基于动态规划的 RAG 语义感知分块方法

谢圣富[1,2] 农 静[1,2]
XIE Shengfu NONG Jing

摘 要 在检索增强生成（RAG）系统中，文档分块质量对后续检索准确性及生成表现具有重要影响。现有方法大多基于局部信息进行划分，因此在语义连贯性和长度控制之间难以取得理想平衡。为解决这一问题，文章提出一种结合动态规划的语义感知全局最优分块方法。该方法首先通过计算句子间的语义相似度构建矩阵，并进一步完成中心化和归一化处理；在此基础上，采用动态规划策略，以增强块内语义连贯性为优化目标，对分块边界进行全局求解。多领域中文数据集上的实验结果显示，所提出的方法在检索任务与生成任务中均优于传统分块方式。表明该方法通过全局优化有效减少了局部决策带来的局限，体现出其在 RAG 场景中的良好适用性与有效性。

关键词 检索增强生成；文档分块；动态规划；语义相似度；全局优化

doi：10.3969/j.issn.1672-9528.2026.04.031

0 引言

随着人工智能技术的快速发展，大语言模型（large language model, LLM）在自然语言处理任务中展现出较强的文本理解和生成能力。然而，在实际应用中，LLMs 仍面临知识更新滞后与生成内容准确性不足的问题[1]。受限于训练语料的实效性，模型通常难以及时获取最新的领域知识和实时信息。

通过微调来更新大语言模型往往成本高、周期长，尤其当处理大规模文本语料时[2]。检索增强生成（retrieval-augmented generation, RAG）通过将信息检索与文本生成相结合，通过引入外部知识库以补充模型上下文，为解决上述问题提供了有效的途径[3]。具体而言，RAG 系统首先根据用户查询，从外部知识库中检索相关文档片段，然后将检索到的信息与原始查询一并输入到 LLM 中，生成更为准确且上下文相关的回答。

检索和生成构成了 RAG 系统中的两个核心组件[4]。检索模块负责从大规模文档集合中快速、准确地定位与查询相关的文本片段，生成模块则基于检索结果产生最终的回答。两个模块协同工作决定了整个系统的性能表现。然而，现有研究主要关注检索算法的优化和生成模型的改进，而对连接两个模块的关键环节——文档分块策略的研究相对较少。

文档分块作为 RAG 系统的基础预处理步骤，其质量直接影响检索效果和生成性能。合理的分块策略可确保每个分块包含完整的语义信息，防止关键内容被截断，为检索模块提供高质量的候选片段。良好的分块还可控制输入到生成模型的上下文长度，在信息完整性与计算效率之间取得平衡[5]。相反，不当的分块策略可能导致语义信息的丢失或冗余，影响最终生成内容的质量。

1 相关工作

优化 RAG 的关键因素之一是文档的分块策略。在传统分块方法中，固定大小分块将文档切为等长片段，如将百科条目按约 100 个词划分为独立段落[5]。该方法实现起来简单直接，但往往无法捕捉文本的语义或结构上的细微差别。滑动窗口分块在固定长度的基础上引入重叠，以缓解跨边界切分造成的信息断裂[6]，但重叠会增加冗余与重复匹配，而过多且重复的片段会稀释关键信息的显著性。递归分块根据换行符或空格等标记来分割文本，在文档具有清晰格式编排时可能较为有效[7]。

近年来，语义分块引起了广泛关注，该方法基于语义相似性对文档进行分割，旨在提供语义完整且上下文连贯的片段。Kamradt 提出以句 / 子句为基本单元，依据相邻片段在嵌入空间的相似度起伏确定断点，取相似度谷值作为话题转折信号的分块方法[8]。但其本质在断点判断采取局部贪心策略，缺乏全局一致性约束与最优性保证，在主题缓慢迁移或跨段

1. 贵州师范大学网络空间安全学院 贵州贵阳 550001
2. 贵州师范大学贵州省先进计算省重点实验室 贵州贵阳 550001

[基金项目] 贵州省科技计划项目“基于组工业务的数据标准化开放协同底座及智能分析技术研究与应用”（黔科合支撑〔2023〕重点 001）

照应语境，容易把原本应连在一起的语义单元拆散。Zhang[9]等针对文档分割任务提出了基于序列模型的分块方法，采用自适应滑动窗口机制处理文本序列，通过序列建模技术预测潜在的分割边界点。该方法通过滑动窗口逐步扫描文档并进行边界判断，虽然在计算效率上有所提升，但每个分割决策仅依赖窗口内的有限信息，难以统筹考虑整个文档的最优分块方案。

