计算机研究与发展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(7): 1597−1610, 2025

基于滑动窗口策略的大语言模型检索增强生成系统

毕枫林[1] 张岂明[1] 张嘉睿[1] 王衍童[1] 陈 阳[2,3] 张琰彬[1] 王 伟[1] 周 烜[1]
[1](华东师范大学数据科学与工程学院 上海 200062)
[2](复旦大学计算机科学技术学院 上海 200433)
[3](上海市智能信息处理重点实验室(复旦大学) 上海 200433)
(fenglin@stu.ecnu.edu.cn)

A Retrieval-Augmented Generation System Based on Sliding Window Strategies in Large Language Models

Bi Fenglin[1], Zhang Qiming[1], Zhang Jiarui[1], Wang Yantong[1], Chen Yang[2,3], Zhang Yanbin[1], Wang Wei[1], and Zhou Xuan[1]
[1](*School of Data Science and Engineering*, *East China Normal University*, *Shanghai* 200062)
[2](*School of Computer Science*, *Fudan University*, *Shanghai* 200433)
[3](*Shanghai Key Laboratory of Intelligent Information Processing* (*Fudan University*), *Shanghai* 200433)

Abstract Leveraging a sliding window strategy, this study presents an innovative retrieval-augmented generation system aimed at enhancing the factual accuracy and reliability of outputs from large language models (LLMs). By applying a sliding window mechanism during the indexing phase, the project effectively addresses the limitations of fixed context window sizes and static retrieval methods. Three specific sliding window strategies are proposed to efficiently process and segment texts, such as fixed window size and fixed step length split (FFS), dynamic window size and fixed step length split (DFS), and dynamic window size and dynamic step length split (DDS). To further enhance retrieval accuracy and relevance, the project employs multiple advanced query techniques, including query expansion and reformulation. Rigorous experimental evaluations are conducted using the state-of-the-art LLaMA-3 model across multiple diverse datasets, encompassing both general knowledge and domain-specific corpora. Results demonstrate optimal performance with a carefully calibrated block size of 1 024 tokens and a step size of 3, significantly improving $F1$ score across various tasks. This configuration highlights the critical importance of balancing document segment length and sliding window step size to maximize information retention and retrieval efficacy.The sliding window strategy effectively preserves contextual information, reduces information loss, and exhibits adaptability across different datasets and query types.

Key words retrieval-augmented generation; sliding window mechanism; large language models; information retrieval; document question answering

摘 要 提出了一种基于滑动窗口策略的检索增强生成系统,旨在提升大语言模型(large language models, LLMs)输出的事实准确性和可靠性.该系统通过在索引阶段应用滑动窗口机制,有效解决了传统固定大小上下文窗口和静态检索方法的局限性.研究提出3种具体的滑动窗口策略以有效处理和分割文本,包括:固定窗口大小和固定步长分割(fixed window size and fixed step length split, FFS)、动态窗口大小和固定步长分割(dynamic window size and fixed step length split, DFS)以及动态窗口大小和动态步长分割

收稿日期:2024-05-31;**修回日期**:2025-02-21
基金项目:国家自然科学基金项目(62137001, 62277017, 61977026)
This work was supported by the National Natural Science Foundation of China (62137001, 62277017, 61977026).
通信作者:张琰彬(ybzhang@dase.ecnu.edu.cn)

(dynamic window size and dynamic step length split, DDS). 此外, 研究还探讨了使用多查询技术进一步提高检索的准确性和相关性. 实验评估使用 LLaMA-3 模型在多个数据集上进行, 结果表明系统在窗口大小为 1 024 和步长为 3 的配置下达到最佳性能, 该配置显著提高了 *F*1 得分, 体现了在文档段长度和滑动窗口步长之间保持平衡的重要性. 滑动窗口策略有效保留了上下文信息, 减少了信息丢失, 并展示了在不同数据集和查询类型中的适应性.

关键词 检索增强生成; 滑动窗口机制; 大语言模型; 信息检索; 文档问答

中图法分类号 TP39

DOI: 10.7544/issn1000-1239.202440411 **CSTR:** 32373.14.issn1000-1239.202440411

大语言模型(large language models, LLMs)是一类基于深度学习技术构建的复杂人工智能系统, 其核心目标在于实现对人类语言的大规模理解、生成与交互[1]. 在人工智能领域的前沿应用中, 特别是诸如知识问答等场景, 生成准确且事实正确的信息变得至关重要[2-3]. 随着 LLMs 能力的不断提升, 其可靠性问题也日益凸显. 一个显著的问题是这些模型倾向于生成看似合理但实际错误的信息, 这种现象被称为 LLMs 幻觉, 其严重削弱了人工智能驱动系统的可信度[4-6].

检索增强生成(retrieval-augmented generation, RAG)框架为解决上述问题提供了重要突破[7]. RAG 通过将检索机制直接集成到 LLMs 的生成过程中, 使模型能够实时访问并整合来自外部的上下文相关信息[8-9]. 在需要事实高度准确的应用中, 这种集成至关重要, 其有助于确保生成的内容是合理、准确并基于可验证的信息. RAG 框架因此极大增强了 LLMs 输出的可靠性, 使其在关键决策场景中更值得用户信赖[10].

尽管 RAG 框架展示了其创新性, 但它仍面临诸多显著挑战, 特别是在高度依赖上下文信息精确检索方面. RAG 的整体有效性在很大程度上依赖于从外部数据源或知识库中检索到的信息的相关性和准确性[11]. 如果检索组件未能成功获取到相关信息, 或获取的数据存在错误或误导性内容, 那将不可避免地对最终生成的内容质量产生显著负面影响[12]. 为应对这些问题, 我们提出使用滑动窗口技术, 该技术可以通过动态调整窗口大小来增强对上下文信息的捕捉, 减少信息丢失, 并提高检索结果的准确性. 这种技术已在相关研究中被证明能有效提升模型的表现, 特别是在处理长文本或多段复杂信息时表现尤为突出[13].

LLMs 正在迅速发展, 现已能够处理超过 200 000 个标记的扩展上下文, 从而显著提升了长文档问答的能力[14-15], 这种进展让 LLMs 在一定程度上不再受限于上下文长度, 使整个文档纳入提示词成为可能, 但 RAG 依然不可或缺, 原因如下: 将大量上下文一次性输入 LLMs 会显著影响推理速度, 而采用分块检索和按需输入则可以大幅提高系统效率; 同时, 基于 RAG 的生成方式可以快速定位 LLMs 生成答案所依据的原始参考资料, 这种特性能够帮助用户验证响应的准确性; 此外, 与仅依赖长上下文生成响应的黑盒性质相比, RAG 的整个检索和推理过程都是可观察的. 长文档问答的能力反而为 RAG 的发展提供了新的机遇, 使其能够解决更为复杂的问题, 并有能力回答那些需要广泛审查材料的综合性或总结性问题[16-17].

本文介绍了滑动窗口检索增强生成系统, 这是一种在 RAG 框架内的新方法, 旨在提高大语言模型的可靠性. 本系统根据文本内容动态调整窗口大小和步长, 旨在提高检索信息的精度和上下文相关性. 项目的贡献包括 3 个方面:

1)提出 3 种不同的文本分割策略以适应多样的文本结构和连贯性需求, 包括固定窗口大小和固定步长分割(FFS)、动态窗口大小和固定步长分割(DFS), 以及动态窗口大小和动态步长分割(DDS). 这些策略灵活调整窗口大小和步长, 以更好捕捉文本中的关键信息, 同时确保分割后文本片段的连贯性.

2)将这些文本分割策略与多查询机制集成, 以提高检索结果的准确性. 通过两者结合, 本系统能够更加准确地从外部数据源提取和整合相关信息, 从而生成更可靠和更可信的内容.

3)经过一系列的实验研究, 本系统在解决生成内容中常见的幻觉问题及提升生成的信息准确性方面的有效性得到充分验证和高度认可.

本文的研究展示了滑动窗口检索增强生成系统在提升 LLMs 输出可靠性方面的潜力, 特别是应用在需要高事实准确性的项目中. 通过动态调整文本分割策略和多查询机制, 滑动窗口检索增强生成系统不仅提高了检索信息的精度, 还增强了生成内容的上下文相关性和连贯性, 从而为用户提供更可信赖的生成式人工智能解决方案.

1 相关工作

Gao 等人[10]将 RAG 的范式划分为初级 RAG、进阶 RAG 和模块化 RAG 三个阶段，这些阶段代表了 RAG 技术从基本的索引检索方法发展到复杂且进一步模块化的演进路径. 每个范式都固有地包含一个检索组件，突显了持续优化的必要性. RAG 系统的优化可以分为检索源优化、索引优化和查询优化[18]三个主要类别.

