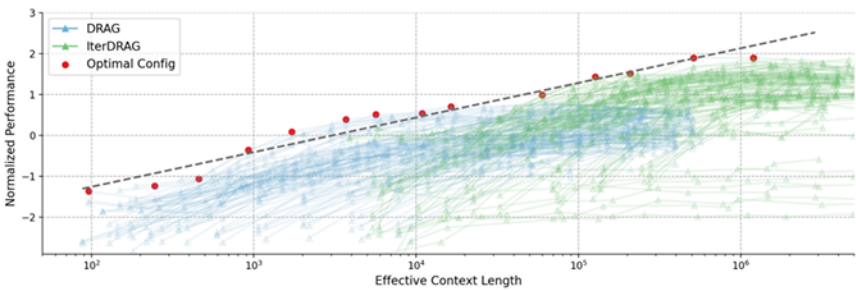


Figure 4: Normalized performance vs. effective context lengths across datasets. Each line represents a fixed configuration, scaled by varying the number of documents. Red dots indicate the optimal configurations, with the dashed line showing the fitting results. The observed optimal performance can be approximated by a linear relationship with the effective context lengths.

实验结果（性能分析）

- 数据集与评测：在 Bamboogle、HotpotQA、MuSiQue、2WikiMultiHopQA 四个多跳问答数据集上，用 Gemini 1.5 Flash（窗口上限 1M 令牌），评测指标包含 Exact Match、F1 和 Accuracy，每个数据集抽样 1.2k 样本。
- 整体性能
 - 在 16k-128k 预算下，DRAG 较传统 RAG（仅检索）和两种 QA（zero-shot、many-shot）表现更优，性能持续上升；
 - 在 ≥ 128k 时，IterDRAG 进一步超越 DRAG
 - 并且可以看到文章中提出的两种 RAG 性能始终优于基线

Table 1: Optimal performance of different methods with varying maximum effective context lengths L_{max} (i.e., the total number of input tokens across all iterations). ZS QA and MS QA refers to zero-shot QA and many-shot QA respectively. Partial results are omitted for methods that do not further scale with increasing L_{max} . For clarity, we mark the best results for each L_{max} in bold.

L_{max}	Method	Bamboogle			HotpotQA			MuSiQue			2WikiMultiHopQA		
		EM	F1	Acc	EM	F1	Acc	EM	F1	Acc	EM	F1	Acc
16k	ZS QA	16.8	25.9	19.2	22.7	32.0	25.2	5.0	13.2	6.6	28.3	33.5	30.7
	MS QA	24.0	30.7	24.8	24.6	34.0	26.2	7.4	16.4	8.5	33.2	37.5	34.3
	RAG	44.0	54.5	45.6	44.2	57.9	49.2	12.3	21.5	15.3	42.3	49.3	46.5
	DRAG	44.0	55.2	45.6	45.5	58.5	50.2	14.5	24.6	16.9	45.2	53.5	50.5
	IterDRAG	46.4	56.2	51.2	36.0	47.4	44.4	8.1	17.5	12.2	33.2	38.8	43.8
32k	RAG	48.8	56.2	49.6	44.2	58.2	49.3	12.3	21.5	15.3	42.9	50.6	48.0
	DRAG	48.8	59.2	50.4	46.9	60.3	52.0	15.4	26.0	17.3	45.9	53.7	51.4
	IterDRAG	46.4	56.2	52.0	38.3	49.8	44.4	12.5	23.1	19.7	44.3	54.6	56.8
128k	RAG	51.2	60.3	52.8	45.7	59.6	50.9	14.0	23.7	16.8	43.1	50.7	48.4
	DRAG	52.8	62.3	54.4	47.4	61.3	52.2	15.4	26.0	17.9	47.5	55.3	53.1
	IterDRAG	63.2	74.8	68.8	44.8	59.4	52.8	17.3	28.0	24.5	62.3	73.8	74.6
1M	DRAG	56.0	62.9	57.6	47.4	61.3	52.2	15.9	26.0	18.2	48.2	55.7	53.3
	IterDRAG	65.6	75.6	68.8	48.7	63.3	55.3	22.2	34.3	30.5	65.7	75.2	76.4
5M	IterDRAG	65.6	75.6	68.8	51.7	64.4	56.4	22.5	35.0	30.5	67.0	75.2	76.9

