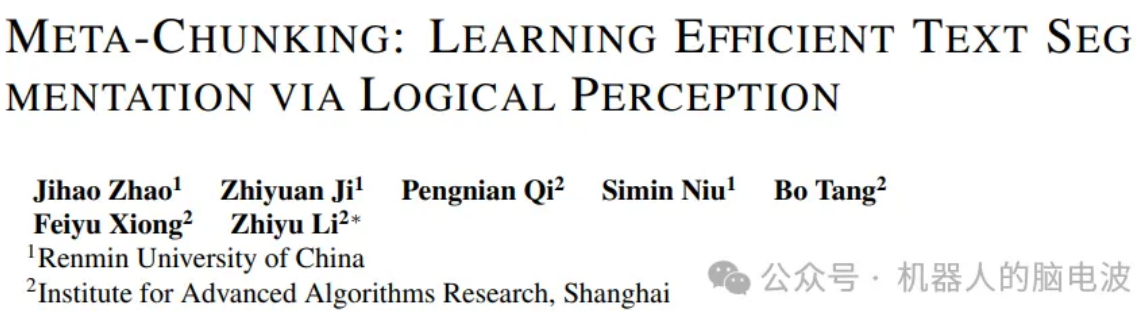
人大Meta-chunking：又快有准的RAG文本分割技术，1.3倍提升，耗时减半

cathy 机器人的脑电波 2024年11月20日 11:18 重庆



META-CHUNKING: LEARNING EFFICIENT TEXT SEGMENTATION VIA LOGICAL PERCEPTION

Jihao Zhao[1] Zhiyuan Ji[1] Pengnian Qi[2] Simin Niu[1] Bo Tang[2]
Feiyu Xiong[2] Zhiyu Li[2]*
[1]Renmin University of China
[2]Institute for Advanced Algorithms Research, Shanghai