在检索源优化阶段，Luo 等人[19]提出了将表格数据转换为文本格式，从而便于应用基于文本的分析方法，然而此方法在实际应用中通常表现出较低的准确性. 值得注意的是，一个显著的进展是 G-Retriever 系统的提出，该系统巧妙地融合了图神经网络、LLMs 和检索生成技术. 通过对 LLMs 进行软提示，G-Retriever 系统显著提升了图结构理解和问答处理能力. 此外，该系统采用奖收集斯坦纳树优化方法，成功实现了有针对性的图检索[20]. 尽管这一方法确实具备显著的优点，但建设、验证以及维护结构化数据库依然需要投入大量的精力与资源.

在索引优化阶段，文档会经过处理、分割，并最终转换成嵌入形式存储在向量数据库中. 高质量的索引构建是至关重要的，直接影响到检索的精度. 一种普遍采用的方法是将文档按固定大小的块进行分割（如 100, 256 或 512 个标记）[21]. 虽然使用较小的块可以有效减少噪声，但这种方式不能完全传达所需的上下文信息. 此外，将文档分割成块可能会导致句子内部发生截断现象，这一问题引发了递归分割技术的研究，该技术通过在多次检索过程中合并全局相关信息，以实现更层次化的检索[22]，然而递归分割技术存在灵活性不足和性能较差的缺点. 本系统在借鉴这一策略的基础上，将数据划分为特定的字符串和词语元素，但在语义完整性与上下文长度之间找到最佳平衡依然困难. 为解决此问题，项目进一步提出滑动窗口策略的思想，通过在相邻文本块间引入适度重叠，增强连贯性并保持文本的完整性. 此外，该策略具有较低的时间复杂度，有效平衡了语义完整性与计算效率.

在查询优化阶段，RAG 的主要挑战之一是其对用户原始查询的依赖，这通常会导致次优结果，Dhuliawala 等人[23]开发的“验证链”方法通过 LLMs 对扩展查询进行验证以提高可靠性. 在实际应用中，原始查询也可能不适合检索，提示 LLMs 重写查询已被验证是有效的，“重写—检索—阅读”模型[24]以及在淘宝 BEOUE 系统中实施查询重写方法[25]均显著提高了长尾查询的召回效果.

2 方法概述

每个基于滑动窗口策略的检索增强生成系统包含 2 个主要组件：由参数θ表示的文本生成模型G_θ，以及由参数ϕ表示并从大型文档集合C中检索文档的检索模型R_ϕ.

2.1 问题定义

RAG 期望效用 U 的定义如式（1）所示，首先设$T=\{(x_1,y_1),(x_2,y_2),\dots,(x_n,y_n)\}$为包含 n 对的 x_i（输入文本）和 y_i（真实输出文本）的训练集，U 接受由 RAG 系统生成的输出$\hat{y}$和真实输出 y，并生成 1 个标量值.

$$U=\frac{1}{n}\sum_{(x,y)\in T}\sum_{\hat{y}\in Y}U(y,\hat{y})p(\hat{y}|x;G_\theta,R_\phi). \tag{1}$$

效用函数可以是任意的度量标准，包括但不限于精确匹配、术语重叠 $F1$、BLEU、ROUGE. 项目假设：

1）效用值越高越好；

2）效用函数在 [0, 1] 范围内；

3）$U(y,\hat{y})=1$.

在某些模型中，输出空间是有限的，例如在事实验证中，输出空间通常是二元的，即给定的候选事实要么是真实的，要么是虚假的. 然而，在其他情况下，如自由形式的文本生成，输出空间则是无限的.

在 RAG 系统中生成任意给定输出$\hat{y}$的概率可以建模为式（2）.

$$\begin{aligned}p(\hat{y}|x;G_\theta,R_\phi)=&\sum_{d\in\pi_k(C)}p(\hat{y},d|x;G_\theta,R_\phi)=\\&\sum_{d\in\pi_k(C)}p(\hat{y}|x,d;G_\theta)p(d|x;G_\theta,R_\phi)=\\&\sum_{d\in\pi_k(C)}p(\hat{y}|x,d;G_\theta)p(d|x;R_\phi),\end{aligned} \tag{2}$$

其中$\pi_k(C)$表示从检索集合C中选择 k 个文档的所有排列组合. 式（2）的第 1 步是使用全概率公式，第 2 步使用链式法则，第 3 步则基于结果列表 d 被检索到的概率与文本生成模型G_θ无关这一事实.

本系统的主要目标在于提高检索相关文本的概率，具体来说，就是增加结果列表 d 中相关信息的概率. 为了实现这一目标，项目引入了索引过程，其关键在于将原始、未经结构化处理的内容转化成计算机能够理解和处理的格式. 这个过程不仅使得内容

的结构变得更加清晰，还显著提升了计算机对数据的解析和操作能力. 通过这样的转化，索引内容得以更高效地被存储和组织起来，从而为后续的信息检索和数据处理任务打下坚实基础.

具体过程如图 1 所示，该过程的初始步骤包括对内容进行索引处理，其中非结构化的内容首先被转换为长文本格式，然后再被细分成多个“块”，每个块包含相对独立的信息单元. 这些块经过处理后会被嵌入并存储在向量数据库中，以确保每条信息可以在需要时进行高效的检索和分析.

为此，项目提出了一种滑动窗口的方法，有效地将文档拆分为一系列独立的块. 如式(3)所示.

$$f(D,w,s)=[W_1,W_2,\cdots,W_n], \tag{3}$$

其中，w 表示窗口大小，s 表示步长，D 表示文档，而 n 表示生成的块的数量，n 取决于 D 的长度、w 和 s. 函数 f 生成一系列块：

该问题的核心挑战在于选择适当的分割方法并确定合适的 w 和 s 值，以确保：

1)每个块内的连贯性，避免在重要语义单元(如句子中间或关键上下文连接处)中断.

2)全面捕捉 D 中的所有关键信息，确保生成的块中没有遗漏.

3)提升方法对各种文本结构和内容的适应性，满足多种应用场景的需求.

