面向工艺规范的树结构检索增强生成方法研究

姜禹辰[1]，王裴岩[1*]，冯煜博[2]，余卓[3]，纪贵阳[1]
[1]沈阳航空航天大学　计算机学院
[2]大连理工大学　计算机科学与技术学院
[3]上海飞机制造有限公司　航空制造技术研究所
{jiangyuchen2,jiguiyang}@stu.sau.edu.cn, wangpy@sau.edu.cn
argmax@126.com
yuzhuo@comac.cc

摘要

检索增强生成（Retrieval-Augmented Generation，RAG）是一种有效优化大语言模型在工艺规范问答任务中性能的方法。然而，基于固定文本长度分块的朴素RAG（Naive RAG）在构建工艺规范问答任务时表现不佳。主要原因在于工艺规范是一类复杂的技术文档，采用固定文本长度分块会丢失工艺规范段落层级之间的结构关系以及隐含的知识关联关系，导致输出结果质量下降。因此，本文提出了一种利用工艺规范篇章段落间隐含的树结构关系来构建RAG的方法，该方法有效解决了固定文本长度分块导致的段落之间的知识关联丢失问题。实验结果表明，树结构RAG在评价指标上优于朴素RAG，其中ACC平均提升3.81%，ROUGE-L提升3.28%，BLEU-4提升2.97%，验证了树结构RAG的有效性。

关键词： 检索增强生成；大语言模型；工艺规范；信息检索

A Tree-Structured Retrieval-Augmented Generation Method for Manufacturing Specifications Document

Yuchen Jiang[1], Peiyan Wang [1*], Yubo Feng [2], Zhuo Yu[3], Guiyang Ji[1]
[1]School of Computer Science, Shenyang Aerospace University
[2]School of Computer Science and Technology, Dalian University of Technology
[3]Aviation Manufacturing Technology Research Institute,
COMAC Shanghai Aircraft Manufacturing Co., Ltd.
{jiangyuchen2,jiguiyang}@stu.sau.edu.cn, wangpy@sau.edu.cn
argmax@126.com
yuzhuo@comac.cc

Abstract

Retrieval-Augmented Generation (RAG) is an effective method for optimizing the performance of large language models in question-answering tasks related to manufacturing specifications. However, the Naive RAG, which relies on fixed-length text chunking, performs poorly when applied to the construction of question-answering tasks for manufacturing specifications. The main reason is that manufacturing specifications are complex technical documents, and the use of fixed-length text chunking results in the loss of structural relationships between paragraphs as well as implicit knowledge associations, leading to a decline in the quality of the output. To address this issue, this paper proposes a method that leverages the implicit tree-structured relationships between sections and paragraphs in manufacturing specifications to construct RAG, effectively solving the problem of lost knowledge associations between paragraphs caused by fixed-length text chunking. Experimental results demonstrate that the tree-structured RAG outperforms the Naive RAG across evaluation metrics, with an average improvement of 3.81% in ACC, 3.28% in ROUGE-L, and 2.97% in BLEU-4, validating the effectiveness of the tree-structured RAG.

第二十四届中国计算语言学大会论文集，第143页-第155页，济南，中国，2025年8月11日至14日。

Keywords: Retrieval-Augmented Generation , Large Language Model , Manufacturing Specifications , Information Retrieval

1 引言

大语言模型（Large Language Models, LLMs）在众多自然语言处理任务中展现了卓越的性能。随着LLMs规模的持续扩大，其参数中编码了大量事实信息(Chowdhery et al., 2023; Kandpal et al., 2023)。因此LLMs 不仅在各类通用自然语言处理任务中表现出色，还满足了多个专业领域的应用需求，例如医疗(Singhal et al., 2023)、金融(Zhang and Yang, 2023)、法律(Fei et al., 2025)、教育(Bai et al., 2024) 和科学研究(Ma et al., 2024)。

工艺规范是对工艺过程中有关技术要求所做的一系列规定，主要包括工艺参数和工艺条件(Li et al., 2024)。工艺规范覆盖了生产过程中的工艺方法、检验规则及质量控制等关键要素，是企业工程文件的典型代表(Zhongjun et al., 2016)。因此选择工艺规范作为研究对象，对于研究LLMs在企业工程文件中的应用具有重要的实践价值。尽管LLMs在预训练阶段通过海量文本学习了广泛的知识，但其解决特定专业领域问题的能力仍然存在局限性。因此，通过对LLMs的参数进行微调(Liu et al., 2022; Hu et al., 2022)，可以使模型重新学习并掌握特定领域的相关知识。然而，LLMs的微调过程通常伴随着较高的算力成本、时间成本以及对训练数据量的要求。在工艺规范数据有限的情况下，例如某型号客机仅有460本工艺规范，导致数据量难以满足LLMs微调的需求。尽管基于工艺规范的知识注入方法(纪贵阳et al., 2024)在少样本参数微调中显著提升了LLMs的性能，但该方法仍然面临训练成本高昂和知识更新滞后等挑战。

问题	CETS0004系列死接头利用热风枪安装时的加热温度是多少
朴素RAG 检索	Chunk1: 7.2 CETS0004 系列的安装(Installation of CETS0004 series). 7.2.1 单芯导体屏蔽导线(Single core shielded cable)..... Chunk2: 7.4.2 热风枪 A02（加热温度为399℃-427℃，需加反射器）从密封套管中间开始加热。 Chunk3: 7.2.1.2 用 热风枪 从套管中间加热，然后再加热两端直至完全收缩。注：该系列死接头安装 加热温度 为 399℃-454℃ 。
朴素RAG 输出	CETS0004 系列死接头利用热风枪安装时的加热温度为 399℃-427℃ **（错误）**

Table 1: 朴素RAG在工艺规范问答任务中遇到的问题

Lewis等人(Lewis et al., 2020)提出了RAG方法，有效缓解了LLMs 在知识更新、处理长尾数据以及高昂训练和微调成本方面的挑战。RAG 通过预训练的检索器获取与问题相关的领域知识，从而提升模型生成答案的准确性。在该方法中，外部知识被划分为固定长度的文本块，并以向量形式表示并存储于向量数据库中，这种RAG范式被称为朴素RAG(Gao et al., 2023)。区别于其他的外部知识(Pratama et al., 2025)，工艺规范是一类复杂的技术文档，具有复杂的段落结构关系以及隐含的知识关联关系。因此采用固定长度文档分块会丢失工艺规范段落层级之间的结构关系以及隐含的知识关联关系，导致输出结果质量下降。如表 1所示：其中零件号“CETS0004”作为特有的工艺知识，仅在工艺规范的二级段落中出现一次。然而，朴素RAG采用固定长度文档分块，难以将零件号“CETS0004”与包含正确答案的文本块关联在一起并提供给LLMs，因此会出现如表 1所示的错误。