线性规模化

各种最优配置下的归一化性能与有效上下文长度在 log 轴上近似直线，且 RAG 性能随文档和上下文示例数量的变化而变化

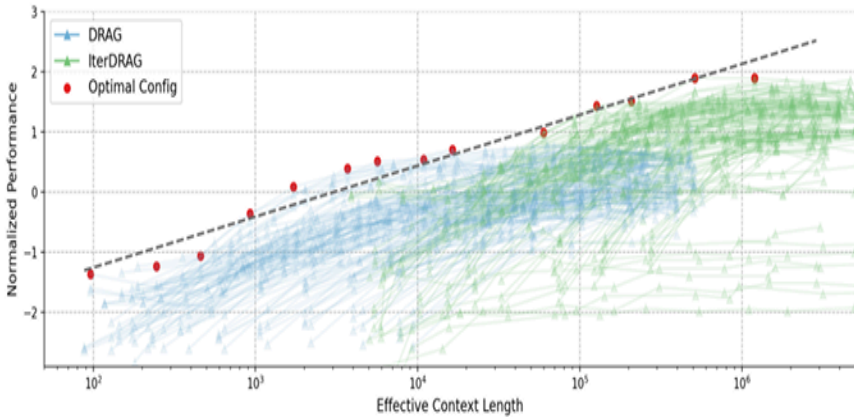


Figure 4: Normalized performance vs. effective context lengths across datasets. Each line represents a fixed configuration, scaled by varying the number of documents. Red dots indicate the optimal configurations, with the dashed line showing the fitting results. The observed optimal performance can be approximated by a linear relationship with the effective context lengths.

泛化能力

计算分配模型在多领域通用性与长度外推上均表现良好，预测性能与 Oracle 差距 < 5%

Table 3: Domain generalization results of the computation allocation model for RAG.

	Bamboogle			HotpotQA			MuSiQue			2WikiMultiHopQA		
	EM	F1	Acc	EM	F1	Acc	EM	F1	Acc	EM	F1	Acc
Baseline	49.6	58.8	51.2	46.3	60.2	51.4	14.9	24.7	16.9	46.5	53.7	51.6
Predict	64.0	75.6	68.0	47.8	63.3	55.3	19.3	32.5	29.3	60.8	72.4	74.9
Oracle	65.6	75.6	68.8	48.7	63.3	55.3	22.2	34.3	30.5	65.7	75.2	76.4

Domain Generalization. We also examine the generalization of the computation allocation model for RAG for unseen domains. In other words, the parameters of Equation (2) are tested on the target domain but learnt from the remaining domains. For inference, only i is derived from the target domain. We report the results for 1M effective context length in Table 3, where we compare to an 8-shot baseline configuration (scaled by increasing retrieved documents) and the optimum results (Oracle). In summary, the results show that computation allocation model significantly outperforms baseline and closely aligns with the oracle results (96.6% of the optimal performance). Notably, Bamboogle and HotpotQA exhibit highly similar target results, with the performance metrics varying by less than 2.5% from the oracle. These results suggest the potential of applying the computation allocation model for RAG to a wider range of knowledge-intensive tasks.

优劣性与创新点分析

当前 LLM 发展迅速，该文章为相关研究中较为新颖前沿的应用型研究，且由于其是较为策略性的，可应用性非常强，可以快速移植到现有 RAG 中。

创新点

- 不仅增加检索文档，还结合示例驱动与迭代推理两种策略；
- 揭示在最优配置下，RAG 性能可随推理计算近线性提升；
- 可预测不同预算下最优推理参数组合，为实际部署提供指导。

优势

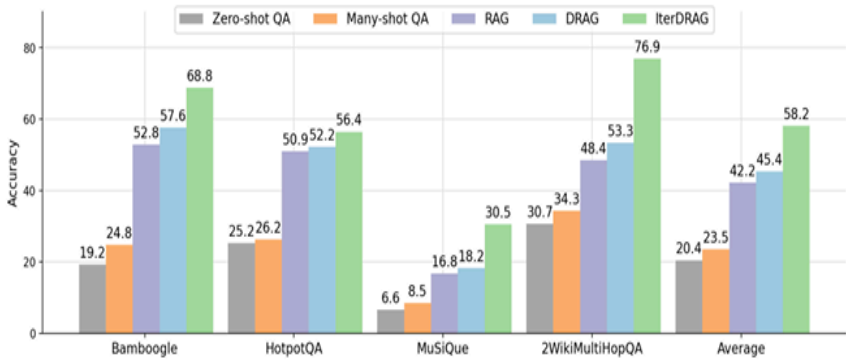
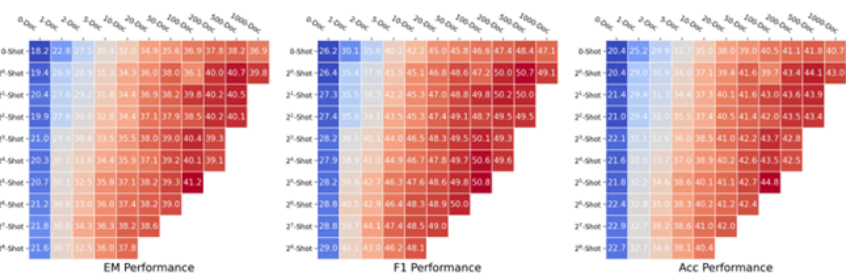
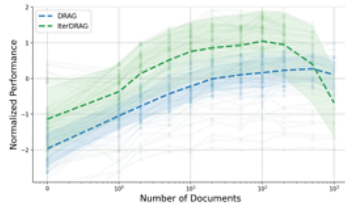