代码：https://github.com/IAAR-Shanghai/Meta-Chunking

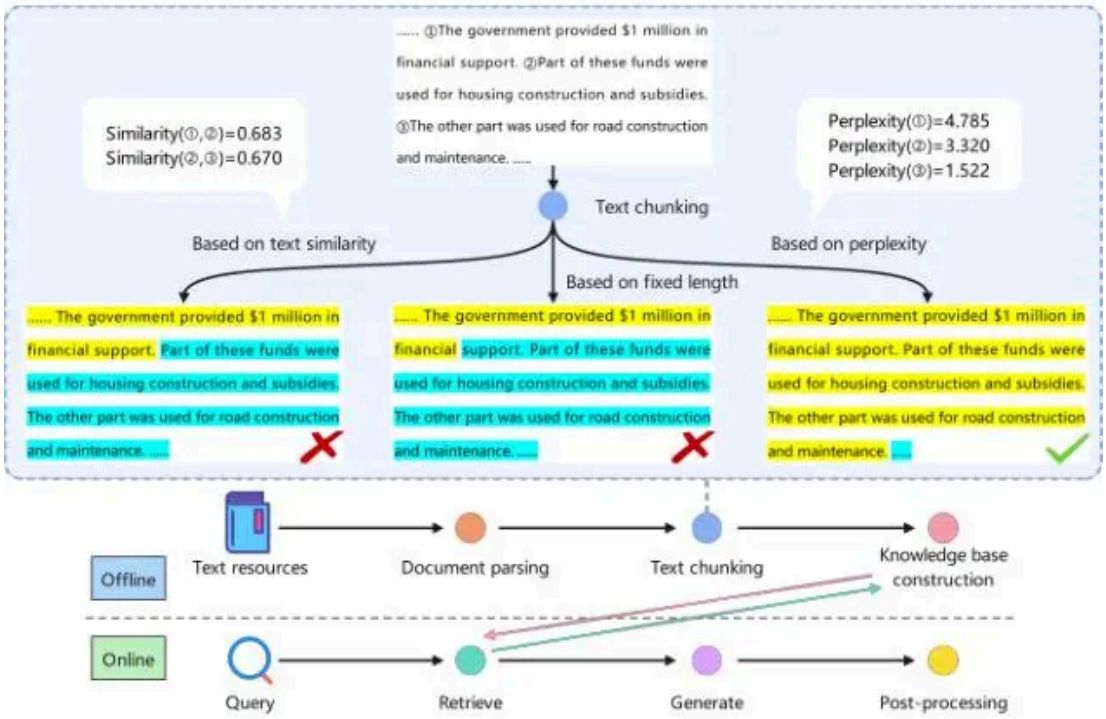
摘要

检索增强生成（RAG），虽然作为大型语言模型（LLM）的有效补充，却常常忽略其流程中文本分块这一关键环节，这会影响知识密集型任务的质量。提出了一种称为元分块（Meta-Chunking）的概念，它是在句子和段落之间的一种颗粒度，由段落中的一组句子组成，这些句子之间具有深层次的语言逻辑联系。为了实施元分块，设计了基于LLM的两种策略：边缘采样分块和困惑度分块。**前者使用LLM对连续句子是否需要分段进行二分类**，通过边缘采样获得的概率差异来做出决策。**后者则通过分析困惑度分布的特征**，精确识别文本的版块边界。此外，还考虑到不同文本的内在复杂性，提出了一种结合元分块与动态合并的策略，以在细粒度与粗粒度文本分块之间实现平衡。基于11个数据集的实验表明，元分块可以更有效地提高基于RAG的单跳和多跳问答的性能。例如，在2WikiMultihopQA数据集上，其相对于相似性分块**提升了1.32，同时仅消耗了45.8%的时间。**

元分块

主要贡献是一种创新的文本分割技术——元分块，该技术利用LLM的能力将文档灵活地划分为逻辑连贯、独立的块。此方法的核心原则是：**允许块大小的可变性，以更有效地捕捉和保持内容的逻辑完整性。**这种颗粒度的动态调整确保每个分段块都包含完整且独立的思想表达，避免在分割过程中逻辑链的断裂。这不仅增强了文档检索的相关性，还提高了内容的清晰度。

如图所示，如下图展示该方法结合了传统文本分割策略的优点，如遵循预设的块长度限制并确保句子的结构完整性，同时提高了在分割过程中的逻辑连贯性保障的能力。关键在于在句子级和段落级文本颗粒度之间引入了一种新的概念：元分块。一个元块是一个段落内的按顺序安排的句子集合，这些句子不仅具有语义上的相关性，更重要的是，包含深层次的语言逻辑联系，包括但不限于**因果关系、过渡关系、平行关系以及递进关系。**这些关系超越了单纯的语义相似性。



为了达到这一目标，设计并实现了以下两种策略：

边缘采样分块：面对一段文本，初始步骤是将其分解为一组句子，记为(c1, C2, ... , n)，目标是进一步将这些句子细分为多个块，形成新的集合(X1, X2, ... , Xk)，其中每个块均是原始句子的一个连贯分组。

可将此方法公式化为：其中(ki, k2)表示对分段判断进行的一种二元决策。Prompt(i, X)代表在i∈{1}r=1与X'之间形成的指令，关于是否应将它们合并，X'范围可包括单个句子或多个句子。通过由模型M得出的概率PM，可以得出两种选择之间的概率差异MarginM(ci)。随后，通过将MarginM(ci)与阈值进行对比，初始给其分配值O，然后记录历史MarginM(ci)并计算其平均值进行调整。