为解决上述朴素RAG遇到的问题，本文提出利用工艺规范中段落层级之间的结构关系以及隐含的知识关联关系来改善LLMs在工艺规范中的问答效果，本文主要有以下贡献：

（1）本文提出了一种基于工艺规范段落层级之间隐含的树结构关系的RAG方法。同时为了更好地利用工艺规范中的树结构关系，本文将工艺规范从非结构化转化为结构化的树结构对象。根据树结构对象的不同使用时机和方式，本文提出了四种不同的树结构RAG。

* 通讯作者

基金项目：辽宁省应用基础研究(2022JH2/101300248)；国家自然科学基金(U1908216)

（2）本文在8种LLM上进行了广泛的实验，实验证明，与朴素RAG 相比，树结构RAG在ACC 指标上平均提升3.81%，在ROUGE-L 指标上平均提升3.28%，在BLEU-4 指标上平均提升2.97%，证明了树结构RAG 能够改善LLMs 在工艺规范中的问答效果。

2 相关研究

2.1 基于查询的RAG

基于查询的RAG（Query-based RAG）(Zhao et al., 2024)是一种将检索后的内容与用户原始查询整合后输入生成器以生成回答的方法。Lewis 等人(Lewis et al., 2020) 在BERT中提出了RAG，旨在提升知识访问与操作的准确性、增强模型决策的可解释性，并解决知识更新的问题。Guu 等人(Guu et al., 2020) 提出了REALM，该方法尝试在传统语言模型中引入一个神经网络检索器。该检索器能够在推理阶段从大规模文本语料库中检索相关文档。Asai 等人(Asai et al., 2024) 提出了一种名为SELF-RAG的方法，旨在生成过程中解决LLMs回答中事实不准确的问题。具体而言，SELF-RAG能够使LLMs在推理阶段按需检索相关段落，并通过反思词元对生成内容进行自我评估与优化。Shi 等人(Shi et al., 2024) 提出了一种名为REPLUG的方法。该方法通过利用LLMs输出序列困惑度之间的KL散度来训练检索器，旨在解决RAG的适用性问题并提升语言模型的性能。Wang 等人(Wang et al., 2024) 提出了一种名为KGP的方法，专注于解决多文档问答问题。KGP采用知识图谱提示方法，通过构建知识图谱来辅助LLMs进行MD-QA。Ram 等人(Ram et al., 2023)提出了一种名为ICRALM的方法。该方法在语言模型生成后，基于当前已生成的文本序列作为查询，从外部知识库中检索一个或多个相关文档来改善语言模型的输出结果。

2.2 树结构在RAG的应用

Kim 等人(Kim et al., 2023) 提出了澄清树（Tree of Clarifications, TOC）方法，旨在解决用户经常提出的具有多义性的模糊查询问题。TOC 通过递归方式构建了一个树结构的澄清问题框架，从而系统化地处理查询中的歧义性。Liu 等人(Liu and Liu, 2024) 提出了检索增强思想树（Retrieval Augmented Tree of Thoughts，RAToT），旨在提升知识密集型多步问答任务中对外部文档的检索性能。通过智能且灵活的文档检索机制，RAToT 有效降低了开放域多步问答任务中生成错误答案的可能性。Hu 等人(Hu et al., 2024) 提出了一种名为自奖励树搜索（Self-Rewarding Tree Search，SeRTS）的方法。该方法旨在解决生物医学领域中知识检索性能次优的问题。SeRTS 是一种基于蒙特卡洛树搜索和自奖励范式设计的即插即用的检索方法。Sarthi 等人(Sarthi et al., 2024) 提出了一种名为Raptor的方法。该方法旨在解决现有大多数RAG方法只能检索较短文本块的问题，这一限制阻碍了LLMs对大规模语篇结构的利用。Raptor设计了一种聚类算法对文本块进行聚类，并利用LLMs生成文本块的摘要，通过聚类和摘要的反复迭代操作，构建文本块和摘要之间的摘要树结构。

本文的贡献在于探讨了在RAG构建过程中，面向复杂技术文档文档段落结构关系及隐含知识关联对RAG效果的影响。然而，上述基于查询的RAG仅将外部知识文本切分为固定长度的文本块，未能充分考虑其结构关系。尽管上述研究考虑了在RAG中应用树结构，但并未充分关注外部知识中段落与篇章之间的树结构关系。因此本文提出了树结构RAG方法，该方法通过利用外部知识中隐含的树结构，优化了RAG的生成效果。

3 方法

3.1 朴素RAG

朴素RAG 的目标是根据输入查询q，生成查询q对应的回答y。给定查询问题$q \in Q$ 和外部知识库$D_c = \{c_1, c_2, ..., c_N\}$，Naive RAG的形式化定义如公式(1)所示：

$$y = LLM\left(p\left(q,\ retrieval\left(q,\ D_c\right)\right)\right) \tag{1}$$

$$\{c_{m1}, c_{m2}, ..., c_{mk}\} = retrieval\left(q,\ D_c\right) \tag{2}$$

在公式(1)中，y为朴素RAG的输出，$LLM(\cdot)$为大语言模型(LLMs)，q为Q中的一个问题，D_c为固定长度文本块集合，$p(\cdot)$为提示函数。公式(2) 中，通过检索函数$retrieval(\cdot)$ 获

取相似度最高的前Top k 个文本块，构成集合$C_k = \{c_{m1}, c_{m2}, ..., c_{mk}\}$。最终，利用函数$p(\cdot)$ 将查询q 与集合C_k 结合，作为提示输入到$LLM(\cdot)$ 生成输出。