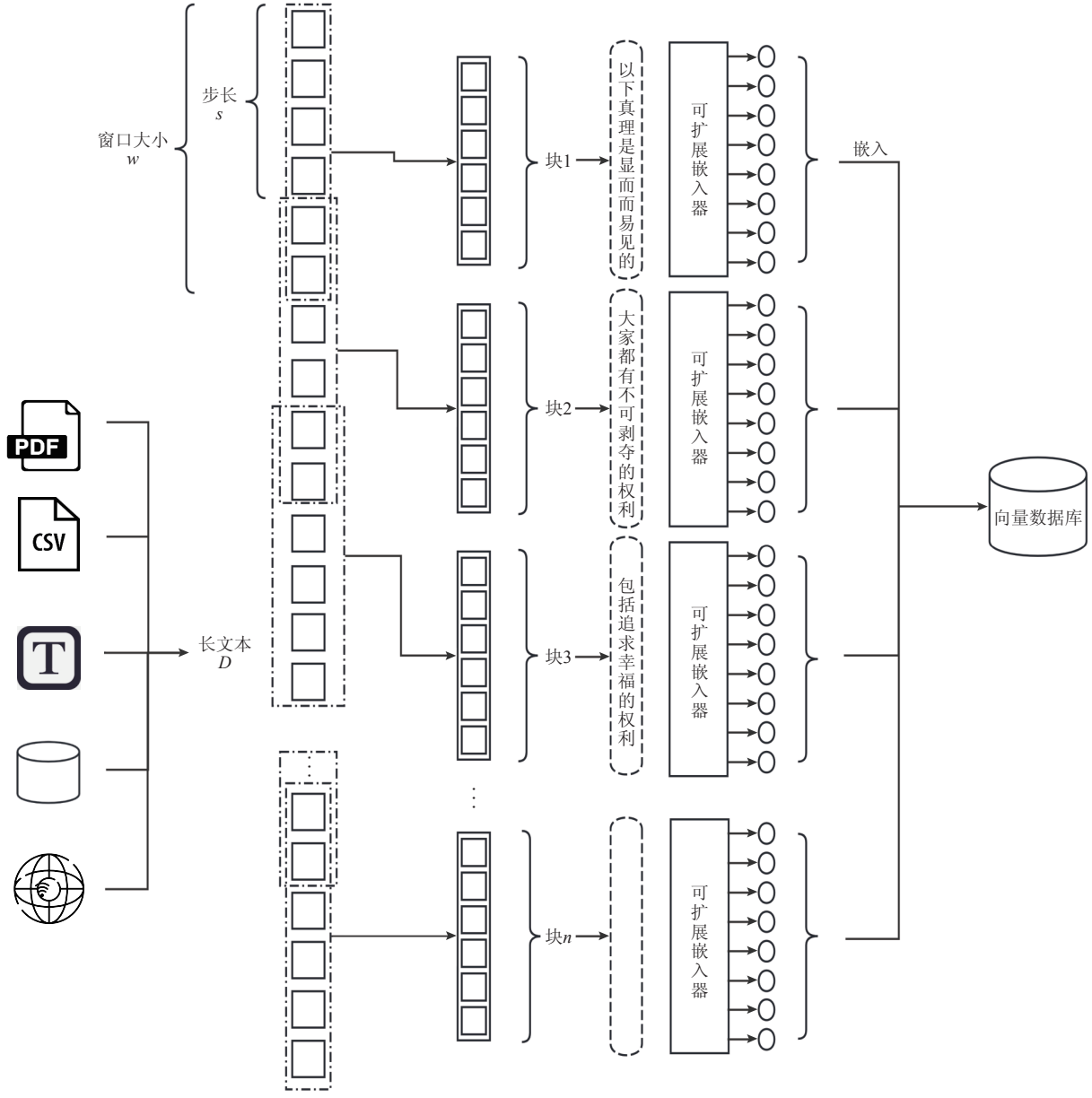


Fig. 1 Workflow of indexing

图 1 索引工作流程

2.2 滑动窗口的基本概念

为了有效处理文本信息并使关键信息的捕获最大化，同时确保文本连贯性，项目提出了一种基于滑动窗口方法的文本分割策略. 考虑到滑动窗口方法的实现目标，项目进一步细化并引入了 3 种具体的文本分割策略，以满足不同的需求.

滑动窗口方法旨在最大化关键信息的捕获，确保文本连贯性并解决模型标记限制带来的约束. 假设文档 D 的长度为 L，窗口大小为 w，步长为 s，则文档可以分割为 N 个块，其中 N 可以近似表示为式(4).

$$N = \left\lceil \frac{L-w}{s} \right\rceil + 1, \tag{4}$$

其中$\lceil x \rceil$表示向上取整函数，其基于初始窗口从文档的开头开始，然后每次移动步长 s. 对于每个窗口 W_i ($1 \leqslant i \leqslant N$)可以表示为式(5).

$$W_i = D[(i-1)\times s : \min((i-1)\times s + w, L)], \tag{5}$$

其中使用 min() 函数确保最后一个窗口不超过文档 D 的长度.

结合式(4)(5)，项目可以清楚地描述如何将文档 D 分解为一系列窗口 W 并计算每个窗口的内容. 此外，在考虑动态窗口大小和步长时，可以根据需要修改式(4)(5)以反映这些动态调整.

2.3 滑动窗口的分割策略

为有效处理和分割文本，项目提出 3 种具体的滑动窗口策略，包括：固定窗口大小和固定步长分割(fixed window size and fixed step length split, FFS)、动态窗口大小和固定步长分割(dynamic window size and fixed step length split, DFS)以及动态窗口大小和动态步长分割(dynamic window size and dynamic step length split, DDS)，这 3 种策略的区别在图 2 中进行了示例说明.

这些策略的设计旨在提高文本处理的效率与准确性，FFS 策略通过预设的固定窗口大小和步长进行分割，适用于结构较为均匀的文本；DFS 策略允许窗口大小根据文本内容动态调整，但步长保持不变，从而适应文本多样性；DDS 策略进一步灵活化，不仅窗口大小可变，步长也可根据具体需求动态调整，从而在复杂文本处理中提供更高适应性和精确性. 这些策略的理论基础和实际应用效果将在后续章节中详细探讨.

2.3.1 固定窗口大小和固定步长分割（FFS）

FFS 是一种直观且简单的文本处理策略. 在这种方法中，窗口在文本中移动的距离以及窗口本身大小都是预设的、不变的. 这种技术适用于结构清晰、内容连贯性要求不高的文本类型. 由于这种方法的窗口大小 w 和步长 s 是预设的，每个分割的窗口可以用式(5)进行表示.

算法 1 是实现固定窗口大小和固定步长分割的伪代码，详细描述了使用固定窗口大小和固定步长对文档进行分割的计算步骤. 该算法通过从头到尾迭代文本，生成可能的窗口，并根据预定义的窗口大小 w 和步长 s 调整起始点和结束点.

算法 1. 固定窗口大小和固定步长分割.

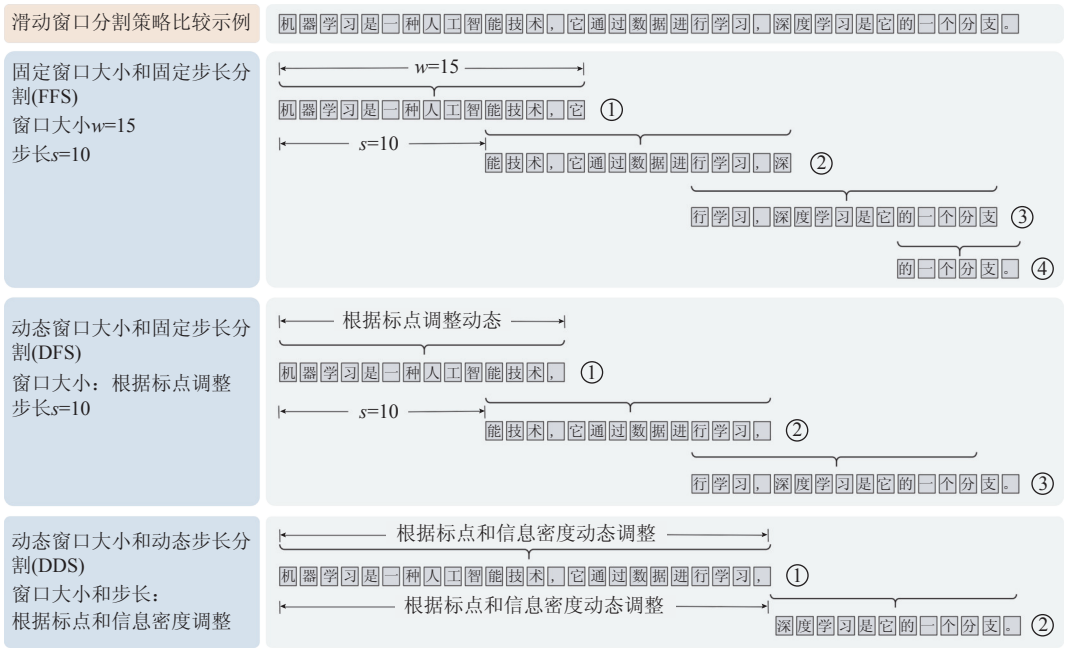


Fig. 2 Comparison of examples of sliding window segmentation strategies

图 2 滑动窗口分割策略的示例比较

输入: 文本 D, 窗口大小 w, 步长 s;
输出: 窗口列表 W.
① $W \leftarrow []$;
② $N \leftarrow \left\lceil \frac{length(D)-w}{s} \right\rceil + 1$;
③ for i=1 to N do
④ $start \leftarrow (i-1)\times s$;
⑤ $end \leftarrow \min(start+w, length(D))$;
⑥ $W_i \leftarrow D[start : end]$;
⑦ Append W_i to W;
⑧ end for
⑨ return W.

2.3.2 动态窗口大小和固定步长分割 (DFS)

该策略允许在固定步长的同时动态调整窗口大小, 以确保文本中的语义单元(如句子或段落)不被不当分割. 这一机制的核心目标是确保文本中的语义单元, 诸如句子或段落, 能够保持其完整性, 避免在处理过程中被不恰当地分割. 此方法尤其适用于那些结构清晰、组织良好的文本, 通常这些文本包含明确的句子边界和段落分隔.

该策略步长 s 保持不变, 而窗口大小 w 可在遇到特定符号(如句号和段落标记)时动态调整. 因此, 尽管窗口大小可能大于 w, 但每个窗口都尽量保持语义完整性. 窗口 W_i 是从文本 D 中提取的第 i 个窗口, 并由式(6)表示, 结束位置由固定窗口大小 w 和起始位置后下一个特定符号的位置共同确定.

$$W_i = D[(i-1)\times s : \min((i-1)\times s + w + next_symbol(D,(i-1)\times s, T), L)]. \quad (6)$$

算法 2 概述了一种实现动态窗口大小和固定步长分割的方法. 该算法通过检测文本中的特殊符号, 动态调整窗口分割的大小. 在具体操作过程中, 算法通过循环遍历文本, 每当经过一个固定步长后, 就定位下一个出现的特殊符号, 然后依据该符号的位置动态调整当前窗口的大小, 从而确保分割后的文本片段在语义上保持连贯性, 并保留了原始文本的结构完整性.