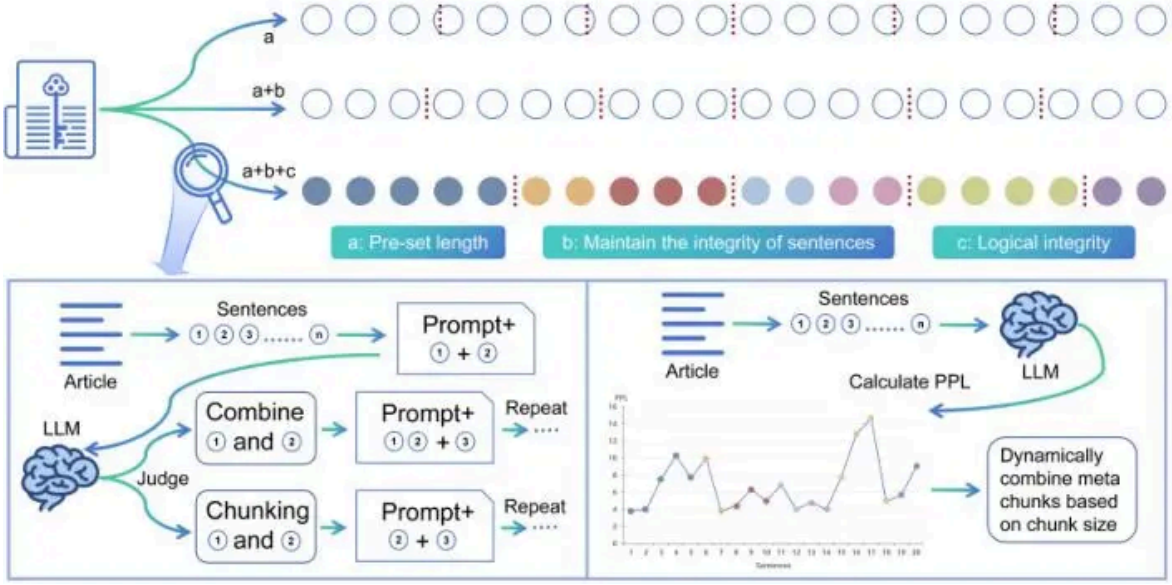
$$\text{Margin}_M(x_i) = P_M\left(y = k_1 | \text{Prompt}(x_i, X')\right) - P_M\left(y = k_2 | \text{Prompt}(x_i, X')\right)$$

困惑度分块：类似地，首先将文本分割成句子，并使用模型计算每个句子的困惑度（PPL）；根据其前置句子：其中K代表着在中总共的标记数t。表示第k个标记，而<t表示所有在i之前的标记。为了定位文本分割的关键点，算法进一步分析PPLseq = (PPLM(ai), PPLM(r2), ..., PPLM(rn))分布特征，特别关注识别最小值：这些最小值被认为是潜在的块边界。如果文本超出了LLM或设备的处理范围，将战略性地引入一个键值（KV）缓存机制。

$$\text{PPL}_M(x_i) = \frac{\sum_{k=1}^{K} \text{PPL}_M(t_k^i | t_{<k}^i, t_{<i})}{K}$$

如，文本被首先根据标记分成几个部分，形成多个子序列。随着困惑度计算的进行，当GPU内存即将超过服务器配置时，调整部分文本，以不牺牲过多的上下文连续性。为了满足用户的多样化分块需求，**仅仅通过调整阈值来控制块大小，有时会导致随着阈值的增加而块大小不均匀。因此，提出了一种结合元分块与动态合并的策略，旨在灵活应对各种分块要求。**首先，根据困惑度分布设定初始阈值O或特定值并进行元分块操作，初步将文档分为一系列基本单位(ci, C2, ... , Ca)。随后，根据用户指定的块长L，逐一合并相邻的元块，直到总长度满足或接近要求。具体而言，如果len(c1, C2, C3) = L或len(c1, C2, C3) < L且len(c1, C2, C3, C4) > L，则将C1, C2, C3视为一个完整的块。

$$\text{Minima}_{index}(\text{PPL}_{seq}) = \Big\{ i \;\Big|\; \min(\text{PPL}_M(x_{i-1}), \text{PPL}_M(x_{i+1})) - \text{PPL}_M(x_i) > \theta,$$
$$or\ \text{PPL}_M(x_{i-1}) - \text{PPL}_M(x_i) > \theta\ and\ \text{PPL}_M(x_{i+1}) = \text{PPL}_M(x_i) \Big\}$$