相较于传统分块方式，语义分块虽然在提升文本语义连贯性方面取得了一定效果，但在 RAG 应用场景下仍然存在改进空间。其主要问题体现在边界划分过程缺乏全局视角。现有多数研究通常将分块边界的确定看作局部断点分类问题，因此在长度限制与窗口条件共同作用下，往往难以得到面向整篇文档的最优划分结果。

为解决上述问题，本文借鉴动态规划中的全局优化思想 [10]，不再仅依据局部信息确定分块边界，而是将边界选择统一纳入整体目标函数中进行求解。具体而言，在满足长度约束的前提下，以片段内部句子两两余弦相似度之和作为优化依据，对整篇文档的分块方案进行搜索。通过这种方式，可以从整体上增强块内语义的一致性，同时减轻跨块信息干扰和内容重复带来的影响。本文主要开展了以下两方面工作：

（1）提出了一种基于语义相似度约束的全局优化分块方法。该方法首先利用句子或子句级嵌入构建相似度矩阵，并进行中心化和规范化处理；然后在长度约束的情况下，以区间语义一致性作为目标，采用动态规划求解文档的整体最优划分。

（2）在检索增强生成任务中，将所提出的方法与固定分块、滑动窗口分块和典型语义分块进行对比，分析其在检索和生成两个阶段的实际表现。

2 基于动态规划的语义感知分块方法

2.1 概述

考虑到现有分块方法在边界选择上往往依赖局部信息、容易导致整体划分结果不够理想，本文提出一种结合动态规划的语义感知分块方法。该方法将文档分块看作一个全局约束下的优化工程，而不是局限于对单个边界位置进行逐步判断。通过动态规划，可以在满足约束条件的前提下，对不同候选分块方案进行整体比较，从而获得语义连贯性更强的划分结果。相比常见的局部决策方法，这种方式更有利于减少局部贪心带来的影响，使文档分块结果在整体上更加合理。

2.2 语义相似度矩阵构建算法

为了描述文本中各最小语义单元之间的语义关系，本文构建语义相似度矩阵。设 $S=\{s_1, s_2,\cdots, s_n\}$ 为文本块中的句子集合，其中 n 是文本块中句子的总数，s_i 为第 i 个句子。

首先，通过递归分块器将原始文本分解为最小语义单元。对于每个句子 s_i，使用嵌入函数 ε 获得其向量表示 $\boldsymbol{e}_i=\varepsilon(s_i)$，其中 $\boldsymbol{e}_i\in\mathbf{R}^d$，$d$ 为嵌入向量的维度。

为提高计算效率，采用批量处理方式计算嵌入向量。设批量大小为 B，对于 $N=\lceil n/B\rceil$ 个批次，第 k 个批次包含句子 $\{s_{(k-1)B+1}, s_{(k-1)B+2},\ldots, s_{\min(kB,n)}\}$。

对嵌入矩阵 $\boldsymbol{E}$ 进行归一化处理，对每个嵌入向量 $\boldsymbol{e}_i$ 进行单位化处理：

$$\tilde{\boldsymbol{e}}_i=\frac{\boldsymbol{e}_i}{|\boldsymbol{e}_i|_2+\epsilon} \tag{1}$$

式中：$\boldsymbol{E}=[\boldsymbol{e}_1, \boldsymbol{e}_2,\cdots, \boldsymbol{e}_n]^{\mathrm{T}}$ 为嵌入矩阵；$\epsilon=1\times10^{-12}$ 为防止除零的小常数。

归一化后的嵌入矩阵为 $\tilde{\boldsymbol{E}}=[\tilde{\boldsymbol{e}}_1,\tilde{\boldsymbol{e}}_2,\ldots,\tilde{\boldsymbol{e}}_n]^{\mathrm{T}}$。语义相似度矩阵定义为：

$$\boldsymbol{M}=\tilde{\boldsymbol{E}}\tilde{\boldsymbol{E}}^{\mathrm{T}} \tag{2}$$