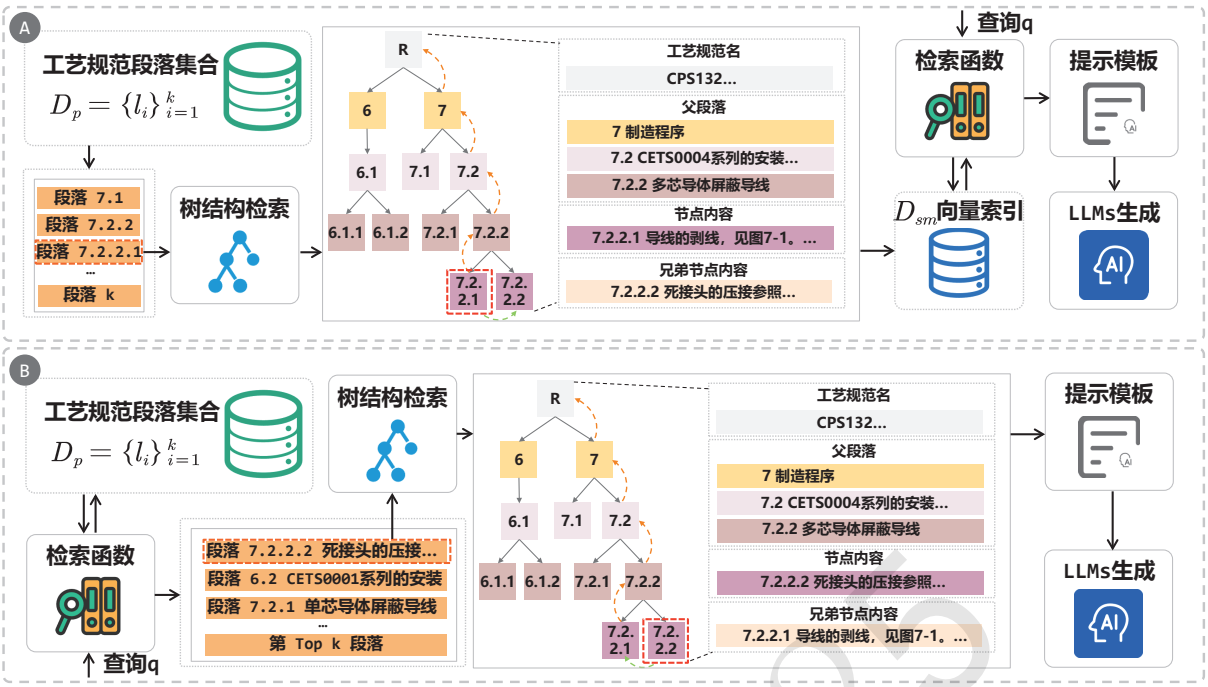

Figure 1: 工艺规范树结构RAG总体结构图，A部分为利用树结构获取工艺规范段落所属的不同层级的段落信息来构建向量索引的方法，B部分为首先利用工艺规范中的全部段落构建向量索引，随后通过树结构信息提取不同层级的段落信息的方法

3.2 树结构RAG

本文提出了一种树结构RAG方法TSR（Tree Structure Retrieval, TSR），旨在利用工艺规范段落之间的树结构关系。根据树结构利用阶段的不同，树结构检索函数$tsr(\cdot)$分为两种类型：第一种$tsr(\cdot)$方法通过树结构获取工艺规范段落所属的不同层级的段落信息来构建向量索引；第二种$tsr(\cdot)$方法则首先利用工艺规范中的全部段落构建向量索引，随后通过树结构信息提取不同层级的段落信息。图1为TSR两种不同类型的总体结构，其中第一种$tsr(\cdot)$方法的形式化定义如公式(3)所示。

$$\{s_1, s_2, ..., s_k\} = tsr\left(q, D_s\right) = retrieval\left(q, D_s\right) \tag{3}$$

第一种$tsr(\cdot)$的输入包括查询q以及工艺规范m中所有段落的最小子树对象集合D_s，其输出为检索到的前Top k个最小子树对象集合$\{s_1, s_2, ..., s_k\}$。其中s_k表示最小子树对象，包含工艺规范m中某一段落l的内容及其所属不同层级树结构中父段落节点或兄弟段落节点的内容。

第二种$tsr(\cdot)$的输入包括查询q以及工艺规范m中所有段落的集合D_p，T_{m_k}为工艺规范m的树结构对象，$traverse(\cdot)$为树结构遍历函数，用于从树结构对象T_{m_k}中获取段落l所属不同层级树结构中的父段落节点和兄弟段落节点的内容，具体的形式化定义如公式(4)所示：

$$\{s_1, s_2, ..., s_k\} = tsr\left(q, D_p\right) = traverse\left(retrieval\left(q, D_p\right) \mid T_{m_k}\right) \tag{4}$$

具体的运行流程如下，首先$retrieval(\cdot)$函数根据输入查询q从工艺规范段落集合D_p中检索到前Top k个相似段落，再利用执行$traverse(\cdot)$函数从树结构对象T_{m_k}中获取前Top k个段落l所属的最小子树对象组成集合$\{s_1, s_2, ..., s_k\}$。

3.3 树结构对象的提取

在工艺规范中，由于不同层级段落分布着不同工艺知识，且各层次段落均有章节编号，这使得工艺规范具有隐含的段落树结构关系，工艺规范的树结构示意图如图2所示。 为利

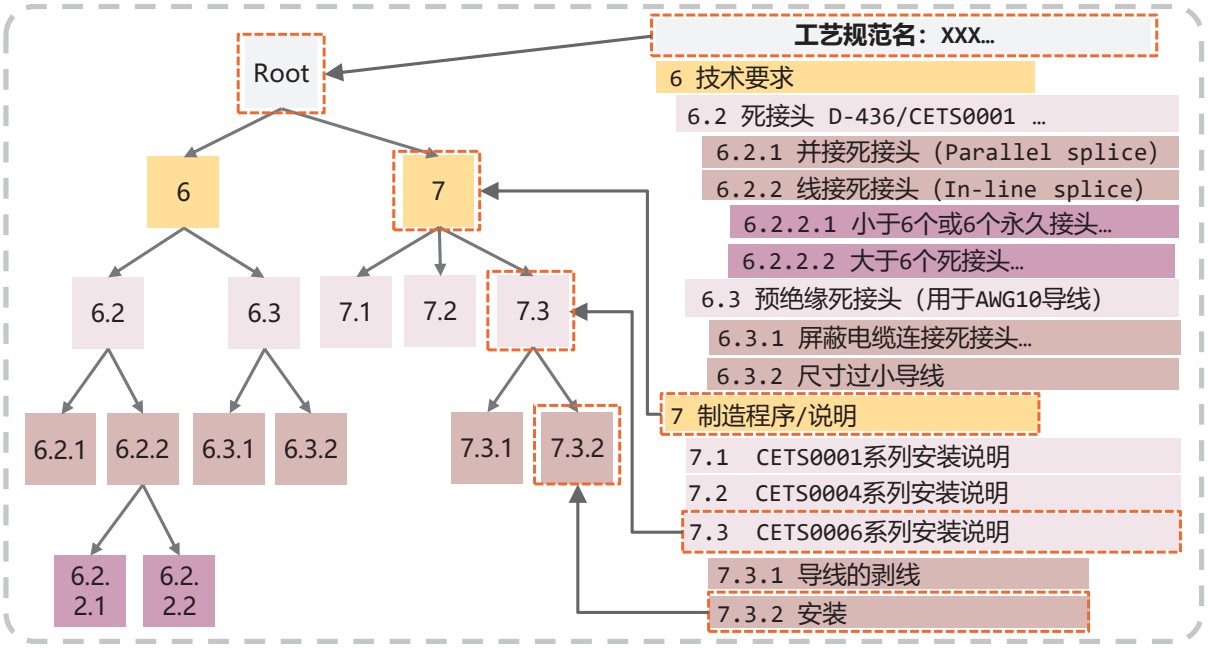

Figure 2: 工艺规范中树结构示意图

用工艺规范中的树结构关系，本文提取该结构并将其存储于树结构对象中。对于工艺规范$m_k \in \{m_1, m_2, \ldots, m_n\}$，其中$l \in m_k$，并将每个段落$l$视为树结构中的一个节点。其中树结构对象为一个集合，包含工艺规范m_k中每个段落l形成的最小子树对象s，其中s为一个四元组，包含l所属的工艺规范名、父节点段落、兄弟节点段落，树结构对象的定义如公式(5)：

$$T_{m_k} = \{s_i \mid i = 1, 2, \ldots, k\},\ s_i = (r_{l_k}, p_{l_k}, l_k, b_{l_k}) \tag{5}$$