理论分析 - 困惑度分块

大型语言模型（LLMs）的设计目标是学习一个分布Q，该分布能够逼近经验分布P，并用作度量。在离散情境下，Q相对于P的交叉熵形式定义如下：

$$H(P,Q) = \mathrm{E}_p[-logQ] = -\sum P(x)\log Q(x) = H(P) + D_{KL}(P||Q)$$

其中H(P)表示经验熵，而DKL(PI||Q)是Q和P之间的Kullback-Leibler（KL）散度。LLMs的困惑度（PPL）在数学上定义为：

$$\text{PPL}(P,Q) = 2^{H(P,Q)}$$

需要注意的是，由于H(p)是不可优化和有界的，真正影响不同LLMs中PPL计算差异的是KL散度，它作为评估分布差异的度量。KL散度越大，两个分布之间的差异越大。此外，**较高的PPL表明LLMs对真实内容的认知偏差，而这些部分不应被分段。**

另一方面，香农（1951年）通过一个函数近似任何语言的熵，

$$\begin{aligned} G_K &= -\sum_{T_k} P(T_k)\log_2 P(t_k|T_{k-1}) \\ &= -\sum_{T_k} P(T_k)\log_2 P(T_k) + \sum_{T_{k-1}} P(T_{k-1})\log_2 P(T_{k-1}) \end{aligned}$$

T表示文本序列中的k个连续标记，熵可以表达为：

$$H(P) = \lim_{K\to\infty} G_K$$

然后，基于附录中的证明，对于所有K≥1，都满足G*{K+1}* ≤ *G*{k}，我们可以推导出，

$$G_1 \geq G_2 \geq \cdots \geq \lim_{K\to\infty} G_K = H(P)$$

我们观察到对于大规模文本处理任务，增加上下文长度往往能降低交叉熵或困惑度，这是反映**LLMs在捕获更广泛的上下文信息后能够进行更有效的逻辑推理及语义理解的现象。**因此，在困惑度分块实验中，最大化长文本序列的输入到LLMs中，期望收获更显著的性能提升。

数据集与指标

进行了针对四个基准、对比Meta-Chunking和多个基线方法的一系列问答（QA）数据集的全面评估，重点关注中文和英文语言，并覆盖多个指标，如答案的正确性、事实性以及检索文本的召回率。CRUD基准（Lyu等，2024）是包含单跳、两跳和三跳问题的中文数据集，使用包括BLEU系列、ROUGE-L和BERTScore在内的指标进行评估。我们使用RAGBench基准中的CUAD数据集，采用与CRUD相同的评估指标。MultiHop-RAG基准（Tang & Yang）评估召回率，采用如Hits@系列、MAP@ 10等指标，并利用八个中英文数据集进行单跳和多跳QA的评估，基于F1和ROUGE-L指标进行评估。

基线

主要将Meta-Chunking与两类方法进行比较，即基于规则的分块和动态分块，注意后者包含语义相似性模型和LLMs。原始的基于规则的方法简单地将长文本分为固定长度的块，忽视句子边界。然而，Llama_index方法则提供了一种更为细致的方式，在维护句子边界的同时，确保每段的标记数量接近预设阈值。另一方面，相似性分块使用句子嵌入模型进行文本的语义相似性分割，有效地将高度相关的句子聚集在一起。LumberChunker也在适应文本上下文和结构方面具有长处。