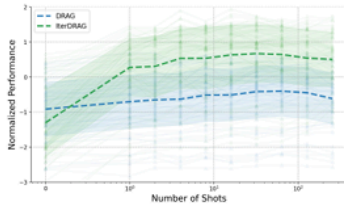
Figure 2: Evaluation accuracy of Gemini 1.5 Flash using different methods: zero-shot QA, many-shot QA, RAG (with an optimal number of documents), DRAG and IterDRAG on benchmark QA datasets. By scaling up inference compute (up to 5M tokens), DRAG consistently outperforms baselines, while IterDRAG improves upon DRAG through interleaving retrieval and iterative generation.



(a) Averaged DRAG performance heatmap for different metrics.



(b) Performance vs. number of documents.



(c) Performance vs. number of shots.

Figure 5: RAG performance changes with varying number of documents and in-context examples. 5a reports the averaged metric values across datasets, whereas in 5b and 5c each line represents the normalized performance of a consistent configuration with progressively increasing documents / shots.

Table 4: Length extrapolation results of the computation allocation model for RAG.

	16k → 32k			32k → 128k			128k → 1M			1M → 5M		
	EM	F1	Acc	EM	F1	Acc	EM	F1	Acc	EM	F1	Acc
Baseline	37.4	47.6	40.4	39.0	49.5	42.2	39.3	49.3	42.8	44.5	55.4	49.8
Predict	37.4	48.2	41.0	41.2	52.0	45.4	48.0	60.9	56.9	47.9	59.8	55.2
Oracle	39.2	49.8	42.7	46.9	59.0	55.1	50.5	62.1	57.7	51.7	62.6	58.1

Length Extrapolation. In addition to predictability on unseen domains, we explore the extrapolation of context length based on the computation allocation model. Here, we estimate the parameters of Equation (2) using experiments with shorter context lengths and assess their predictive accuracy on longer ones. We assess different extrapolation settings and present the predicted metric values in Table 4. Our observations are: (1) The predictions are accurate and consistently outperform the 8-shot baseline. For instance, the average difference between the predicted and oracle results from 128k to 1M tokens is just 2.8%. (2) Extrapolating from 32k to 128k is challenging. This is because DRAG performs best around 32k, while IterDRAG typically excels at a long context of 128k, as evidenced in Figure 4. Consequently, it creates a discrepancy between training and predicting performance distribution. (3) 5M context length is less predictable, with the average performance difference between predicted and oracle metrics observed at a substantial 5.6%. Overall, length extrapolation with computation allocation model is accurate and more effective for target lengths below 1M.

- 在大计算预算下，性能显著优于传统 RAG ；
 - 上下文示例、复杂查询拆为简单查询并分多轮进行
- 可根据预算与任务自动推荐参数，减少大量人工调参成本。
- 局限性
 - 仅适用于长上下文 LLM
 - 对 Retrieval Quality 依赖较大，当检索噪声高时性能提升有限；
 - 当然，这也是所有 RAG 共有的问题
 - 超过 1M token 后，长上下文建模能力有限，收益递减；
 - IterDRAG 在增强性能的同时会显著增加推理延迟与成本，实际应用时需权衡实时性与推理成本要求。

总结与思考

文章系统研究了长上下文检索增强生成（RAG）的推理规模化问题，提出了两种策略——DRAG（基于演示的 RAG）和 IterDRAG（迭代的 DRAG）。通过上下文学习与迭代查询分解，这些策略实现了 RAG 性能随有效上下文长度的近线性增长，并揭示了在最优推理参数下，RAG 性能与测试时计算量呈线性关系的“推理规模化规律”。基于此，作者构建了计算分配模型，用于预测不同预算下的最优推理参数组合，实验结果表明该模型能够准确估计最优配置。在多跳问答基准上，文章提出的方法显著超越了传统 RAG，展现了良好的跨领域泛化能力和长度外推能力。这些研究为优化长上下文 RAG 的推理策略提供了理论与实践指导，并为知识密集型任务中的计算资源优化奠定了坚实基础。