矩阵元素 M_{ij} 表示句子 s_i 与 s_j 之间的余弦相似度：

$$M_{ij}=\cos(\boldsymbol{s}_i,\boldsymbol{s}_j)=\frac{\boldsymbol{s}_i\cdot\boldsymbol{s}_j}{|\boldsymbol{s}_i|_2|\boldsymbol{s}_j|_2} \tag{3}$$

算法 1：语义相似度矩阵构建

输入：

句子集合 $S=\{s_1, s_2,\cdots, s_n\}$；

嵌入函数 ε；

批量大小 B；

输出：

语义相似度矩阵 $\boldsymbol{M}$;

算法步骤：

1. 初始化嵌入矩阵 $\boldsymbol{E}\leftarrow0$;

2.for $i\leftarrow1$ to $\lceil n/B\rceil$ do

3. 获取批次句子 batch$\leftarrow\{s_{(i-1)\boldsymbol{E}+1},\ldots,s_{\min(i,\boldsymbol{E})}\}$;

4. 计算批次嵌入 $\boldsymbol{E}_b\leftarrow\varepsilon$(batch);

5. 拼接嵌入矩阵$\boldsymbol{E}\leftarrow[\boldsymbol{E};\boldsymbol{E}_b]$;

6.end for

7.for i--1 to n do

8. 计算 L_2 范数 $|\boldsymbol{e}_i|\leftarrow|\boldsymbol{e}_i|_2$;

9. $\tilde{\boldsymbol{e}}_i\leftarrow\boldsymbol{e}_i/(|\boldsymbol{e}_i|+\epsilon)$;

10.end for

11. 归一化嵌入矩阵$\tilde{\boldsymbol{E}}\leftarrow[\tilde{\boldsymbol{e}}_1,\tilde{\boldsymbol{e}}_2,\ldots,\tilde{\boldsymbol{e}}_n]^{\mathrm{T}}$;

12. 计算相似度矩阵 $\boldsymbol{M}\leftarrow\tilde{\boldsymbol{E}}\tilde{\boldsymbol{E}}^{\mathrm{T}}$;

13.return $\boldsymbol{M}$;

2.3 动态规划最优分段算法

基于上述构建的语义相似度矩阵，采用动态规划算法寻找全局最优分块边界。该算法的目标是在满足分块大小约束的条件下，最大化分块内部的语义连贯性。

设 $C = \{c_1, c_2, \cdots, c_k\}$ 为候选分块集合，其中 $c_j=(a_j, b_j)$ 表示第 j 个分块的起始和结束位置。定义分块内部语义连贯性奖励函数为：

$$R(a,b) = \sum_{i=a}^{b}\sum_{j=a}^{b} M_{ij} \quad (4)$$

为避免矩阵数值分布影响优化效果，对相似度矩阵进行中心化处理：

$$\boldsymbol{M} = \boldsymbol{M} - \mu \quad (5)$$

其中 μ 为矩阵上三角部分（不包含对角线）的均值：

$$\mu = \frac{2}{n(n-1)}\sum_{i<j} M_{ij} \quad (6)$$

同时，将对角线元素设为 0 以避免自相似性的影响：

$$M_i = 0, \forall i \in \{1,2,\ldots,n\} \quad (7)$$

经过上述预处理后，奖励函数实际计算的是分块内所有句子对之间的相对语义相似度之和，突出了语义相关性的差异化特征。

设 dp[i] 表示对前 i+1 个句子进行最优分段所能获得的最大奖励值；seg[i] 表示第 i 个位置对应的最优分块起始位置。动态规划递推关系为：

$$\mathrm{dp}[i] = \max_{1\le k\le \min(L,i+1)} \{R(i-k+1,i) + \mathrm{dp}[i-k]\} \quad (8)$$

式中：L 为最大分块大小约束，dp[−1] = 0。

算法 2：动态规划最优分段

输入：语义相似度矩阵 $\boldsymbol{M}$；最大分块大小 L；

输出：最优分块边界集合 C；

算法步骤：

1. 计算矩阵均值$\mu \leftarrow \frac{2}{n(n-1)}\sum_{i<j} M_{ij}$;