算法 2. 动态窗口大小和固定步长分割.
输入: 文本 D, 特定符号 T, 窗口大小 w, 步长 s;
输出: 窗口列表 W.
① $W \leftarrow []$;
② $splits \leftarrow split_text_with_regex(D, T)$;
③ $start \leftarrow 0$;
④ while $start < length(D)$ do
⑤ $end \leftarrow start + w$;
⑥ for each $split$ in $splits$ do
⑦ $split_start \leftarrow find(split, start)$;
⑧ if $split_start \geqslant end$ then
⑨ $end \leftarrow next_symbol(D, split_start, T)$;
⑩ break;
⑪ end if
⑫ end for
⑬ $chunk \leftarrow D[start : end]$;
⑭ $W.append(chunk)$;
⑮ $start \leftarrow start + s$;
⑯ end while
⑰ return W.
⑱ function $next_symbol(D, position, T)$
⑲ for each i from $position$ to $length(D)$ do
⑳ if $D[i]$ is in T then
㉑ return i+1;
㉒ end if
㉓ end for
㉔ return $length(D)$;
㉕ end function

2.3.3 动态窗口大小和动态步长分割 (DDS)

该策略根据文本内容动态调整窗口大小和步长, 确保全面覆盖所有重要细节, 特别是当关键或密集信息分散在文本中的情况下. 该方法适用于信息密度变化较大或关键信息分散在特定位置的文本. 值得注意的是, 该策略中的步长由遇到的特定字符数量决定, 这与 FFS 和 DFS 中使用固定字符步长不同.

窗口大小和步长的动态调整由式(7)表示, 其中 W_i 表示文档 D 中提取的第 i 个窗口. 窗口从起始位置 $(i-1)\times s$ 开始, 跨度由窗口大小 w 和起始位置后下一个特殊符号位置共同决定. 为确保窗口不超过文本 D 的总长度, 故取 $(i-1)\times s + w + next_symbol(D,(i-1)\times s, T)$ 和 L 的最小值.

$$W_i = D[(i-1)\times s + next_symbol(D,(i-1)\times s, T) : \min((i-1)\times s + w + next_symbol(D,(i-1)\times s, T), L)]. \quad (7)$$

算法 3 是实现动态窗口大小动态步长划分的伪代码. 该算法首先使用特殊符号 T 对文本 D 进行分割, 并用以识别重要段落, 然后通过动态调整窗口大小和步长, 基于内容和特殊符号位置迭代地创建窗口 W, 确保嵌套循环结构覆盖整个文本, 同时适应不同信息密度.

算法 3. 动态窗口大小和动态步长分割.
输入: 文本 D, 特定符号 T, 窗口大小 w, 步长 s;
输出: 窗口列表 W.

① $W \leftarrow []$;
② $splits \leftarrow split_text_with_regex(D,T)$;
③ $start \leftarrow 0$;
④ while $start < length(splits)$ do
⑤ $start_index \leftarrow start$;
⑥ $total_length \leftarrow 0$;
⑦ $temp_chunk \leftarrow''$;
⑧ while $start_index < length(splits)$ do
⑨ $total_length \leftarrow total_length+length(splits[start_index])$;
⑩ $temp_chunk \leftarrow temp_chunk+splits[start_index]$;
⑪ if $total_length \geqslant w$ then
⑫ $W.append(temp_chunk)$;
⑬ break;
⑭ end if
⑮ end $\leftarrow next_symbol(D, split_start)$;
⑯ end while
⑰ $start \leftarrow next_symbol(splits, start, s, T)$;
⑱ end while
⑲ return W.
⑳ function $next_symbol(splits, position, count, T)$
㉑ $symbol_count \leftarrow 0$;
㉒ for each i from $position$ to $length(splits)$ do
㉓ if $splits[i]$ is in T then
㉔ $symbol_count \leftarrow symbol_count + 1$;
㉕ end if
㉖ if $symbol_count \geqslant count$ then
㉗ return i+1;
㉘ end if
㉙ end for
㉚ return $length(D)$;
㉛ end function

3 系统实现

滑动窗口检索增强生成系统通过在检索过程中实施滑动窗口机制，解决了固定窗口和静态检索方法的常见限制，这些限制通常会导致上下文丢失和信息检索错误. 本系统架构由多个关键组件组成，每个组件在生成准确响应的过程中发挥着重要作用.

如图 3 所示，工作流程始于收集各种文档格式，包括 PDF、CSV 和纯文本，这些文档作为信息检索的私有数据源.

文档首先经过预处理，以识别句子和段落等语义单元，这一步确保了分割不破坏语义边界. 然后应用滑动窗口技术将文本分割成重叠的块，这些块被索引并嵌入到向量数据库中. 这种索引方法允许更细粒度和语境上更准确的信息检索，从而增强 LLMs 生成可靠和连贯响应的能力. 具体实现过程有 3 个：

1）窗口分割. 此过程主要包括对基于内容的语义结构进行动态分析，从而调整窗口的大小和步距. 这一调整的目的是高效识别文档中的特定标记，如标点符号和段落分隔符. 通过灵活设定窗口大小，系统能够在每个文本段落中包含完整的语义单元，进而显著提升了上下文检索的相关性和信息的完整性. 这种方法确保了在处理和分析文本时，能够更准确地捕捉和理解文档的结构和内容.

2）嵌入和索引. 每个分割后的窗口将被输入到经过预训练的语言模型中处理，最终转化为相应的向量表示. 这些生成的嵌入向量随后会被存储在向量数据库中，以确保数据能够实现高效存储和精确检索[26].

3）预检索. 此阶段参考了 MultiQueryRetriever[27] 的技术，将查询嵌入到基于距离的向量数据库中的高维空间进行检索. 然而，若查询或嵌入未能准确捕捉数据语义的变化将可能导致不同的检索结果. 此阶段有时需要手动调整或通过提示工程来解决这些问题，这可能既繁琐又低效. 为了克服这些限制，项目在滑动窗口检索增强生成系统中采用了一种先进特性，即 LLMs 自动生成给定用户查询的多个变体. 这种方法旨在从多个角度处理查询，通过增加获取最相关和语境上适当文档的机会来增强检索过程. 图 4 展示了这一过程，其中多个查询被用于探测向量数据库，显著提高了系统检索信息的准确性和相关性.

1）检索. 在接收到用户的多个查询后，系统将通过先进的检索算法访问向量数据库中最相关的段落. 对于每个查询，系统检索到一组相关文档后，将融合所有查询以形成一个更大且更加相关的文档集. 该技术通过提供同一问题的多个视角，提高了对主题的全面覆盖，并增强了对细微差异的捕捉能力.

2）回答. 检索到的文本片段将作为上下文输入，供 LLMs 处理. LLMs 综合利用这些信息，生成连贯且符合事实的回应. 通过调用全面的训练数据和所提供的上下文，LLMs 能够构建不仅相关且准确的答案.