T_{m_k}为工艺规范m_k的树结构对象，s_i为段落l_k所属的最小子树对象，其中l_k被视为叶子节点。设R为工艺规范m_k中所有段落l所属的工艺规范名集合，$r_{l_k} \in R$ 表示R中存储的l_k对应的工艺规范名；设P为工艺规范m_k中所有段落l所属的父段落节点集合，$p_{l_k} \in P$ 表示P中存储的l_k的父节点；设B为工艺规范m_k中所有段落l所属的兄弟段落节点集合，$b_{l_k} \in B$ 表示B中存储的l_k的兄弟节点。对于工艺规范m_k中的每个段落l，执行如公式(5)所示的操作，并将s_i存储至树结构对象T_{m_k}中，其中T_{m_k}表示工艺规范m_k的树结构对象。

为确定l_k所属的工艺规范名r_{l_k}和父段落节点p_{l_k}，以l_k为起点进行遍历，其获取方式如公式(6)(7)所示：

$$\mathrm{R}(l_k) = \{l_j \in A(l_k) \mid \lambda(l_j) = 0\} \tag{6}$$

$$P(l_k) = \langle l_j \in A(l_k) \mid \lambda(l_j) < \lambda(l_k) \ \wedge \ \lambda(l_j) > 0 \rangle_{\prec\lambda} \tag{7}$$

在公式(6)中，$\mathrm{R}(l_k)$表示工艺规范m_k中每个l_k所属工艺规范名的集合，$A(l_k)$表示段落l_k在工艺规范m_k中的所有前置段落的集合（以图2中的7.3.2 为例，7.3.1到工艺规范名之间的全部段落即为7.3.2 的前置段落），$\lambda(\cdot)$用于确定段落l_k所属的层级，（例如工艺规范名的层级为0，7.3.2 的层级为3。）对于前置段落集合$A(l_k)$中的每一个段落l_j，通过函数$\lambda(\cdot)$判断其所属层级，其中层级为0的段落即为l_k所属的工艺规范名。

在公式(7)中，其中P是一个有序集合，用于从前置段落集合$A(l_k)$中寻找段落l_k所属的父段落。具体方法是利用$\lambda(\cdot)$寻找层级低于l_k且$\lambda > 0$的段落，保留每一个第一次出现的段落级别小于$\lambda(l_k)$的段落作为父段落，并根据这些段落的层级λ按升序排序后存储到P。

在获取工艺规范名和父段落集合后，为利用l_k所属的兄弟段落关系，以l_k为起点在工艺规范m_k中进行遍历，从而获取与l_k具有兄弟关系的段落。如图2中，段落7.1、7.2 和7.3处于相同的段落层级，7.1和7.3即为7.2的兄弟段落，其获取方式如公式(8)(9)(10)所示：

$$B_{\mathrm{up}}(l_k) = \langle l_j \in m_k \mid j < k,\ \lambda(l_j) = \lambda(l_k) \rangle_{\prec j} \tag{8}$$

Algorithm 1: 树结构对象构建

Input: 工艺规范集合$M = \{m_1, m_2, \ldots, m_k\}$
Output: 每个工艺规范对应的树结构对象T_{m_k}
for each m_k in M **do**
  for each l_k in m_k **do**
    $\mathrm{R}(l_k) \leftarrow$ Eq.6
    $\mathrm{P}(l_k) \leftarrow$ Eq.7
    $B \leftarrow$ Eq.8, Eq.9, Eq.10
    $\mathrm{T_{m_k}} \leftarrow$ Eq.5
  end for
end for

$$B_{\text{down}}(l_k) = \langle l_j \in m_k \mid j > k,\ \lambda(l_j) = \lambda(l_k) \rangle_{\prec j} \tag{9}$$

$$B(l_k) = B_{\text{up}}(l_k) \cup B_{\text{down}}(l_k) \tag{10}$$

其中，B_{up}表示以l_k为起点向前搜索其所属兄弟段落集合，B_{down}表示以l_k为起点向后搜索其所属兄弟段落集合，B为l_k所属的全部兄弟段落集合。以公式(8)为例，首先在m_k中获取l_k的前置段落l_j，若l_j与l_k的段落层级相同，则将l_j存入B_{up}，若l_j与l_k的段落层级不同则停止循环。类似地，公式(9)的处理逻辑与之相同，用于以l_k为起点向后搜索其所属兄弟段落，集合$B(l_k)$包含了l_k所属的全部兄弟段落。

通过上述方法，本文将工艺规范从非结构化内容转化为结构化的树结构对象。提取树结构对象的伪代码如算法1所示。

3.4 树结构检索方法TSR

本文根据树结构利用时机的不同以及树结构层次的利用方式的不同，提出了四种结合不同检索时机与树结构层次的RAG方法，分别为$\mathrm{TSR_b}$、$\mathrm{TSR_p}$、$\mathrm{TSR_{tb}}$ 和$\mathrm{TSR_{tp}}$。$\mathrm{TSR_b}$的核心思想是在$retrieval(\cdot)$函数检索前，利用树结构对象获取工艺规范中每个段落所属的工艺规范名、父段落、兄弟段落内容，具体定义如公式(11)所示：

$$\{s_{m1}, s_{m2}, ..., s_{mk}\} = tsr_b\,(q, D_{sm}) = retrieval\,(q, D_{sm}) \tag{11}$$

其中$D_{sm} = \{s_{m1}, s_{m2}, ..., s_{mn}\}$为最小子树对象$s_{mn}$的集合，其中的每个最小子树对象包含段落$l$所属的工艺规范名$r_l$，父段落信息$p_l$，兄弟节点信息$b_l$，最小子树对象的提取方法如公式(12)所示：

$$s_{mn} = (r_l, p_l, l, b_l) = traverse\,(l \mid T_{m_k}) \tag{12}$$

在上述公式中，s_{mn}为段落l所属的最小子树对象，T_{m_k}为包含工艺规范名r_l，父段落信息p_l，兄弟节点信息b_l的树结构对象。$traverse(\cdot)$为遍历函数，用于从树结构对象T_{m_k}中获取l所属的r_l，p_l，b_l。对工艺规范m中的每一个段落做上述操作，得到最终的最小子树对象集合$D_{sm} = \{s_{m1}, s_{m2}, ..., s_{mn}\}$，其中包含了工艺规范$m$中从第一个段落到第$n$个段落中每个段落对应的最小子树对象。