文章提出策略的可用性确实比较强，不过还是属于比较锦上添花的提示词工程层面的措施，感觉实际 RAG 性能还是更多地取决于检索质量（比如我将向量化模型 m3e 换为 BGE 后检索性能有了一些提升，生成质量也相应地得到了推进）与使用的模型质量（llama、Qwen、deepseek 的性能肯定是有差异的，更别说模型内的参数量差异也很大（比如 deepseek 从 1.5B 到 671B）），在这两者性能表现较好（达到优化瓶颈）的情况下就可以通过引入文章中的策略来从提示词工程的层面上进一步优化系统性能（此时的优化开销会低于进一步优化那两者的开销）。不过文章里提出的计算分配模型就是非常有意义的了，使用得当的情况下可以节省很多不必要的资源开销。

我一月份的时候用 langchain-chromadb-m3e 方案做过一个法律助手 RAG，当时每次查询仅附带检索结果与问题，没有附带示例，下面是我当时使用的模板与对问题 借款人去世，继承人是否应履行偿还义务？ 的提问结果。

```
identity = "你是专业的法律知识问答助手，必须严谨地按要求回答用户的提问。\\n用户会向你提供参考文档（以Doc_{{i}}命名，包含问题与回答，请直接忽略仅包含问题的片段。），\\n如果回答时你需要引用文档请直接叙述其中内容，\\n不允许在回答中包含文档编号或提到用户提供了文档（这对用户不可见）。"

template = """请结合Context中文档的上下文片段来回答问题，禁止根据常识或其它已知信息回答问题。\\n如果你不知道答案，直接简明扼要地回答“未找到相关答案”。


Question: {question}
Context: {context}

Answer:
"""
```

```
PS D:\Study Work\Electronic_data\CS\AAUniversity\Web\Lab\L3> & D:/python/python.exe d:/Study Work/Electronic_data/CS/AAUniversity/Web/Lab/L3/Askv2.py
借款人去世后，继承人是否应履行偿还义务取决于他们是否继承了借款人的遗产。根据《中华人民共和国继承法》第三十三条的规定，继承遗产应当清偿被继承人生前所负债务，偿还的范围以遗产的实际价值为限。如果继承人继承了遗产，则需在遗产实际价值范围内偿还债务；如果继承人放弃继承，则无需对债务负责。

因此，如果继承人继承了借款人的遗产，他们应在遗产实际价值范围内履行偿还义务；如果未继承遗产或放弃继承，则不应履行偿还义务。
```

上个月我又做了一个 arXiv 上 cs.CL 领域论文的 RAG （BGE-faiss-langchain），使用了以下模板

 提问模板

You must reply in English. You are a professional academic assistant specializing in the cs.CL domain. Strictly based on the provided literature information, answer the user's query. You may slightly expand and refine the response using existing knowledge, but avoid fabricating information. The response language must be English. Ensure the response is concise and limited to 500 words. Include a citation with the corresponding literature link (like [text](f"https://arxiv.org/abs/{id}")) and adjust the response style based on the temperature parameter (from 0.0 to 1.0). If the temperature is low (e.g., 0.2), provide a strict and precise answer. If the temperature is high (e.g., 0.8), provide a more creative and exploratory answer. And you must reply in English! Do not use Chinese or any other language!

Temperature: {temperature}

Question: {question}

Relevant Literature:

{context}

Answer:

例如下面就是一次完整的查询及其结果的大致展示

```
# 构造完整问答对
from time import time

i = 0
total_time = 0
all_text = []
for extracted_data, result in zip(queryys, results):
    query = [extracted_data]
    orig_q = [search.prompts.questions[i]]
    start_time = time()
    identity, res = search.search(query, orig_q)
    cost_time = time() - start_time

    all_text.append(identity+"\n" + res + "\n" + result)
    i += 1
    total_time += cost_time

print(f"avg time: {total_time / len(queryys):.4f} seconds")
```

Python

生成嵌入中: 100%|██████████| 1/1 [00:00<00:00, 14.60it/s]

You must reply in English. You are a professional academic assistant specializing in the cs.CL domain. Strictly based on the provided literature information, answer the user's query. You may slightly expand and refine the re:

Temperature: 0.2

Question: 什么是注意力机制？

Relevant Literature:

ID: 1810.10126, Distance: 0.3811461925506592, Content: {'title': 'Area Attention', 'doi': None, 'categories': 'cs.LG cs.AI cs.CL stat.ML', 'abstract': ' Existing attention mechanisms are trained to attend to individual item

ID: 2211.03495, Distance: 0.40938127040863037, Content: {'title': 'How Much Does Attention Actually Attend? Questioning the Importance of\n Attention in Pretrained Transformers', 'doi': None, 'categories': 'cs.CL cs.LG', 'al

ID: 1904.05873, Distance: 0.4193165898323059, Content: {'title': 'An Empirical Study of Spatial Attention Mechanisms in Deep Networks', 'doi': None, 'categories': 'cs.CV cs.CL cs.LG', 'abstract': ' Attention mechanisms have

ID: 1905.09856, Distance: 0.4276387691497803, Content: {'title': 'Copy this Sentence', 'doi': None, 'categories': 'cs.LG cs.CL stat.ML', 'abstract': ' Attention is an operation that selects some largest element from some se

ID: 2211.07714, Distance: 0.43440112471580505, Content: {'title': 'Revisiting Attention Weights as Explanations from an Information\n Theoretic Perspective', 'doi': None, 'categories': 'cs.CL cs.AI cs.LG', 'abstract': ' At

Answer:

Attention mechanisms are computational methods that allow models to focus on specific parts of input data when generating output. They were initially introduced to improve the performance of sequence-to-sequence models, such

In traditional attention mechanisms, the model computes an alignment score between elements of the input and the output being generated. These scores are then normalized (often using a softmax function) to produce attention v

A more advanced variant, called "area attention," extends this concept by allowing the model to attend to groups or areas of items rather than individual items [0]. For example, in image processing, this could mean attending

Despite the widespread use of attention mechanisms, recent studies have questioned their necessity in pretrained models. One study found that replacing input-dependent attention with constant matrices resulted in only minor i

From an explanatory perspective, attention weights are often interpreted as indicators of input importance. Yet, research has shown mixed results regarding their effectiveness as explanations [3]. While some attention mechan

[0] [Area Attention](https://arxiv.org/abs/1810.10126)

[1] [Copy this Sentence](https://arxiv.org/abs/1905.09856)


[2] [How Much Does Attention Actually Attend?](https://arxiv.org/abs/2211.03495)

[3] [Revisiting Attention Weights as Explanations](https://arxiv.org/abs/2211.07714)

=====

而如果要应用文章中提到的 DRAG 方法，仅需在 Temperature 的上方加上示例即可，示例可以通过精心准备优质问题进行检索并手动提问来构造，构造完毕后按实际要求对示例进行符合任务要求的精修；同时按照文章中的理论，应该将最相关的文档放在离查询最近的位置，我的模板中已经实现了这一策略，不过如果将查询放在最下面（更接近回答）然后倒转参考文档的顺序可能会更好一点？

在我的 RAG 中，包含一个“查询扩展”模块，其会将用户的提问拆分为多个领域相关的关键词，并将改写后的内容输入给检索系统进行向量化查询操作（引入原因一个是希望减少问题中的无效内容，另一个是需要与数据库中的语言保持一致）。

 查询问题改写

你是一个专业的学术助手，专注于cs.CL及其延伸领域。你的任务是从用户的问题中提取关键学术内容，并以关键词的形式表述。注意：你不需要直接回答问题，也不需要扩写问题，而是提取问题中的核心学术概念或主题，并以英文输出（严格输出且仅输出回复）。确保输出适合用于相似性向量查询。

Query: {query}

Extracted Academic Content:

就比如下面是几个示例。那么就可以对改写后的内容使用文章中的 IterDRAG 策略，每个子查询以逗号分隔，执行多次 检索-生成 任务，并将生成结果与原始问题综合起来形成新的提示词输入给模型用于最终答案的生成。不过如果要进行这项任务，那这里的提示词应该还是需要优化一下，比如下面第二个问题里的 performance comparison 单独作为一个问题显然不是太有必要。

原始问题	改写后
什么是注意力机制？	attention mechanism
对比 Transformer 与 RNN 在机器翻译中的性能	Transformer, RNN, machine translation, performance comparison
大语言模型的“幻觉”问题本质是语义理解缺陷还是推理过程偏差？ 如何构建可靠性评估框架？	large language models, hallucination problem, semantic understanding, reasoning bias, reliability evaluation framework
零样本学习中，提示词工程能否真正激活模型的“潜在知识”？ 其理论边界在哪？	zero-shot learning, prompt engineering, latent knowledge activation, theoretical boundaries