4 实验评估

项目对滑动窗口检索增强生成系统的评估采用

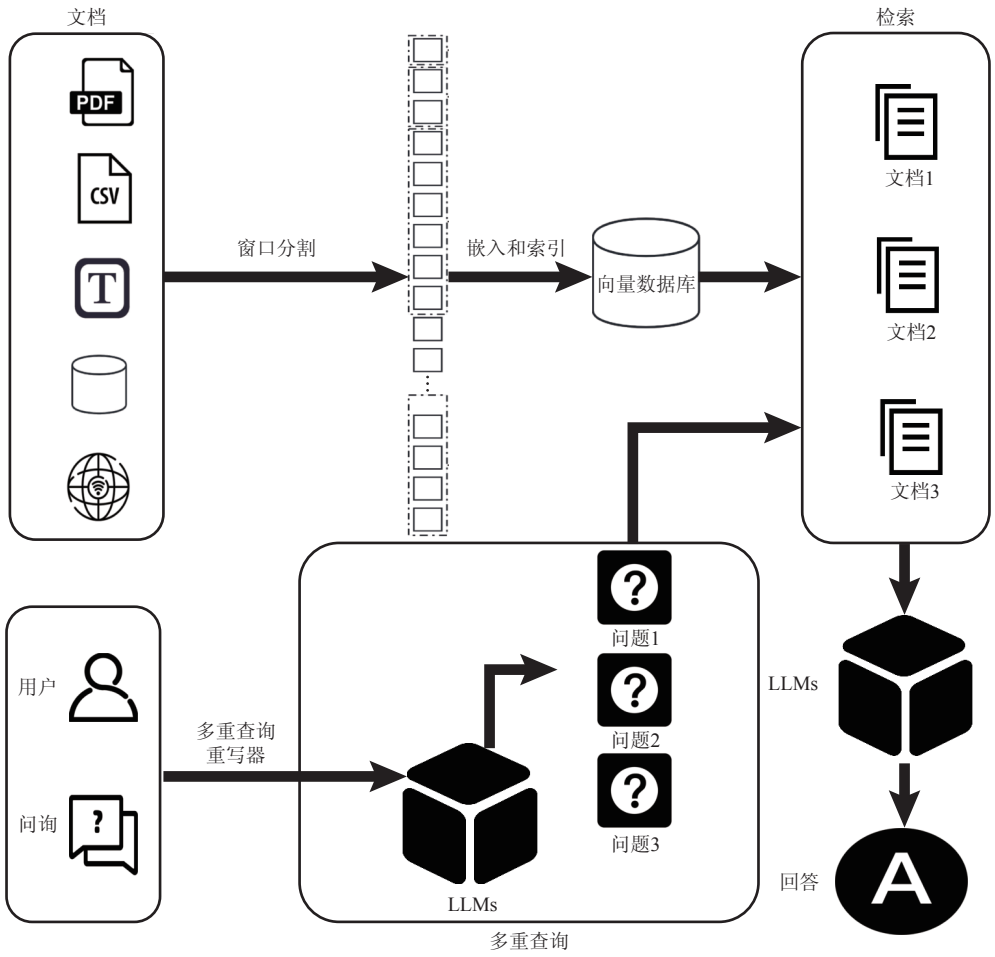


Fig. 3 Sliding window strategy based retrieval enhancement generation system architecture

图 3 基于滑动窗口策略的检索增强生成系统架构

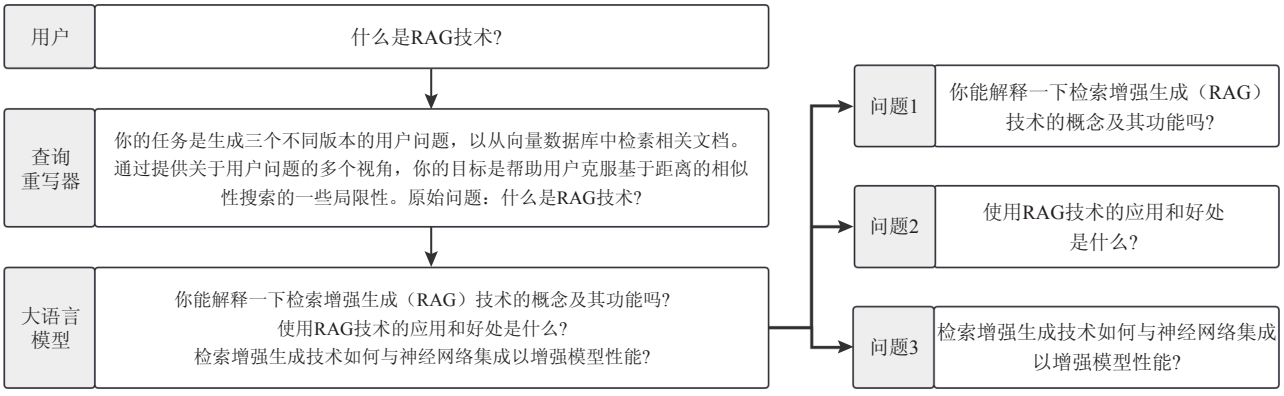


Fig. 4 Sliding window strategy-based multiple query example in retrieval-augmented generation system

图 4 基于滑动窗口策略的检索增强生成系统中的多重查询示例

了严格的方法论，其中响应主要利用 LLaMA-3 模型[①]生成. LLaMA-3 模型由 Meta AI 开发，代表了最新的 LLMs，在各种自然语言处理基准测试中表现出色[28]. 项目的实验包括对比分析各种滑动窗口策略和现有的段落分割方法. 此外，项目还进行了多查询与单查询场景的消融实验，以进一步揭示系统在不同条件下的性能. 通过这些系统性的实验，项目不仅验证了滑动窗口检索增强生成系统的有效性，还为未来优

① https://llama.meta.com/llama3

化和提升此类系统提供了宝贵的数据和经验.

4.1 实验数据集

实验数据集源自 LongBench[29] 和 Scrolls[30] 基准，这些基准因其复杂性和多样性在研究社区中得到了认可和广泛应用. 为确保分析的全面性和准确性，本项目选取并测试了 5 个数据集，涵盖了不同类型的文档和多样的查询场景，具体见表 1. 这 5 个数据集的选择旨在为不同情境下的分析提供依据，从而深入探讨其在多种应用场景中的性能表现.

4.2 实验设计

对于文档存储和检索，项目使用 ChromaDB 向量数据库①，采用最大边际相关性算法进行文档选择，该算法优化了查询的相关性和文档的多样性.

项目嵌入模型选择 LLaMA-3-chat embeddings，它是 LLaMA-3 模型的扩展，基于优化的自回归变压器框架构建，并专门针对对话和问答任务微调.

在推理过程中，项目采用贪婪解码，温度设置为 0，重复惩罚为 1.1，以确保生成的响应精确且不重复. 项目使用的 RAG 提示词如下：

> 基于以下上下文回答问题为{context}；
> 问题为{question}.

项目实验一方面通过对比分析各种滑动窗口策略和现有段落分割方法来考察系统的有效性，另一方面设计并实施了多查询与单查询情境下的消融实验以深入了解系统在不同场景下的性能表现.

项目对 FFS、DFS、DDS 以及常用的方法进行了一系列评估. 此外，项目还进一步使用了常用的 RCS（RecursiveCharacterTextSplitter）[34] 作为基线，将 RCS 的窗口大小分别设置为 512, 1 024, 2 048，以便在相同数据集上进行比较分析.

实验参数选择如下：对于 FFS 和 DFS，项目分别使用了窗口大小为 512，步长为 256, 128, 64；窗口大小为 1 024，步长为 512, 256, 128；窗口大小为 2 048，步长为 1 024, 512, 256. 对于 DDS，步长由特殊字符的数量动态确定，项目测试了步长为 2, 3, 5 下不同分割粒度的影响. 用于 DFS 和 DDS 的特殊分割字符包括“\n\n”“\n”“.”“,”“？”“！”.

项目选定了 512, 1 024, 2 048 这 3 个窗口大小，长度参考自 GPT-3.5 之后的 LLMs 常用的 4 096 个令牌范围[35]，并考虑到多数研究认为短文本长度为 4 096 个令牌[36]，因此项目以 4 096 的 1/2 及其变体作为窗口大小的基准. 这样的设置不仅符合常规的文本处理习惯，也能有效处理不同长度的文本. 至于步长，其设置对数据处理的重叠程度和结果的连续性有直接影响，项目将步长定为窗口大小的 1/2, 1/2, 1/8. 在 DDS 算法中，项目选择使用质数作为步长，以避免不同步长之间的倍数关系导致的内容重叠，确保数据分割的多样性与广泛覆盖，优化了算法的整体性能.

4.3 评估指标

项目的评估指标为 $F1$ 得分，它提供了精确度（*Precision*）和召回率（*Recall*）的平衡度量. 在项目的分析中，*Unigramspred* 表示系统输出中的单元集合，而 *Unigramscorrect* 表示参考文本中的单元集合. 精确度和召回率根据式（8）和式（9）计算.

$$Precision = \frac{|Unigramspred \cap Unigramscorrect|}{|Unigramspred|}, \tag{8}$$

$$Recall = \frac{|Unigramspred \cap Unigramscorrect|}{|Unigramscorrect|}. \tag{9}$$

因此，$F1$ 得分可进一步通过式 (10) 计算：

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \tag{10}$$

4.4 实验情况

实验结果如表 2 所示. 在英文数据集 MultiFieldQA-en 中，不同策略之间的性能差异并不显著，仅 DDS 略微优于其他策略，其 $F1$ 得分达到 49.81%. 在中文数据集 MultiFieldQA-zh 中，DDS 表现尤为突出，尤其在窗口大小为 1 024 时优势更加明显. 在窗口较

Table 1 Datasets Used in the Study

表 1 研究中使用的数据集

数据集	来源	平均长度	语言	数据点	问答类型
NarrativeQA [31]	文学、影视	18 409 个令牌	英文	200	单文档问答
Qasper [32]	科学	3 619 个令牌	英文	200	单文档问答
MultiFieldQA-en [29]	多领域	4 559 个令牌	英文	150	单文档问答
MultiFieldQA-zh [29]	多领域	6 701 个令牌	中文	200	单文档问答
HotpotQA [33]	维基百科	9 151 个令牌	英文	200	多文档问答