$\mathrm{TSR_p}$的核心思想是在$retrieval(\cdot)$函数检索前，利用树结构对象获取工艺规范中除了兄弟节点外的其它全部层级的段落内容，具体定义如公式(13)所示：

$$\{s_{p1}, s_{p2}, ..., s_{pk}\} = tsr_p\,(q, D_{sp}) = retrieval\,(q, D_{sp}) \tag{13}$$

其中$D_{sp} = \{s_{p1}, s_{p2}, ..., s_{pn}\}$为最小子树对象$s_{pn}$的集合，其中的每个最小子树对象包含段落$l$所属的工艺规范名$r_l$，父段落信息$p_l$，$s_{pn}$的提取方法如公式(14)所示：

$$s_{pn} = (r_l, p_l, l) = traverse\,(l \mid T_{m_k}) \tag{14}$$

在上述公式中，s_{pn}为段落l所属的最小子树对象，$traverse(\cdot)$为遍历函数，用于从树结构对象T_{m_k}中获取l所属的r_l，p_l。对工艺规范m中的每一行做上述操作，得到最终的最小子树对象集

合$D_{sp}=\{s_{p1},s_{p2},...,s_{pn}\}$，其中包含了工艺规范$m$中从第一个段落到第$n$个段落中每个段落对应的最小子树对象。

TSR_{tb}的核心思想是在$retrieval(\cdot)$函数检索工艺规范中的段落后，利用树结构对象获取工艺规范中每个段落所属的工艺规范名、父段落、兄弟段落内容，具体定义如公式(15)所示：

$$\{s_{m1},s_{m2},...,s_{mk}\}=tsr_{tb}\left(q,D_p\right)=traverse\left(retrieval\left(q,D_p\right)\mid T_{m_k}\right) \tag{15}$$

在上述公式中，当$retrieval(\cdot)$函数检索完成后，利用遍历函数$traverse(\cdot)$从树结构对象T_{m_k}中获取被检索到的段落l所属的工艺规范名r_l，父段落信息p_l，兄弟节点信息b_l，组成新的最小子树对象s_m。

TSR_{tp}的核心思想是在$retrieval(\cdot)$函数检索工艺规范中的段落后，利用树结构对象获取工艺规范中除了兄弟节点外的其它全部层级的段落内容，具体定义如公式(16)所示：

$$\{s_{p1},s_{p2},...,s_{pk}\}=tsr_{tp}\left(q,D_p\right)=traverse\left(retrieval\left(q,D_p\right)\mid T_{m_k}\right) \tag{16}$$

在上述公式中，当$retrieval(\cdot)$函数检索完成后，利用遍历函数$traverse(\cdot)$从树结构对象T_{m_k}中获取被检索到的段落l所属的工艺规范名r_l，父段落信息p_l，组成新的最小子树对象s_p。

4 实验

4.1 实验设置

数据集 实验中使用的问答对数据集来源于某型号飞机的5本工艺规范，涵盖零件、设备和操作等各种类型的线束安装工艺。其中，零件类和设备类工艺规范明确了零件及工具适用或应符合的工艺条件，而操作类工艺规范则对工艺过程中的工艺参数与工艺方法进行了规定和要求。经过人工标注，共构建了311个问答对。

实验环境 本文在八个LLM上进行了实验，包括Yi1.5-9B-Chat、Llama3-8B-Instruct、ChatGLM3-6B、GLM4-9B-Chat、Qwen1.5-7B-Chat、Qwen1.5-14B-Chat、Qwen2-7B-Instruct和Gemma-2-9B-IT。实验GPU采用NVIDIA A800 80G PCIe。向量表征模型采用bge-large-zh(Xiao et al., 2024)，向量数据库采用FAISS(Johnson et al., 2021)。

4.2 评价指标

本文从准确性与流畅性两个维度对RAG系统的性能进行了评估。评估过程中采用概念准确性（Accuracy of Concepts, ACC）指标衡量LLMs的输出质量，其中生成答案覆盖正确答案中关键实体的数量越多，其准确性得分越高。本文通过正则匹配方法从人工标注的实体集合中提取目标实体，并据此进行准确性计算。具体评估标准定义如公式(17)所示，其中$Num_{correct}$为输出答案中总实体数，Num_{total}为正确答案中总实体数，N为问答对数量。流畅性本文使用生成任务常用的BLEU-4 (Papineni et al., 2002)和ROUGE-L(Lin, 2004)评价指标进行评价。

$$ACC=\frac{\sum_{i=1}^{N}(Num_{correct}/Num_{total})}{N} \tag{17}$$

4.3 对比方法

为了验证本文提出的树结构RAG的有效性，本文将树结构RAG与另外四种方法进行对比，这四种方法分别是：朴素RAG、LLM_{Base}、Raptor(Sarthi et al., 2024)和SPC。

- LLM_{Base}：将构建的问答对中的问题输入至未在该领域进行微调的LLMs中，记录其输出结果并进行评估。
- 朴素RAG：基于LangChain(Topsakal and Akinci, 2023)框架，本文将工艺规范文档切分为固定长度文本块。具体实现中文本块长度分别设置为250字符和750字符两种规格，并在相邻文本块间保留50字符的重叠部分以维持上下文连贯性。
- RAPTOR: RAPTOR通过迭代执行文本块的聚类并生成层次化摘要树结构。该算法首先对文本进行分块处理并聚类，生成每个聚类块的摘要；随后通过循环迭代聚类与摘要生成过程，最终构建出具有层级关联的树形文本块架构。基于此结构，$\text{RAPTOR}_{\text{Tree}}$采用自底向

上的遍历策略：从摘要树底层开始，每层选取与查询向量相似度排名前k的文本块，迭代至根节点层时终止遍历。$\text{RAPTOR}_{\text{Col}}$首先对摘要树进行展平处理，继而采用自下而上逐层筛选，当累计选中的文本块总词元数达到LLMs预设的上下文长度阈值时，筛选过程终止。

- SPC：次级段落块（Secondary Paragraph Chunk, SPC）是基于工艺规范二级标题结构设计的层级化分块策略。SPC以工艺规范的二级标题作为切分依据来生成文本块。SPC_{H}将一级段落信息与SPC块结合，形成一个SPC_{H}块。类似地，SPC_{HD}从SPC_{H} 扩展而来，在SPC_{H}的基础上补充工艺规范名信息。