实验设置

主要使用Qwen2-1.5B、Internlm2-1.8B、Baichuan2-7B和Qwen2-7B进行Meta-Chunking。

此外，还探讨了元分块对较小模型的适用性和性能，使用了三个不到1B的模型：Pythia-0.16B、Pythia-0.41B和Qwen2-0.5B。

对于较长的文本，我们采用KV缓存的策略，以在PPL计算期间维持句子之间的逻辑连贯性，同时避免GPU内存溢出。数据集文本分割使用NVIDIA H800进行，评估使用NVIDIA GeForce RTX 3090。为了控制变量，我们在每个数据集中维护了各种分块方法的一致块长。

主要结果

对比基线方法。系统性地评估了四种基线方法的性能，重点关注其中三种的结果，如表所示。

Dataset	2WikiMultihopQA		Qasper		MultiFieldQA-en		MultiFieldQA-zh		MultiHop-RAG			
Chunking Method	F1	Time	F1	Time	F1	Time	F1	Time	Hits@10	Hits@4	MAP@10	MRR@10
Baselines with rule-based or similarity-based chunking												
Original	11.89	0.21	9.45	0.13	29.89	0.16	22.45	0.06	0.6027	0.4523	0.1512	0.3507
Llama_index	11.74	8.12	10.15	5.81	28.30	6.25	21.85	5.53	0.7366	0.5437	0.1889	0.4068
Similarity Chunking	12.00	416.45	9.93	307.05	29.19	318.41	22.39	134.80	0.7232	0.5362	0.1841	0.3934
Margin Sampling Chunking based on different models												
Pythia-0.16B$_{sent.}$	13.14	478.91	9.15	229.68	31.19	273.10	-	-	0.6993	0.5069	0.1793	0.3773
Pythia-0.41B$_{sent.}$	11.86	926.29	9.76	498.46	29.30	545.15	-	-	0.7259	0.5596	0.1934	0.4235
Qwen2-0.5B$_{sent.}$	11.74	788.30	9.67	599.97	31.28	648.76	23.35	480.35	0.7162	0.5246	0.1830	0.3913
Qwen2-1.5B$_{sent.}$	11.18	1908.25	10.09	1401.30	32.19	1457.31	22.27	1081.64	**0.7805**	**0.6089**	**0.2106**	**0.4661**
Qwen2-7B$_{sent.}$	**13.22**	7108.37	10.58	5207.87	32.32	5316.62	23.24	4212.00	0.6993	0.5197	0.1794	0.3835
Qwen2-1.5B$_{chunk}$	11.30	2189.29	9.49	1487.27	32.81	1614.01	22.08	1881.15	0.7109	0.5517	0.1970	0.4252
Qwen2-7B$_{chunk}$	12.94	8781.82	**11.37**	5755.79	**33.56**	6287.31	**24.24**	5084.95	0.7175	0.5415	0.1903	0.4141
Perplexity Chunking based on different models												
Internlm2-1.8B$_{comb.}$	12.37	355.53	10.02	200.69	30.81	251.06	22.53	161.15	0.7237	0.5499	0.1897	0.4121
Qwen2-1.5B$_{comb.}$	13.32	190.93	9.82	122.44	31.30	136.96	22.57	107.94	**0.7366**	0.5570	0.1979	0.4300
Baichuan2-7B$_{comb.}$	12.98	858.99	**10.04**	569.72	**32.55**	632.80	**23.36**	569.72	0.7206	**0.5636**	**0.2048**	**0.4406**
Qwen2-7B$_{comb.}$	**13.41**	736.69	9.39	486.48	32.35	523.74	22.81	424.96	0.7215	0.5521	0.1967	0.4229

值得注意的是，第四个基线（LumberChunker和Qwen2-7B）在Qasper数据集上取得了10.65的得分和2883.43秒的分块时间，但在其他四个数据集上未能有效工作。这表明了此策略在适应具有7B参数及以下模型时的显著局限性。所提出的边缘采样分块表现出比基线任务更高的性能，如表所示。在首四个数据集的准确性评估中，7B模型通常表现更好。然而，在以召回为关键指标的MultiHop-RAG数据集中，1.5B模型表现更佳。此外，分析结果显示了数据处理颗粒度对性能的影响。通过比较使用单个句子和整个块作为输入单元的实验结果，可以看出，尽管两种方法均可有效提高任务性能，但处理整个块在大多数情况下通常能获得更好的性能。