2. 中心化矩阵 $\boldsymbol{M}\leftarrow\boldsymbol{M}-\mu$；

3.for i=−1 to n do

4.　设置对角线元素 $M_{ii}\leftarrow 0$；

5.end for

6. 初始化 dp←[0,0,⋯,0], seg←[0,0,⋯,0]；

7.for i from 0 to n−1 do

8.　for k=−1 to min(L, i+1) do

9.　计算奖励 $r\leftarrow(i-k+1, i)$；

10.　ar←r；

11.　if $i-k \ge 0$ then

12.　　计算调整奖励 ar←ar+dg[$i-k$]；

13.　end if

14.　if ar > dp[i] then

15.　　dp[i]←ar；

16.　　seg[i]← $i-k$+1；

17.　end if

18.　end for

19.end for

20.C← 回溯构建分块边界集合；

21.return C；

3 实验

3.1 实验环境

项目使用 BAAI/bge-m3 嵌入模型 [11] 将文本转换为 1024 维语义向量，利用 FAISS 向量库实现高效的相似性检索，并通过近似最近邻搜索策略提升检索速度和效果。

选用参数规模约 320 亿的 DeepSeek-R1-Distill-Qwen-32B 作为底座大语言模型，为确保生成的响应精确且不重复，推理时采用贪婪解码，将温度设置为 0，并将重复惩罚系数调整为 1.1。

整个流程基于 RAG 框架进行，所使用的提示词如下：

基于上下文内容回答下列问题：

上下文：{context}

问题：{question}

3.2 数据集构建

为全面评估本研究所提出的分块方法在跨领域场景下的泛化能力，本研究收集并构建了一个多领域、多类型的测试数据集。数据集构成如图 1 所示。

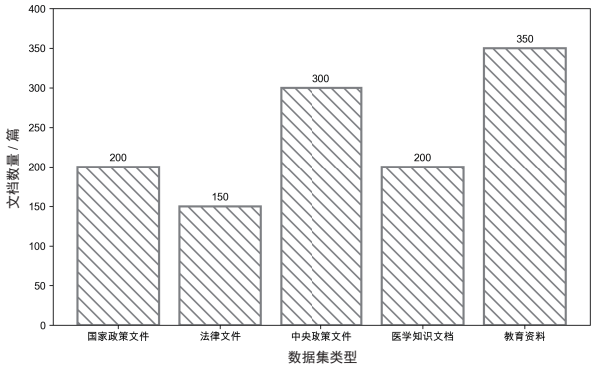

图 1 数据集文档数量分布图

在数据选取阶段，本文在每个文档类别中随机抽取 100 篇文本，共得到 500 篇原始文档。各类别的抽样数量保持一致，主要是为了避免测试集在领域分布上出现明显偏向，使后续实验结果更具可比性。在问答数据构建过程中，本文采用了人工编写与模型生成相结合的方式。人工部分从 500 篇文档中进一步选取 100 篇作为标注对象，围绕文档内容整理得到 500 条质量较高的 < 问题，答案 > 对。这部分数据主要用于保证测试集中的问答样本具有较好的准确性和可读性。除人工构建外，本文还借助通义千问（Qwen-Plus）生成候选问答对，以扩大数据规模。生成过程中使用预设提示词约束输出格式与内容，要求模型依据给定中文文本生成与事实内容直接对

应的问题，并同时给出支撑作答的参考文本。具体提示如下：

请根据提供的中文文本，生成准确且符合专业要求的问答对，并以 JSON 格式输出结果，结果中包含以下两个字段：

— 'question': 与文本事实直接对应的问题，并确保该问题只能依据提供的参考答案进行回答；

— 'references': 能够支撑该问题答案的文本内容。

考虑到模型直接生成的样本质量并不完全稳定，本文在生成完成后又进行了多轮筛选。首先，利用文本嵌入计算问题与参考答案之间的余弦相似度，去除相关性不足的样本；其次，对生成问题进行去重，若两条样本在语义上过于接近，则仅保留其中质量更高的一条；最后，再由人工逐条检查，对表述不准确、语义不清或存在事实偏差的样本进行修正或删除。

经过上述处理，最终保留模型生成的 2 500 条高质量问答对。随后，本文将人工构建与模型生成的数据合并，并按照各领域等比例随机抽取 1 600 条样本，构成最终测试所使用的问答集。

3.3 评估指标

为了更全面地观察不同分块方法在 RAG 场景中的实际效果，本文分别从检索阶段和生成阶段进行评价。

在检索性能方面，本文采用 Precision、Recall 和 F_1 指标 [11] 作为主要指标；对于生成阶段，本文使用 ROUGE-L[12] 和 Accuracy 指标进行综合评估。