4.4 对比实验

本文将TSR_{b}与对比方法进行比较，对比实验结果如表 2所示，从中可以得出三个结论。

大语言模型	Yi1.5-9B-Chat			Llama3-8B-Instruct			Chatglm3-6b			GLM4-9b-Chat		
方法	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4
TSR_{b}	69.67	**30.94**	**16.54**	56.14	**32.72**	**18.06**	**64.05**	**34.35**	**18.64**	**66.74**	**35.72**	20.78
LLM_{Base}	44.23	5.33	0.76	31.21	7.36	1.11	42.28	9.79	2.48	45.85	6.18	0.96
RAG_{250}	65.35	27.55	13.91	49.57	30.46	15.98	62.83	32.79	17.24	62.92	32.24	17.48
RAG_{750}	63.74	25.19	12.19	44.73	27.09	13.46	58.48	29.50	15.92	62.85	29.80	16.20
$\text{Raptor}_{\text{Tree}}$	62.84	25.26	10.73	59.98	28.40	13.26	55.27	27.06	12.06	59.98	28.40	13.26
$\text{Raptor}_{\text{Col}}$	65.80	26.85	12.83	**66.21**	27.78	13.56	55.78	27.58	12.01	66.21	27.78	13.56
SPC	64.95	28.91	14.75	54.18	30.85	16.73	61.25	32.25	17.12	63.81	33.27	18.99
SPC_{H}	**70.08**	27.35	13.58	52.31	30.02	16.08	62.31	32.50	16.72	66.19	34.69	20.01
SPC_{HD}	67.59	29.39	15.72	54.09	29.72	15.76	62.00	33.84	18.02	66.58	35.16	**20.80**
大语言模型	Qwen1.5-7B-Chat			Qwen1.5-14B-Chat			Qwen2-7B-Instruct			Gemma2-9b-it		
方法	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4
TSR_{b}	65.33	**32.33**	**16.40**	**64.77**	34.14	**17.53**	**61.59**	**33.24**	**18.82**	43.91	**42.19**	**26.50**
LLM_{Base}	43.91	7.17	1.04	43.91	8.12	1.45	44.65	6.10	0.95	**43.95**	8.68	1.32
RAG_{250}	63.15	31.00	14.55	64.20	30.28	14.50	58.11	28.60	14.66	35.69	36.42	21.19
RAG_{750}	61.33	30.15	14.15	62.48	30.14	14.34	57.36	26.84	14.04	35.11	34.20	19.89
$\text{Raptor}_{\text{Tree}}$	62.84	25.26	10.73	33.23	20.20	5.17	55.27	27.06	12.06	41.61	37.37	21.99
$\text{Raptor}_{\text{Col}}$	**65.80**	26.85	12.83	27.64	17.38	3.46	55.78	27.58	12.01	32.00	33.68	18.73
SPC	62.46	31.47	15.09	62.00	32.50	16.17	57.25	31.79	17.97	39.00	39.45	24.72
SPC_{H}	63.50	31.63	15.15	63.02	34.09	16.66	59.66	32.03	18.48	39.72	39.18	23.76
SPC_{HD}	64.44	31.69	15.95	63.74	**34.29**	17.05	58.52	32.07	18.45	41.50	40.77	25.21

Table 2: TSR_{b} 与其他方法的对比结果

1. TSR_{b}方法的表现优于其他对比方法，取得了最优的平均结果。与RAG_{250} 相比，TSR_{b}在ACC 指标上平均提升了3.81%，在ROUGE-L 指标上提升了3.28%，在BLEU-4 指标上提升了2.97%。此外，次优方法SPC_{HD} 在ACC 指标上平均提升了1.72%，在ROUGE-L 指标上提升了1.09%，在BLEU-4 指标上提升了0.79%，因此TSR_{b}的有效性得到了验证。

2. LLM_{Base} 的效果显著低于全部RAG方法，这表明实验中使用的LLMs并未在工艺规范领域进行预训练。此外结果还可以验证利用RAG可以有效提升针对工艺规范的问答性能。

3. $\text{Raptor}_{\text{Tree}}$和$\text{Raptor}_{\text{Col}}$方法的评价指标低于$\text{TSR}_{\text{b}}$和$\text{RAG}_{250}$。表明在工艺规范这种具有复杂的段落结构关系以及隐含的知识关联关系的文档中生成摘要不准确，这表明为工艺规范构建文本块生成摘要的方法不可行。

4.5 树结构RAG对比实验

为了进一步验证TSR_{b}的有效性，本文将TSR_{b}与不同参数的树结构RAG进行比较，实验结果如表 3所示。

1. TSR_{b} 相较于TSR_{tb}，在ACC 上平均提升3.07%，在ROUGE-L 上平均提升3.02%，在BLEU-4 上平均提升2.96%；而TSR_{p} 相较于TSR_{tp}，在ACC 上平均提升2.04%，在ROUGE-L 上平均提升3.01%，在BLEU-4 上平均提升2.57%。这种性能差异源于利用树结构的时机不同，其中TSR_{b} 和TSR_{p} 方法在检索前利用树结构信息构建向量索引。实验结果表明，在向量表征过程中引入树结构关系能够显著提升RAG的生成质量。

大语言模型	Yi1.5-9B-Chat			Llama3-8B-Instruct			Chatglm3-6b			GLM4-9b-Chat		
方法	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4
RAG_{250}	65.35	27.55	13.91	49.57	30.46	15.98	62.83	32.79	17.24	62.92	32.24	17.48
TSR_b	**69.67**	**30.94**	**16.54**	**56.14**	**32.72**	**18.06**	**64.05**	**34.35**	**18.64**	**66.74**	**35.72**	**20.78**
TSR_p	65.63	29.49	15.65	54.08	28.96	14.43	62.06	34.29	17.76	62.51	34.59	19.16
TSR_{tb}	65.33	26.32	13.01	53.76	31.43	16.44	62.2	31.88	16.28	65.67	32.62	17.67
TSR_{tp}	65.62	24.75	11.68	50.28	27.86	13.00	59.95	31.07	15.11	62.66	31.86	16.82
大语言模型	Qwen1.5-7B-Chat			Qwen1.5-14B-Chat			Qwen2-7B-Instruct			Gemma2-9b-it		
方法	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4
RAG_{250}	63.15	31	14.55	64.2	30.28	14.5	58.11	28.6	14.66	35.69	36.42	21.19
TSR_b	**65.33**	**32.33**	**16.4**	**64.77**	**34.14**	**17.53**	**61.59**	**33.24**	**18.82**	**43.91**	**42.19**	**26.50**
TSR_p	62.68	29.54	13.63	61.59	33.13	16.18	55.41	30.29	16.61	40.80	40.02	23.53
TSR_{tb}	64	30.61	14.09	63.11	33.29	15.77	56.83	28.64	14.95	36.71	36.71	21.40
TSR_{tp}	61.84	27.67	11.76	61.91	31.22	14.44	51.27	26.23	13.08	34.89	35.57	20.50

Table 3: 树结构RAG对比结果

2. TSR_{tb} 相较于RAG_{250}在ACC 上平均提升0.72%，在ROUGE-L 上平均提升0.27%，在BLEU-4 上平均提升0.01%。尽管RAG_{250} 和TSR_{tb} 在检索过程中均未包含工艺规范名、父段落及兄弟段落内容，但TSR_{tb} 在构建提示时通过遍历树结构对象补充了这些信息。虽然TSR_{tb} 的评价指标低于TSR_b，但其结果仍表明，利用树结构对象在检索后获取工艺规范名、父段落及兄弟段落内容的方法能够有效提升生成结果的质量，也进一步证明利用树结构的有效性。