① https://www.trychroma.com

Table 2 *F*1 Scores Under Various Window Sizes and Step Configurations

表 2 不同窗口大小和步长配置下的 *F*1 得分

方法	窗口大小	步长	*F*1 得分/%	
			MultiFieldQA-en 数据集	MultiFieldQA-zh 数据集
RCS（基线）	512		47.35	44.89
	1 024		47.55	**50.47**
	2 048		**48.10**	49.59
FFS（本文）	512	64	46.82	**45.65**
	512	128	**48.84**	39.70
	512	256	49.21	30.95
	1 024	128	45.49	40.38
	1 024	256	46.02	31.95
	1 024	512	47.56	21.98
	2 048	256	48.18	30.69
	2 048	512	47.52	22.00
	2 048	1 024	47.86	19.76
DFS（本文）	512	64	40.76	48.46
	512	128	**47.61**	49.10
	512	256	46.12	48.62
	1 024	128	47.51	48.77
	1 024	256	46.38	52.15
	1 024	512	46.41	50.51
	2 048	256	48.93	48.07
	2 048	512	46.37	47.12
	2 048	1 024	44.86	**53.05**
DDS（本文）	512	2	45.37	45.48
	512	3	46.91	47.61
	512	5	45.92	46.15
	1 024	2	47.57	51.24
	1 024	3	**49.81**	**53.36**
	1 024	5	46.90	52.42
	2 048	2	46.10	51.87
	2 048	3	45.70	50.89
	2 048	5	45.12	51.45

注：黑体数值表示最优结果.

小(512)的英文数据集中, FFS 策略表现相对较好. 整体而言, FFS 策略在中文数据集中的表现相对较差, 尤其容易出现诸如“无法根据上下文回答”的情况, 这表明 FFS 不适合应用于中文文本分割. 相比之下, DFS 在 2 种数据集中表现较为平衡, 虽然在英文数据集中表现最差, 但在中文数据集中与 DDS 的差距并不显著.

综合来看, DDS 在中英文语料库中都呈现出最佳表现, 特别是中文语料库中, 其相比于基线 RCS 效果更显著. 对于 FFS 和 DFS, 在不同窗口大小和步长下性能有所波动, 部分窗口大小和步长下能超越 RCS.

此外, 项目还进行了多查询消融实验, 以测试滑动窗口检索增强生成系统在多查询情况下是否能提高答案的准确性. 项目测试了所有数据集, 并采用 DDS 策略. 图 5 的结果显示, 多查询在大多数数据集中比单查询更具性能优势. 此外, 最佳窗口大小和步长分别为 1 024 和 3 时, 提供了最高的 *F*1 得分.

上述结果表明, 滑动窗口检索增强生成系统不仅在滑动窗口策略上明显优于传统段落分割方法, 而且在多查询场景下显著提升了检索结果的准确性和相关性.

5 讨 论

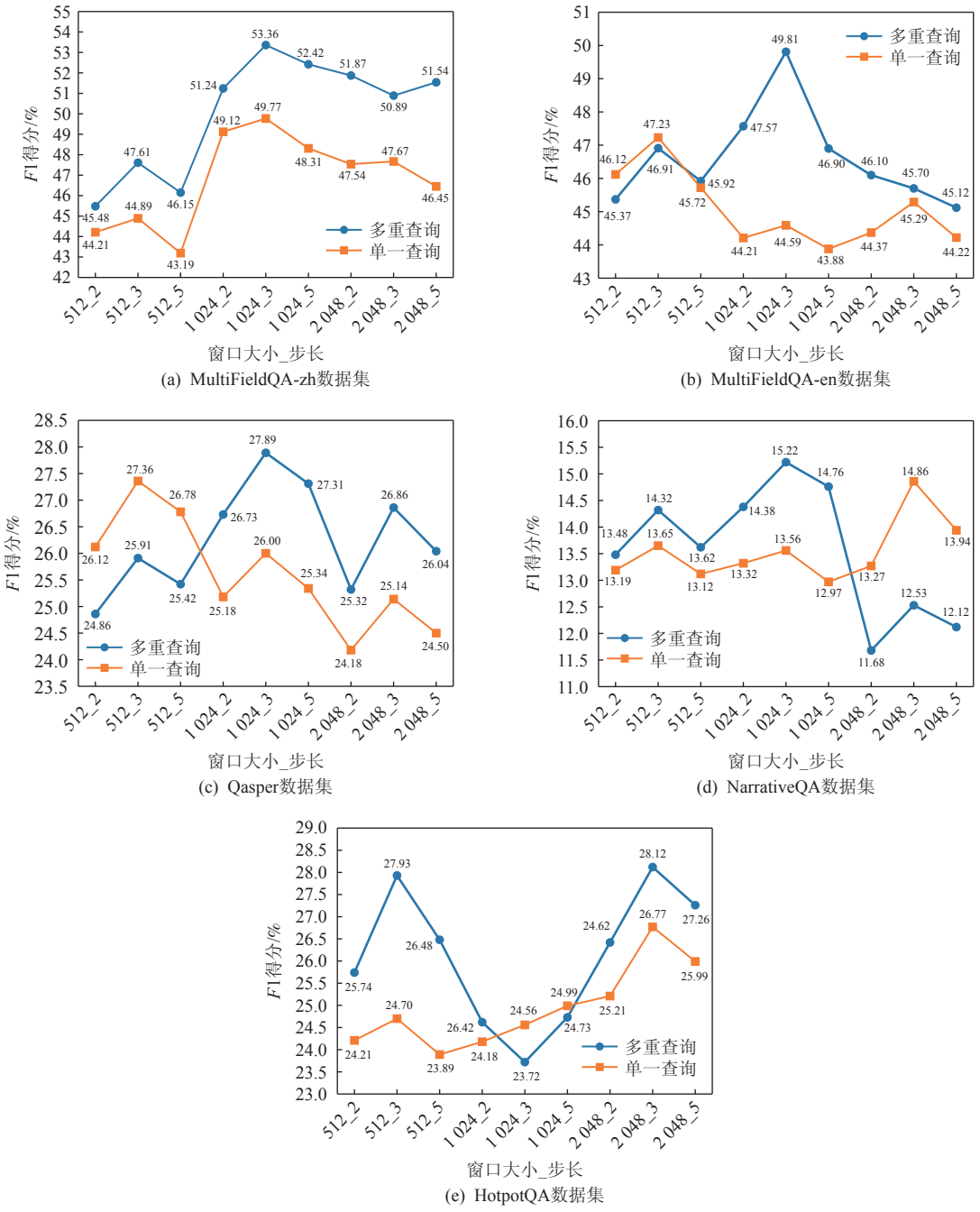
项目的实验结果表明, 基于 DDS 的方法在多个数据集上表现良好, 特别是在使用多查询时显著提高了性能. 以下是对这些结果的详细讨论.

5.1 影响因素分析

1)窗口大小和步长对 *F*1 得分的影响. 如图 5 所示, 滑动窗口检索增强生成系统在 MultiFieldQA-zh、Qasper、MultiFieldQA-en、NarrativeQA 等多个数据集中使用 1 024 的窗口大小和 3 的步长时达到了最高的 *F*1 得分. 实验表明, 最小(512)和最大(2 048)的窗口大小都未能提供最佳性能, 相反, 多个数据集上的最佳窗口大小均为 1 024. 不同的参数会对模型造成较大的影响, 例如 MultiFieldQA-zh 数据集下, *F*1 得分最高能达到 53.36%, 但相同策略中不同参数的 *F*1 得分最大相差能达到 7.88 个百分点. 这一发现表明, 在使用 RAG 解决 LLMs 幻觉问题时, 检索所用块的大小是一个关键参数, 使用过小的块可能导致上下文缺失, 而过大的块可能引入无关信息, 这两者都会对生成响应的事实准确性产生负面影响.

实验结果表明, 最优参数配置的存在依赖于多个因素, 包括数据集特性、算法策略和性能指标. 例如在 DDS 策略中, 中等窗口为 1 024 和步长为 3 时在多数数据集上表现最佳, 但在长度较长的数据集(NarrativeQA, 平均长度 18 409 个令牌)中表现不佳, 其所有参数组合下 *F*1 得分为 11.68~15.22.