如表所示，使用1.5B和7B参数规模的模型时，PPL分块在问答系统和信息检索中的性能实现了显著的改善。具体而言，相比基线任务，两种模型配置均在准确性和召回指标上显示出明显的提升。此外，Qwen2-1.5B在多个数据集上的表现优于Internlm2-1.8B，除了在Qasper数据集中略有差异。关于7B系列模型，实验结果显示Baichuan2-7B在大多数情况下表现良好，仅在2WikiMultihopQA和Hits @10上略逊于Qwen2-7B。

效率与准确性权衡。在边缘采样分块中，使用整个块作为输入可以提高性能，但也增加了数据处理时间。然而，此方法的分块时间类似于LumberChunker算法，两者都达到了难以用于实际应用的阈值范围，这突显了LLMs在处理分块任务方面的低效。相反，PPL分块展示了显著的优势，不仅在保持或接近由边缘采样分块提供的性能水平方面表现优异，还在处理效率上实现了重大飞跃。此外，其效率超过了其他文本分块技术，如基于相似性的分割和LumberChunker。以不同模型规模分析PPL分块的性能时，发现尽管7B模型的处理时间相比1.5B模型增加，两者均能够完成分块。**为了充分发挥模型的预测能力，推荐使用7B模型进行分块。**

弱者有多弱？作为一项基础任务，文本分块在使用LLMs（如GPT-4或Gemini）时消耗了大量的标记，这常常导致资源利用与任务效益之间的显著不平衡。因此，使用轻量级模型是一个实用的选择。由于此方法适用于大小模型，测试了1.5B和7B模型之外，还探索了更小的低于1B参数的模型。随着模型规模减小，文本分块任务的执行时间显著减少，反映了小模型在提升处理效率方面的优势。不过，这一优势往往伴随着性能折衷。此外，小模型在跨语言适应性方面的限制尤其突出。以Pythia模型为例，其当前版本主要关注于英文文本，使其难以直接应用于多语种文本分块。因此，**在追求高性能和高效率的双重目标时，中等规模模型（如1.5B参数水平的那些）展示了较为均衡的表现。**

重叠分块策略的影响

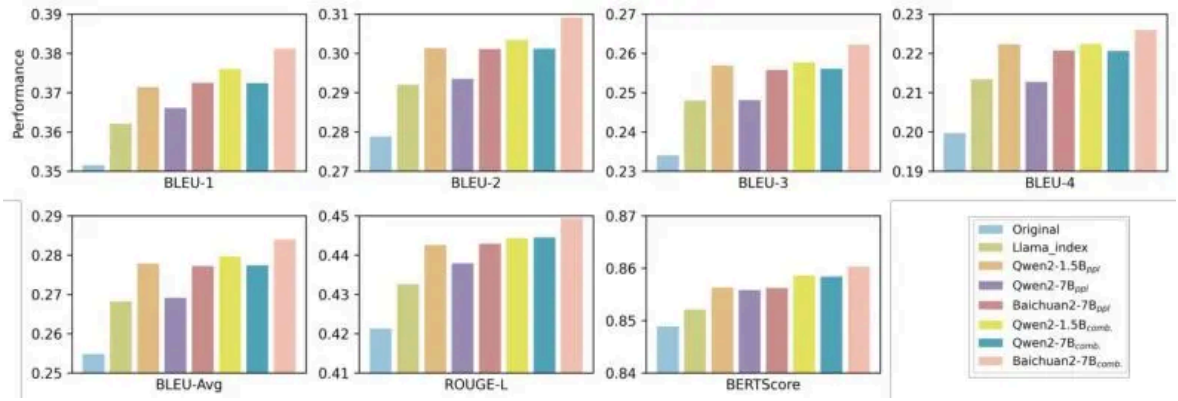
随着进一步深入探讨文本分块策略对复杂QA任务的影响，进一步研究了在使用重叠块时各种分块策略的表现。原始的分块重叠方法使用固定数量的字符从一个块的结尾与下一个块的开始进行重叠。Llama_index重叠方法在此基础上进一步考虑句子的完整性。**PPL分块重叠策略则动态地将PPL最小值所代表的句子同时分配给前后块，导致动态重叠。**这些方法通常产生的重叠长度大约为50个汉字。