3.3.1 检索指标

精确率（Precision）：表示在系统返回的文档中，真正与问题相关的文档所占比例，用来反映检索结果的准确性，计算公式为：

$$\text{Precision} = \frac{\text{相关文档} \cap \text{检索文档}}{\text{检索文档}} \tag{9}$$

召回率（Recall）：表示所有相关文档中被成功检索出的比例，用于描述系统对相关信息的覆盖能力，其定义为：

$$\text{Recall} = \frac{|\text{相关文档} \cap \text{检索文档}|}{|\text{相关文档}|} \tag{10}$$

F_1 值：综合考虑了精确率和召回率，是二者的调和平均数，能够更全面地反映检索性能，其公式为：

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

3.3.2 生成指标

ROUGE-L：基于最长公共子序列（LCS）进行计算，能够反映生成文本与参考文本之间的相似程度，定义为：

$$\text{ROUGE-L} = \frac{\text{LCS}(X, Y)}{|Y|} \tag{12}$$

式中：X 为模型生成文本；Y 为参考文本；LCS(X, Y) 为两者的最长公共子序列长度。

准确率（Accuracy）：用于衡量生成答案与标准答案是否一致，从整体上反映模型回答的正确率，其定义为：

$$\text{Accuracy} = \frac{\text{正确回答的问题数}}{\text{总问题数}} \tag{13}$$

3.4 实验基线介绍

为验证本文方法的有效性，实验选取以下三种分块方法作为对比基线：

（1）固定大小分块 [5]（fixed-size window segmentation）：按照预先设定的统一长度，对文档进行顺序切分，得到若干相邻且互不重叠的等长文本片段，并将这些片段作为后续索引与检索的基本单元。

（2）滑动窗口分块 [6]（sliding-window segmentation）：在固定大小分块的基础上引入窗口重叠，通过滑动方式构造文本块，以减少语义相关内容因边界切分而被分开的情况。

（3）Kamradt 语义分块 [8]（kamradt semantic chunking）：利用句子嵌入向量之间的语义相似度进行文本分段，通过计算相邻句子的余弦相似度，在语义变化较大的位置确定边界，从而形成语义更加连贯的文本块。

3.5 实验设置

3.5.1 检索数量选择

为了研究每次输入原始查询后，LLM 从知识库中检索文档的数量 top-K 对问答系统性能所产生的影响，本文在国家政策、法律文件、中央政策文件、医学知识以及教育资料数据集上，随机选取了每个数据集的 200 条问答对展开测试。

测试结果如图 2 所示，当 top-K 取值过大时，检索过程中可能会出现冗余信息，从而降低问答系统的性能；而当 top-K 取值过小时，可能导致检索的文档覆盖不全面。在本文的数据集上，多个数据集在 top-K 设置为 3 或 4 时，问答系统的 F_1 指标表现最佳。基于这些结果，后续实验选择 top-K 为 3 作为最终设置进行进一步研究。

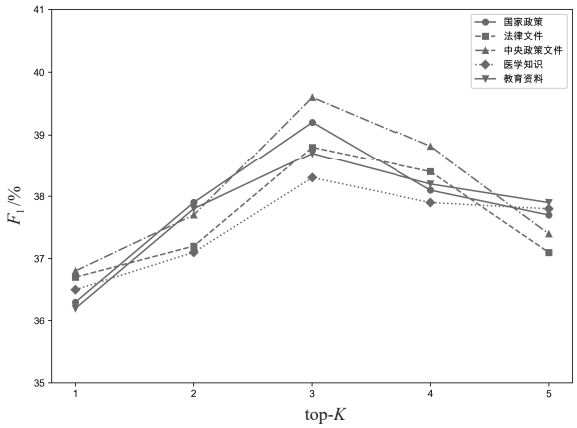

图 2 不同文档类型在不同 top-K 下的 F_1 表现

3.5.2 分块大小设置

本文通过测试不同的分块大小来评估我们的分块策略。在大多数相关研究中，4 096 token 被认为是短文本的典型长度[13]，尤其是在 GPT-3.5 之后的大语言模型中，4 096 是常用的最大输入长度[14-15]。因此本文以 4 096 的 1/8 及其变体作为分块大小的基准，选取了两档分块大小：256 token、512 token，并按如下方式设置各分块策略。