2)单查询与多查询的对比. 本研究数据集涵盖单文档问答和多文档问答, 确保了实验的全面性和结果的广泛适用性, 特别是使用了诸如 NarrativeQA



(a) MultiFieldQA-zh数据集　(b) MultiFieldQA-en数据集

(c) Qasper数据集　(d) NarrativeQA数据集

(e) HotpotQA数据集

Fig. 5　*F*1 scores for various window sizes and step lengths in multiple and single query contexts

图 5　不同窗口大小以及各种步长在多重查询和单一查询情况下的 *F*1 得分

和 HotpotQA 等复杂数据集，对系统的理解和生成能力提出了更高的要求. 在多查询实验中，当参数设置为 1024_3 时，模型在 MultiFieldQA、Qasper 和 NarrativeQA 数据集上表现最佳，但在 HotpotQA 数据集上效果最差.这一结果表明，对于复杂数据集（尤其是涉及多文档查询的任务），最优参数的选择与单文档数据集存在显著差异.实验结果显示，多重查询方法在多数数据集上整体优于单一查询方法，这是因为多重查询方法能够更加全面地涵盖查询的不同方面，从而提升检索文档的多样性和相关性，从而提高生成答案的准确性. 然而，在 NarrativeQA 和 HotpotQA 数据集中，DDS 方法在 1 024 和 2 048 窗口大小下，单一查询方法表现优于多重查询方法，这表明对于这些复杂数据集，即使采用单一查询，DDS 方法也能高度有效，体现了精心配置的单一查询有时也能提供更优的结果.

5.2　滑动窗口检索增强生成系统的优势与局限性

1）优势. 滑动窗口检索增强生成系统通过动态

调整窗口大小和步长,有效保持了文本的语义完整性和结构连贯性,显著提升了信息检索的准确性和回答的相关性. 该系统展现了出色的适应性,能够处理各种数据集和查询类型,显示了其广泛的应用潜力. 与固定窗口的系统如 Facebook AI 的 RAG[37] 和 Google 的 REALM[38] 相比, Facebook AI 的 RAG 在自然问题数据集上的 $F1$ 得分为 44.5%, REALM 的 $F1$ 得分为 40.5%, 而本项目的滑动窗口检索增强生成系统在同一数据集上达到了 49.8% 的 $F1$ 得分. 具备更高准确率优势的主要原因在于减少了信息丢失,并能够更好地应对实时更新的数据需求. 此外,系统通过引入多查询处理技术从而生成多个查询变体,有效增加了获取最相关文档的机会,提高了复杂查询的处理能力. 这种结合检索与生成的方法为检索增强生成系统的发展开辟了新的可能性,特别是在保持高检索精度和生成回答的准确性方面表现突出.

2)局限性. 滑动窗口检索增强生成系统由于滑动窗口策略和多查询方法,由此增加的计算成本和存储开销在处理大规模数据时可能成为瓶颈.

从时间复杂度的角度看,基线 RCS 在最坏情况下,即需要频繁重新组合或分割大量元素时,时间复杂度达到 $O(n^2)$, 其中 n 是原始文本长度; FFS 的时间复杂度为 $O(n/s\times w)$, 其中 s 是步长, w 是窗口的长度; DFS 和 DDS 策略的时间复杂度分为使用分割符分割文本和创建块 2 部分,两者的分割文本的时间复杂度都是 $O(m\times n)$, 其中 m 是分割符的数量, n 是文本长度, DFS 创建块的时间复杂度为 $O(k\times n)$, 其中 k 为分割块数量,所以 DFS 的总体时间复杂度为 $O((m+k)\times n)$, DDS 创建块的时间复杂度为 $O(n^2)$, 因此 DDS 的时间复杂度为 $O(m\times n^2)$. 综上,基于滑动窗口的分割方法中 FFS 的优化策略可以选择合适的窗口长度和步长,而 DFS 和 DDS 的优化策略可以从提前计算分割文本和缓存分割点方面考虑.

从存储成本的角度来看,关键因素包括块的数量和每个块的大小. DFS 和 FFS 策略的存储开销主要取决于窗口的大小,其中较大的窗口意味着更高的内存消耗. DDS 策略在块数量多或步长小的情况下存储耗费较大,因为每个块可能被多次读取和处理,增加了对缓存和额外数据存储的需求. 存储成本优化可以考虑合理选择步长和块的大小以平衡准确性和存储需求.

综合考虑计算和存储开销, FFS 策略由于其块大小和步长的固定性,提供了最佳的计算和资源使用效率,但若是在精度需求较高的场景,则可以使用 DDS 策略. 此外,需要进一步进行参数优化,例如为不同的数据集和任务调整窗口大小和步长设置,这增加了系统部署和应用的复杂性.

5.3 未来研究方向

基于当前的研究结果,未来对滑动窗口检索增强生成系统的探索和改进可以集中在 3 个方面. 首先可以开发自适应参数优化方法,自动根据不同任务和数据集调整窗口大小和步长,以实现最佳性能. 其次,将滑动窗口检索增强生成系统与知识图谱检索和领域自适应预训练等技术结合,可以进一步提高生成答案的准确性和可靠性. 最后,通过探索更高效的算法和硬件加速技术来优化计算效率,可以降低滑动窗口检索增强生成系统的计算成本[39],使其在实际应用中更具可行性.

6 结 论

滑动窗口检索增强生成系统为提高 LLMs 输出的事实准确性和可靠性提供了一种富有前景的方法. 通过在检索过程的索引阶段引入滑动窗口机制,系统解决了与固定大小上下文窗口和静态检索方法相关的常见限制.

项目对滑动窗口检索增强生成系统进行了全面评估,特别关注了 DDS 策略,证明了其在多个数据集和查询类型中的有效性. 实验结果表明,滑动窗口检索增强生成系统,尤其是使用 DDS 策略时,在 $F1$ 得分方面表现出色,结果优于 RCS 等基线方法.

尽管滑动窗口检索增强生成系统展示了其有效性,但也存在一些局限性,包括增加了计算成本和参数优化的需求. 然而,其在上下文保留和强适应性方面的优势超越了这些限制,使得滑动窗口检索增强生成系统成为解决 LLMs 输出中事实缺乏准确性和可靠性问题的解决方案.

通过进一步的研究和优化,滑动窗口检索增强生成系统有望在实际应用中发挥更大的作用,提供更准确和更可靠的答案生成能力. 未来的研究方向包括开发自适应参数优化方法、结合知识图谱检索和领域自适应预训练技术,以及优化计算效率以降低计算成本. 通过这些改进,滑动窗口检索增强生成系统将能够更好地满足各种复杂任务的需求,进一步提升 LLMs 生成输出的质量.

致谢: 本文的研究工作是在开放原子开源大赛

（https://competition.atomgit.com/）——基于 BigDL-LLM 的轻量级大语言模型应用开发创新赛的框架下完成的，得到了开放原子开源基金会及赛项共建方英特尔（中国）有限公司的大力支持.

作者贡献声明：毕枫林负责方案及算法设计；张豈明、张嘉睿负责实验及论文撰写；王衍童负责实验数据分析；陈阳负责论文修订；张琰彬负责对研究提出意见并指导论文修改；王伟、周烜参与论文修改及定稿讨论.

参 考 文 献

[1] Zhao W X, Zhou Kun, Li Junyi, et al. A survey of large language models[J]. arXiv preprint, arXiv: 2303.18223, 2023

[2] Taffa A T, Usbeck R. Leveraging LLMs in scholarly knowledge graph question answering[J]. arXiv preprint, arXiv: 2311.09841, 2023

[3] Wang Chaojie, Xu Yishi, Peng Zhong, et al. Keqing: Knowledge-based question answering is a nature chain-of-thought mentor of LLM[J]. arXiv preprint, arXiv: 2401.00426, 2023

[4] Rawte V, Sheth A, Das A. A survey of hallucination in large foundation models[J]. arXiv preprint, arXiv: 2309.05922, 2023

[5] Zhang Yue, Li Yafu, Cui Leyang, et al. Siren's song in the AI ocean: A survey on hallucination in large language models[J]. arXiv preprint, arXiv: 2309.01219, 2023

[6] Zhu Zhiying, Sun Zhiqing, Yang Yiming. HaluEval-Wild: Evaluating hallucinations of language models in the wild[J]. arXiv preprint, arXiv: 2403.04307, 2024

[7] Lewis P, Perez E, Piktus A, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks[J]. Advances in Neural Information Processing Systems, 2020, 33: 9459−9474

[8] Finardi P, Avila L, Castaldoni R, et al. The chronicles of RAG: The retriever, the chunk and the generator[J]. arXiv preprint, arXiv: 2401.07883, 2024

[9] Tian Yonglin, Wang Xingxia, Wang Yutong, et al. RAG-PHI: Retrieval-enhanced generation-driven parallel humans and parallel intelligence[J]. Journal of Intelligent Science and Technology, 2024, 6(1): 41−51 (in Chinese)
（田永林, 王兴霞, 王雨桐, 等. RAG-PHI: 检索增强生成驱动的平行人与平行智能[J]. 智能科学与技术学报, 2024, 6(1): 41−51）