3. TSR_b对比TSR_p在ACC 上平均提升3.43%，ROUGE-L 上平均提升1.92%，BLEU-4 上平均提升了2.04%。TSR_{tb} 对比TSR_{tp} 在ACC 上平均提升2.40 %，ROUGE-L 上平均提升1.91 %，BLEU-4 上平均提升了1.65 %。其性能差异在于TSR_b 和TSR_{tb} 利用了树结构对象中的兄弟段落内容，证明了同级段落间的内容和存在的知识关联关系能够显著提高生成质量。

4. 在实际应用中，TSR_b 在评价指标上的表现优于其他树结构RAG方法，同时在上下文长度和时间效率方面也较其他RAG方法取得了显著改进，具体实验结果详见第4.7节和第4.8节。

4.6 消融实验

为了进一步验证TSR_b中利用不同层级树结构的有效性，本文在TSR_b的基础上进行消融实验，其中$TSR_{p(rl)}$为不使用父段落信息p、兄弟段落信息b的方法。$TSR_{p(l)}$则为不使用工艺规范名r、父段落信息p、兄弟段落信息b的方法。实验结果如表 4所示，由此可得出以下结论：

大语言模型	Yi1.5-9B-Chat			Llama3-8B-Instruct			Chatglm3-6b			GLM4-9b-Chat		
方法	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4
TSR_b	**69.67**	**30.94**	**16.54**	**56.14**	**32.72**	**18.06**	**64.05**	**34.35**	**18.64**	**66.74**	**35.72**	**20.78**
TSR_p	65.63	29.49	15.65	54.08	28.96	14.43	62.06	34.29	17.76	62.51	34.59	19.16
$TSR_{p(rl)}$	65.09	22.26	9.58	46.09	28.11	13.53	59.53	31.46	15.55	60.74	31.90	15.69
$TSR_{p(l)}$	61.63	25.46	12.43	46.45	28.24	13.92	57.70	32.01	15.51	58.20	30.02	14.82
大语言模型	Qwen1.5-7B-Chat			Qwen1.5-14B-Chat			Qwen2-7B-Instruct			Gemma2-9b-it		
方法	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4	ACC	ROUGE-L	BLEU-4
TSR_b	**65.33**	**32.33**	**16.4**	**64.77**	**34.14**	**17.53**	**61.59**	**33.24**	**18.82**	**43.91**	**42.19**	**26.50**
TSR_p	62.68	29.54	13.63	61.59	33.13	16.18	55.41	30.29	16.61	40.80	40.02	23.53
$TSR_{p(rl)}$	62.53	28.03	11.85	60.87	31.10	13.97	49.38	25.55	11.93	36.59	36.85	21.42
$TSR_{p(l)}$	61.07	27.75	11.84	60.20	27.85	11.45	47.45	24.80	11.69	33.62	35.47	20.67

Table 4: TSR_b消融实验对比结果

1. TSR_b在所有模型和指标上均显著优于消融版本。其性能差异在于TSR_b利用了完整的树结构信息，证明同时利用工艺规范名、父段落和兄弟段落内容能够实现最优的问答效果。

2. TSR_p 相较于$TSR_{p(rl)}$在ACC 上平均提升了2.99%，在ROUGE-L 上平均提升了3.13%，在BLEU-4 上平均提升了2.93%。提升的主要原因在于工艺规范中部分工艺知识仅出现在父

段落中并且仅出现一次。例如，零件号“CETS0001”出现在7.1 节中，而若回答“CETS0001安装时剥掉导线绝缘层的长度为多少”这一问题时，其中“导线绝缘层的长度”则位于7.1.2.1节中。因此，如果不考虑父段落内容，则无法准确回答此类问题。

3. $TSR_{p(rl)}$ 相较于TSR_l在ACC 上平均提升了1.81%，在ROUGE-L 上平均提升了0.46%，在BLEU-4 上平均提升了0.15%。提升的主要原因在于回答工艺知识具体出处的问题时需要利用工艺规范名。例如，对于问题“剥去CETS0001 系列死接头外护套的参考文件是什么”，如果不考虑工艺规范名则无法准确回答此类问题。

4.7 上下文长度实验

为探究提示词长度对生成结果的影响及不同RAG方法的词元消耗成本，本文设计了上下文长度实验。通过LLM分词器对输入提示进行分词处理，统计各提示的平均词元数与最大词元数。具体结果如表 5 所示，由此可得出以下结论：

1. TSR_b 生成的外部知识文档质量最高，并且包含最多有用信息。在最佳结果的情况下，TSR_b 的平均词元数仅高于RAG_{250} 和TSR_p。这表明TSR_b 在尽可能多地携带有用信息的同时，减少了输入LLM上下文的长度。

2. 提示词长度的增加并不一定能够提升问答效果。在4.4节的对比实验中，TSR_b 的表现优于RAG_{750}、SPC_{HD}和RAPTOR 等提示词长度更长的方法。这表明提示词长度越长，可能引入的混淆性信息越多，同时也证明了TSR_b 在较短提示词的情况下仍能准确定位到包含有效信息的文档。

大语言模型	Yi1.5-9B-Chat		Chatglm3-6b		GLM4-9b-Chat		Qwen1.5-7B-Chat	
方法	平均(Avg)	最大(Max)	平均(Avg)	最大(Max)	平均(Avg)	最大(Max)	平均(Avg)	最大(Max)
TSR_b	**894**	**2153**	**915**	**2222**	**823**	**2046**	**895**	**2208**
RAG_{250}	417	536	455	589	417	536	451	594
RAG_{750}	1392	1658	1422	1691	1330	1579	1443	1730
$Raptor_{Tree}$	5356	6282	3367	3480	1533	1771	3561	4078
$Raptor_{Col}$	3352	3515	3144	3898	3383	3458	3429	3549
SPC	1106	3661	1131	3715	1034	3434	1115	3697
SPC_H	1160	3706	1185	3761	1077	3470	1168	3742
SPC_{HD}	1181	3760	1209	3818	1095	3515	1187	3790
TSR_{tb}	1058	2415	1083	2474	978	2281	1064	2442
TSR_p	512	1289	523	1303	455	1160	500	1269
TSR_{tp}	501	1186	513	1190	449	1055	494	1176
大语言模型	Qwen1.5-14B-Chat		Qwen2-7B-Instruct		Llama3-8B-Instruct		gemma-2-9b-it	
方法	平均(Avg)	最大(Max)	平均(Avg)	最大(Max)	平均(Avg)	最大(Max)	平均(Avg)	最大(Max)
TSR_b	**895**	**2208**	**895**	**2208**	**949**	**2368**	**907**	**2252**
RAG_{250}	451	594	451	594	476	638	452	597
RAG_{750}	1443	1730	1443	1730	1534	1844	1423	1722
$Raptor_{Tree}$	2494	2823	4801	5254	5945	7100	2344	2587
$Raptor_{Col}$	3429	3549	3398	3527	3383	3458	3390	3475
SPC	1115	3697	1115	3697	1212	4165	1142	3812
SPC_H	1168	3742	1168	3742	1263	4207	1192	3854
SPC_{HD}	1187	3790	1187	3790	1278	4252	1207	3896
TSR_{tb}	1064	2442	1064	2442	1128	2801	1078	2545
TSR_p	500	1269	500	1269	515	1409	509	1298
TSR_{tp}	494	1176	494	1176	506	1256	500	1183