固定大小分块：直接按 256 token、512 token 分文档，块与块之间无重叠。

滑动窗口分块：在与固定大小分块相同的分块长度上，分别设置分块大小的 1/2、1/4 作为重叠块，以获得 50% 与 75% 区块重叠度。

Kamradt 语义分块：设置相似度阈值为 0.3、0.5，同时限制分块大小在 256 token、512 token 范围内，当语义边界产生的分块超出 ±50% 范围时采用强制切分。

本文方法：设置最大分块大小为 256 token、512 token，最小分块单元固定为 50 token。此外，最大分块长度约束 K 分别设置为⌊256/50⌋=5 和⌊512/50⌋=5（以句子数量计），以控制分块的粒度。

3.6 实验结果与分析

本文使用基于动态规划的语义感知分块和基线分块构建问答系统，通过对比不同分块大小的结果如表 1~2 所示，其中最优结果用加粗标出，次优结果用下划线标出。

表 1 256 token 分块大小下的性能对比

单位：%

Method	Precision	Recall	F_1	ROUGE-L	Accuracy
Fixed-size	35.1	37.4	36.2	32.8	38.5
Sliding-50%	38.8	40.2	39.5	35.6	41.2
Sliding-75%	39.4	42.1	40.7	37.1	42.8
Kamradt-0.3	40.5	43.6	42.0	38.4	44.1
Kamradt-0.5	<u>42.1</u>	<u>44.5</u>	<u>43.3</u>	<u>39.8</u>	<u>45.8</u>
本文方法	**50.7**	**51.9**	**51.3**	**47.2**	**54.2**

表 2 512 token 分块大小下的性能对比

单位：%

Method	Precision	Recall	F_1	ROUGE-L	Accuracy
Fixed-size	38.3	40.1	39.2	35.4	41.3
Sliding-50%	40.3	42.1	41.2	37.8	43.7
Sliding-75%	41.5	44.4	42.9	39.3	45.6
Kamradt-0.3	43.8	47.3	45.5	41.7	48.2
Kamradt-0.5	<u>44.1</u>	<u>47.8</u>	<u>45.9</u>	<u>42.3</u>	<u>49.1</u>
本文方法	**52.9**	**53.0**	**53.0**	**48.9**	**56.8**

从实验结果可以看出，本文方法在 256 token 和 512 token 两种设置下都取得了最好的结果，说明其在不同分块长度条件下都具有较好的稳定性。具体来看，在 256 token 条件下，本文方法的 F_1 为 51.3%，明显高于其他基线方法；在 512 token 条件下，F_1 和 Accuracy 分别达到 53.0% 和 56.8%，仍然保持领先。整体上看，本文方法不仅在检索指标上更优，在生成指标上也展现出更好的表现。

实验中，无论是在较短分块还是较长分块设置下，本文方法的 ROUGE-L 都是所有方法中最高的。这说明采用本文方法后，模型生成的答案在内容组织和表达结果上更接近标准答案。其原因在于，本文方法在分块过程中并非依赖单一步骤的局部判断，而是从整体出发对边界进行统一优化，因此能够更好地保持语义单元的完整性。

相比之下，固定大小分块方法的效果最弱。原因在于该方法只根据长度进行切分，没有考虑文本本身的语义结构，因此容易造成上下文被强行截断。特别是在较短分块条件下，文本能够提供的信息有限，容易影响生成答案的完整性；而当分块长度增加后，虽然信息量有所提升，但无关内容也会相应增多，进而干扰模型对重点信息的识别。

滑动窗口分块在结果上优于固定大小分块，说明重叠策略在一定程度上有助于缓解边界问题。不过，从实验结果看，随着重叠比例由 50% 增加到 75%，整体性能并没有出现特别明显的提升。这表明重叠窗口虽然可以增加上下文覆盖，但对语义断裂问题的解决仍然有限。

Kamradt 语义分块方法在各项指标上整体优于固定分块和滑动窗口方法，这说明基于语义相似度进行切分比单纯依赖长度规则更合理。但由于其边界判断主要基于局部相似度变化，因此仍难以保证整篇文档层面的最优划分。本文方法通过动态规划引入全局优化思想，在满足长度约束的前提下实现了更合理的边界选择，因此在检索与生成两个阶段都取得了更好的结果。

本文提出的动态规范语义感知分块策略，能够在分块粒度和语义完整性之间取得更好的平衡，有效提升 RAG 系统的整体性能。实验结果验证了该方法的可行性，也说明从全局优化角度改进文档分块策略具有较高的应用价值。