[10] Gao Yunfan, Xiong Yun, Gao Xinyu, et al. Retrieval-augmented generation for large language models: A survey[J]. arXiv preprint, arXiv: 2312.10997, 2023

[11] Tian Xuan, Wu Zhichao. Review of knowledge base question answering based on information retrieval[J]. Journal of Computer Research and Development, 2025, 62(2): 314−335 (in Chinese)
（田萱, 吴志超. 基于信息检索的知识库问答综述[J]. 计算机研究与发展, 2025, 62(2): 314−335）

[12] Anantha R, Bethi T, Vodianik D, et al. Context tuning for retrieval augmented generation[J]. arXiv preprint, arXiv: 2312.05708, 2023

[13] Ortiz L J, Olaya A G, Borrajo D. A dynamic sliding window approach for activity recognition[C]//Proc of the 19th Int Conf on User Modeling, Adaption and Personalization (UMAP 2011). Berlin: Springer, 2011: 219−230

[14] Packer C, Fang V, Patil S G, et al. MemGPT: Towards LLMs as operating systems[J]. arXiv preprint, arXiv: 2310.08560, 2023

[15] Xu Peng, Ping Wei, Wu Xianchao, et al. Retrieval meets long context large language models[J]. arXiv preprint, arXiv: 2310.03025, 2023

[16] Han Bingtao, Liu Tao. Key technologies and applications of large models[J]. ZTE Communications, 2024, 30(2): 76−88 (in Chinese)
（韩炳涛, 刘涛. 大模型关键技术与应用[J]. 中兴通讯技术, 2024, 30(2): 76−88）

[17] Chen H, Xu Fangyuan, Arora S, et al. Understanding retrieval augmentation for long-form question answering[J]. arXiv preprint, arXiv: 2310.12150, 2023

[18] Chor B, Kushilevitz E, Goldreich O, et al. Private information retrieval[J]. Journal of the ACM, 1998, 45(6): 965−981

[19] Luo Ziyang, Xu Can, Zhao Pu, et al. Augmented large language models with parametric knowledge guiding[J]. arXiv preprint, arXiv: 2305.04757, 2023

[20] He Xiaoxin, Tian Yijun, Sun Yifei, et al. G-Retriever: Retrieval-augmented generation for textual graph understanding and question answering[J]. arXiv preprint, arXiv: 2402.07630, 2024

[21] Teja R. Evaluating the ideal chunk size for a RAG system using LlamaIndex[EB/OL]. [2024-04-01]. https://www.llamaindex.ai/blog/evaluating-the-ideal-chunk-size-for-a-rag-system-using-llamaindex-6207e5d3fec5

[22] LangChain. Recursively split by character[EB/OL]. [2024-05-01]. https://python.langchain.com/docs/modules/dataconnection/documenttransformers/recursivetextsplitter

[23] Dhuliawala S, Komeili M, Xu Jing, et al. Chain-of-verification reduces hallucination in large language models[J]. arXiv preprint, arXiv: 2309.11495, 2023

[24] Ma Xinbei, Gong Yeyun, He Pengcheng, et al. Query rewriting for retrieval-augmented large language models[J]. arXiv preprint, arXiv: 2305.14283, 2023

[25] Peng Wenjun, Li Guiyang, Jiang Yue, et al. Large language model based long-tail query rewriting in Taobao search[J]. arXiv preprint, arXiv: 2311.03758, 2023

[26] Zhang Jiarui, Zhang Qiming, Bi Fenglin, et al. Locally lightweight course teaching-assistant system based on IPEX-LLM[J]. Journal of East China Normal University: Natural Science, 2024(5): 162−172 (in Chinese)
（张嘉睿, 张豈明, 毕枫林, 等. 基于 IPEX-LLM 的本地轻量化课程教学智能辅助系统[J]. 华东师范大学学报: 自然科学版, 2024(5): 162−172）

[27] LangChain. MultiQueryRetriever documentation[EB/OL]. [2024-05-11]. https://python.langchain.com/v0.1/docs/modules/data_connection/retrievers/MultiQueryRetriever

[28] Hugging Face. Open LLM leaderboard[EB/OL]. [2024-05-01]. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

[29] Bai Yushi, Lv Xin, Zhang Jiajie, et al. LongBench: A bilingual, multitask benchmark for long context understanding[J]. arXiv preprint, arXiv: 2308.14508, 2023

[30] Shaham U, Segal E, Ivgi M, et al. Scrolls: Standardized comparison over long language sequences[J]. arXiv preprint, arXiv: 2201.03533, 2022

[31] Kočisk`y T, Schwarz J, Blunsom P, et al. The NarrativeQA reading

comprehension challenge[J]. Transactions of the Association for Computational Linguistics, 2018, 6: 317−328

[32] Dasigi P, Lo K, Beltagy I, et al. A dataset of information-seeking questions and answers anchored in research papers[J]. arXiv preprint, arXiv: 2105.03011, 2021

[33] Yang Zhilin, Qi Peng, Zhang Saizheng, et al. HotpotQA: A dataset for diverse, explainable multi-hop question answering[J]. arXiv preprint, arXiv: 1809.09600, 2018

[34] LangChain. RecursiveCharacterTextSplitter-LangChainAPI documentation[EB/OL]. [2024-05-01]. https://api.python.langchain.com/en/latest/character/langchain_text_splitters.character.RecursiveCharacterTextSplitter.html

[35] OpenAI. GPT−3.5 Turbo model documentation[EB/OL]. [2024-07-01]. https://platform.openai.com/docs/models/gpt-3-5-turbo

[36] Fu Yao. Challenges in deploying long-context transformers: A theoretical peak performance analysis[J]. arXiv preprint, arXiv: 2405.08944, 2024

[37] Lewis P, Perez E, Piktus A, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks[J]. Advances in Neural Information Processing Systems, 2020, 33: 9459−9474

[38] Guu K, Lee K, Tung Z, et al. REALM: Retrieval-augmented language model pre-training[J]. arXiv preprint, arXiv: 2002.08909, 2020

[39] Feng Yangyang, Wang Qing, Xie Wenhui, et al. From BERT to ChatGPT: Challenges and technological developments in storage systems for large model training[J]. Journal of Computer Research and Development, 2024, 61(4): 809−823 (in Chinese)
(冯杨洋, 汪庆, 谢旻晖, 等. 从 BERT 到 ChatGPT: 大模型训练中的存储系统挑战与技术发展[J]. 计算机研究与发展, 2024, 61(4): 809−823)



Bi Fenglin, born in 1993. PhD candidate. Student member of CCF. His main research interests include large language models, open source software supply chain and ecology, and software bots.

毕枫林, 1993 年生. 博士研究生. CCF 学生会员. 主要研究方向为大语言模型、开源软件供应链和生态、软件机器人.



Zhang Qiming, born in 1999. Master candidate. Student member of CCF. His main research interests include large language model applications and computational pedagogy.

张岂明, 1999 年生. 硕士研究生. CCF 学生会员. 主要研究方向为大语言模型应用、计算教育学.



Zhang Jiarui, born in 2000. Master candidate. His main research interests include large language models and computational pedagogy.

张嘉睿, 2000 年生. 硕士研究生. 主要研究方向为大语言模型、计算教育学.



Wang Yantong, born in 2000. Master candidate. His main reserach interests include large language models and open source software ecology.

王衍童, 2000 年生. 硕士研究生. 主要研究方向为大语言模型、开源软件生态学.



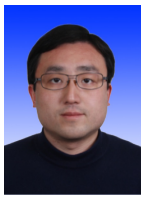
Chen Yang, born in 1981. PhD, professor, PhD supervisor. Senior member of ACM, IEEE and CCF. His main research interests include computer networks, social computing, and big data analytics.

陈　阳, 1981 年生. 博士, 教授, 博士生导师. ACM, IEEE, CCF 高级会员. 主要研究方向为计算机网络、社交计算、大数据分析.



Zhang Yanbin, born in 1983. Master, engineer, master supervisor. Member of CCF. Her main research interests include computational pedagogy and software engineering.

张琰彬, 1983 年生. 硕士, 工程师, 硕士生导师. CCF 会员. 主要研究方向为计算教育学、软件工程.



Wang Wei, born in 1979. PhD, professor, PhD supervisor. Member of CCF. His main research interests include open source measurement and digital ecosystems.

王　伟, 1979 年生. 博士, 教授, 博士生导师. CCF 会员. 主要研究方向为开源度量、数字生态系统.



Zhou Xuan, born in 1979. PhD, professor, PhD supervisor. Member of CCF. His main research interests include high performance database and information retrieval.

周　烜, 1979 年生. 博士, 教授, 博士生导师. CCF 会员. 主要研究方向为高性能数据库、信息检索.