Table 5: 词元消耗数统计

4.8 时间效率实验

为探究TSR_b 的运行效率，本文设计了时间效率实验。通过记录不同对比方法在数据集上问答的具体时间进行比较，用时越短表明效率越高。实验结果如表 6 所示，由此可得出以下结论：

1. TSR_b 的平均耗时为16.16 分钟，仅高于RAG_{250} 的15.04 分钟，这表明TSR_b 在效果最优的同时仍能保持较高的运行效率。

2. 平均耗时和输入到LLMs中提示的长度呈正相关。在4.7节的上下文长度实验中，TSR_b 的平均词元消耗数为897、而RAG_{250}、SPC_{HD}和$Raptor_{Col}$的平均词元消耗数分别为：446、1191、3363，其对应的平均耗时分别为15.04、16.85、19.34。证明了输入到LLMs 中提示的长度越短，运行效率越高，同时进一步证明了TSR_b的运行效率较高。

大语言模型	Yi1.5-9B-Chat	Chatglm3-6b	GLM4-9b-Chat	Qwen1.5-7B-Chat
方法	时间(min)	时间(min)	时间(min)	时间(min)
LLM_{base}	28.93	11.23	12.44	14.88
RAG_{250}	29.4	10.61	11.95	16.26
$Raptor_{Col}$	35.63	15.04	18.83	23.25
SPC_{HD}	34.56	13.84	13.52	18.23
TSR_b	33.31	13.58	13.71	17.86
大语言模型	Qwen1.5-14B-Chat	Qwen2-7B-Instruct	Llama3-8B-Instruct	gemma-2-9b-it
方法	时间(min)	时间(min)	时间(min)	时间(min)
LLM_{base}	24.8	20.43	19.97	3.78
RAG_{250}	13.72	16.88	14.78	6.74
$Raptor_{Col}$	14.01	21.04	18.83	8.05
SPC_{HD}	14.92	15.95	16.65	7.11
TSR_b	14.47	15.58	15.92	4.88

Table 6: 时间消耗统计

5 结论

本文研究了不同RAG方法对工艺规范领域问答构建的影响。其中朴素RAG采用固定长度的文本分块，但这种方法会丢失工艺规范段落层级之间的结构关系以及隐含的知识关联关系，导致输出效果不佳。本文提出了基于工艺规范树结构的RAG。实验结果表明，TSR_b在不同参数规模的LLMs上均表现最佳，并且具有较低的应用成本。这表明，在诸如工艺规范等复杂技术文档中，考虑文档结构对构建其RAG具有积极影响。

参考文献

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yu Bai, Jun Li, Jun Shen, and Liang Zhao. 2024. Investigating the efficacy of chatgpt-3.5 for tutoring in chinese elementary education settings. *IEEE Trans. Learn. Technol.*, 17:2156–2171.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Bosma, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Zhiwei Fei, Songyang Zhang, Xiaoyu Shen, Dawei Zhu, Xiao Wang, Jidong Ge, and Vincent Ng. 2025. Internlm-law: An open-sourced chinese legal large language model. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 9376–9392.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, et al. 2023. Retrieval-augmented generation for large language models: A survey. *CoRR*, abs/2312.10997.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, et al. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Minda Hu, Licheng Zong, Hongru Wang, Jingyan Zhou, et al. 2024. Serts: Self-rewarding tree search for biomedical retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 1321–1335. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, 7(3):535–547.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, pages 15696–15707. PMLR.

Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, and Jaewoo Kang. 2023. Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 996–1009. Association for Computational Linguistics.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Ruiting Li, Peiyan Wang, Libang Wang, Danqingxin Yang, and Dongfeng Cai. 2024. A corpus and method for chinese named entity recognition in manufacturing. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*, pages 264–272. ELRA and ICCL.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Xuemin Liu and Jie Liu. 2024. Retrieval augmented tree of thoughts. In *Natural Language Processing and Chinese Computing - 13th National CCF Conference, NLPCC 2024, Hangzhou, China, November 1-3, 2024, Proceedings, Part II*, volume 15360 of *Lecture Notes in Computer Science*, pages 462–473. Springer.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, et al. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 61–68. Association for Computational Linguistics.

Pingchuan Ma, Tsun-Hsuan Wang, Minghao Guo, Zhiqing Sun, et al. 2024. LLM and simulation as bilevel optimizers: A new paradigm to advance physical scientific discovery. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

I Putu Agus Eka Pratama, I Made Oka Widyantara, Linawati, Nyoman Gunantara, Sinung Suakanto, and Eka Tresna Irawan. 2025. Technology-enhanced learning for user security awareness using ai-based naive rag: A design and prototype. In *2025 International Conference on Advancement in Data Science, E-learning and Information System (ICADEIS)*, pages 1–6.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. RAPTOR: recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

第二十四届中国计算语言学大会论文集，第143页-第155页，济南，中国，2025年8月11日至14日。

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, et al. 2024. REPLUG: retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 8371–8384. Association for Computational Linguistics.

Karan Singhal, Shekoofeh Azizi, Tao Tu, Mahdavi, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Oguzhan Topsakal and Tahir Cetin Akinci. 2023. Creating large language model applications utilizing langchain: A primer on developing llm apps fast. In *International Conference on Applied Engineering and Natural Sciences*, volume 1, pages 1050–1056.

Yu Wang, Nedim Lipka, Ryan A. Rossi, Alexa F. Siu, et al. 2024. Knowledge graph prompting for multi-document question answering. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 19206–19214. AAAI Press.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, et al. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 641–649. ACM.

Xuanyu Zhang and Qing Yang. 2023. Xuanyuan 2.0: A large chinese financial chat model with hundreds of billions parameters. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 4435–4439.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, et al. 2024. Retrieval-augmented generation for ai-generated content: A survey. *CoRR*, abs/2402.19473.

Ding Zhongjun, Yang Zhenghui, Zhang Jinjun, Wang Huijun, and Xiang Ying. 2016. Construction of process specification system based on job knowledge management. *Aerospace Industry Management*, (05):58–61.

纪贵阳, 王裴岩, and 余卓. 2024. 面向工艺规范文本的大语言模型知识注入方法研究. 计算机科学与探索, 18(09):2361–2369.