4 结论

本文围绕 RAG 系统中文档分块过程中容易受到局部决策影响、难以获得整体最优结果的问题，提出了一种基于动态规划的语义感知全局优化分块方法。该方法以句子间语义相似度矩阵为基础，在完成中心化和归一化处理后，利用动态规划对分块边界进行全局求解，使块内语义连贯性尽可

能提高，从而获得更合理的文档划分结果。实验结果表明，本文方法在检索性能和生成质量等多项指标上均优于对比方法，说明从全局角度进行分块决策有助于更好地保留文本语义完整性，并进一步提升 RAG 系统的整体表现。

在后续研究中，还可以从两个方面继续展开。一方面，可考虑引入自适应分块机制，根据文档类型、文本结构以及查询需求的差异，动态调整分块长度约束和相关参数，以提高方法在不同任务场景下的适用性；另一方面，可将该方法进一步拓展到跨语言或多语言文档处理场景中，以验证其在多语言 RAG 系统中的应用效果。

参考文献：

[1]JI Z W, LEE N, FRIESKE R, et al. Survey of hallucination in natural language generation[J]. ACM computing surveys, 2023, 55(12): 1-38.

[2]JI S X, PAN S R, CAMBRIA E, et al. A survey on knowledge graphs: representation, acquisition, and applications[J]. IEEE transactions on neural networks and learning systems, 2021, 33(2): 494-514.

[3]LEWIS P, PEREZ E, PIKTUS A, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks[C]//NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems.NewYork:ACM,2020, 793: 9459-9474.

[4]GAO Y F, XIONG Y, GAO X Y, et al. Retrieval-augmented generation for large language models: a survey[EB/OL]. (2024-03-27)[2026-03-19].https://doi.org/10.48550/arXiv.2312.10997.

[5]KARPUKHIN V, OGUZ B, MIN S, et al. Dense passage retrieval for open-domain question answering[C]//Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) .Brussels:ACL,2020: 6769-6781.

[6]LIU N F, LIN K, HEWITT J, et al. Lost in the middle: how language models use long contexts[J].Transactions of the association for computational linguistics,2024,12:157-173.

[7]YEPES A J, YOU Y, MILCZEK J, et al. Financial report chunking for effective retrieval augmented generation[EB/OL].(2024-03-16)[2026-03-19].https://doi.org/10.48550/arXiv.2402.05131.

[8]NGUYEN H T, NGUYEN T D, NGUYEN V H. Enhancing retrieval augmented generation with hierarchical text segmentation chunking[EB/OL].(2025-07-14)[2026-03-19].https://doi.org/10.48550/arXiv.2507.09935.

[9]ZHANG Q L, CHEN Q, LI Y L, et al. Sequence model with self-adaptive sliding window for efficient spoken document segmentation[EB/OL].(2021-10-09)[2026-03-19].https://doi.org/10.48550/arXiv.2107.09278.

[10]KEHAGIAS A, FRAGKOU P, PETRIDIS V. Linear text segmentation using a dynamic programming algorithm[C]//EACL'03: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics. NewYork: ACM, 2003:171-178.

[11]CHEN J Y, XIAO S T, ZHANG P T, et al. M3-Embedding: multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation[C]//Findings of the Association for Computational Linguistics: ACL 2024. Brussels:ACL, 2024: 2318-2335.

[12]POWERS D M W. Evaluation: from precision, recall and *F*-measure to ROC, informedness, markedness and correlation[EB/OL]. (2020-10-11)[2026-03-19].https://doi.org/10.48550/arXiv.2010.16061.

[13]LIN C Y. ROUGE: a package for automatic evaluation of summaries[C]//Text summarization branches out. Brussels: ACL, 2004: 74-81.

[14]FU Y. Challenges in deploying long-context transformers: a theoretical peak performance analysis[EB/OL].(2024-05-14)[2026-03-19].https://doi.org/10.48550/arXiv.2405.08944.

[15]OpenAI. GPT-3.5 Turbo model documentation[EB/OL]. [2026-03-19]. https://platform.openai.com/docs/models/gpt-3-5-turbo.

【作者简介】

谢圣富（2000—），男，四川资阳人，硕士，研究方向：人工智能。

农静（1973—），女，广西蒙山人，博士，高级工程师，研究方向：人工智能。

（收稿日期：2025-08-28 修回日期：2026-03-20）