Chunking Method	Overlap	BLEU-1	BLEU-2	BLEU-3	BLEU-4	BLEU-Avg	ROUGE-L	BERTScore
Single-hop Query								
Original	Fixed	0.3330	0.2641	0.2214	0.1881	0.2410	0.4060	0.8425
Llama_index	Dynamic	0.3326	0.2645	0.2214	0.1890	0.2413	0.4039	0.8439
Qwen2-1.5B$_{ppl}$	Dynamic	0.3592	0.2888	0.2435	0.2081	0.2644	0.4332	0.8555
Qwen2-7B$_{ppl}$	Dynamic	0.3582	0.2898	0.2450	0.2097	0.2657	0.4308	0.8548
Baichuan2-7B$_{ppl}$	Dynamic	0.3656	0.2952	0.2497	0.2143	0.2705	0.4393	0.8549
Two-hop Query								
Original	Fixed	0.2251	0.1300	0.0909	0.0689	0.1114	0.2579	0.8747
Llama_index	Dynamic	0.2223	0.1282	0.0896	0.0677	0.1099	0.2555	0.8732
Qwen2-1.5B$_{ppl}$	Dynamic	0.2295	0.1331	0.0934	0.0709	0.1143	0.2609	0.8700
Qwen2-7B$_{ppl}$	Dynamic	0.2312	0.1353	0.0949	0.0719	0.1162	0.2638	0.8751
Baichuan2-7B$_{ppl}$	Dynamic	0.2336	0.1350	0.0940	0.0710	0.1154	0.2650	0.8754
Three-hop Query								
Original	Fixed	0.2384	0.1268	0.0832	0.0602	0.1066	0.2546	0.8823
Llama_index	Dynamic	0.2331	0.1250	0.0825	0.0598	0.1049	0.2517	0.8796
Qwen2-1.5B$_{ppl}$	Dynamic	0.2453	0.1319	0.0881	0.0643	0.1114	0.2599	0.8808
Qwen2-7B$_{ppl}$	Dynamic	0.2447	0.1330	0.0891	0.0651	0.1122	0.2618	0.8817
Baichuan2-7B$_{ppl}$	Dynamic	0.2463	0.1324	0.0887	0.0651	0.1120	0.2596	0.8811

如表所示，PPL分块重叠策略在多跳QA场景中显示了尤为显著的性能表现。具体而言，除BERTScore指标外，PPL分块重叠方法在单跳任务上的性能提升了约2%-3%。在双跳和三跳任务中，尽管提高的速度稍有放缓，仍保持了0.3%-1%的稳定增益。此外，所有三个模型的性能随着模型参数规模的增加而呈现上升趋势。尽管1.5B模型在总体性能上略微落后于7B模型，但与传统分块方法相比，它仍展示了显著的改进，进一步验证了PPL分块的有效性。

两种PPL分块策略的比较分析

如图所示，对比了两种PPL分块策略：直接PPL分块和PPL分块与动态组合，这两者在CRUD数据集上均有效。通过实验分析，发现后者表现较优。这主要是因为**直接PPL分块可能导致块过长，而PPL分块与动态组合方法则有效维持了块长度和逻辑一致性。**此外，PPL分块在BLEU系列指标和ROUGE-L上相比传统分割方法实现了显著的性能提升。这表明，方法提高了生成文本与参考文本的准确性和流畅性。此外，此实验揭示了模型大小与性能之间的微妙平衡。具体而言，在该评估框架下，Qwen2-1.5B和Baichuan2-7B的表现接近，且在多个指标上常常超越Qwen2-7B模型。



长文本分块及策略选择

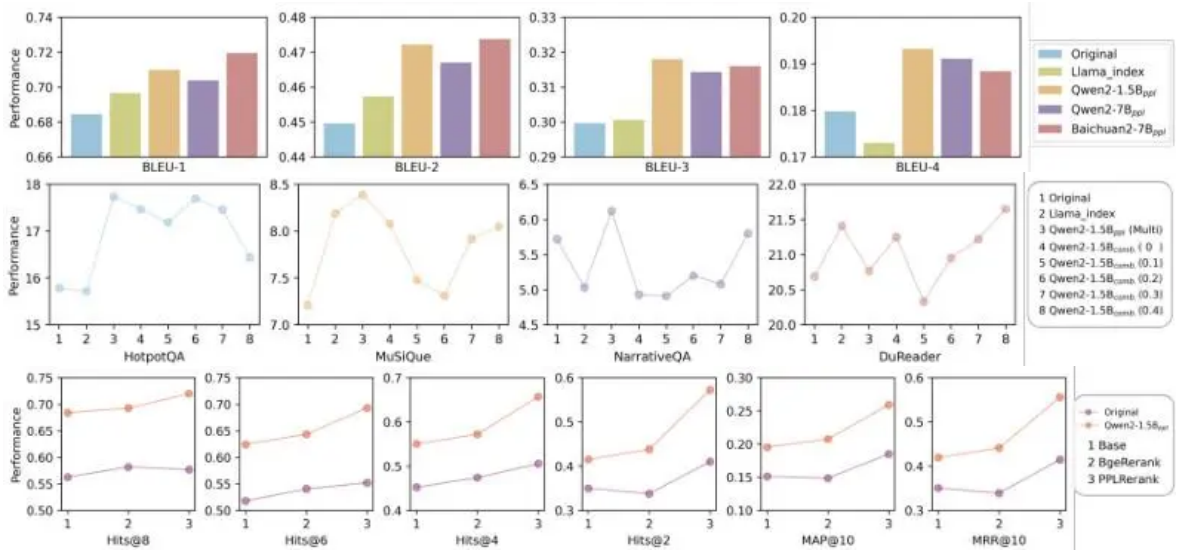
当处理较长文本时，采用KV缓存以在保持连贯性的前提下计算句子的PPL值，从而优化GPU内存的利用率和计算精度。通过使用CUAD数据集（平均长度11k），测试图4中展示的三个模型，其在BLEU系列指标上实现了显著改进。此外值得注意的是，Qwen2-1.5B和Baichuan2-7B的表现相近，这进一步证实了1.5B模型在应对不同长度的文本分块时能在性能和效率之间保持良好的平衡。

另一方面，深入探讨在LongBench的四个长文本QA数据集中的分块，并**对PPL分块的阈值进行了梯度实验（0到0.4，步长为0.1）**，以揭示PPL分布与分块效果之间的内在关系。当块长度较长时，具有动态组合的PPL分块表现更好。

实验结果表明，PPL分块的最佳配置依赖于文本的PPL分布：当PPL分布相对稳定时，选择较低阈值如在HotpotQA、MuSiQue和DuReader中将阈值设置为0更为合适；而当PPL分布出现大幅波动时，选择较高的阈值（如在NarrativeQA中将阈值设置为0.4）可以有效区分信息密度不同的段落，提升分块效果。因此，在进行基于PPL的分块时关键是综合考虑块长度和文本PPL分布两个因素，以确定相对最优的配置从而最大化性能。

重排序性能的分块方法探讨

为了探索分块策略对RAG系统的影响，评估了不同分块和重排序方法的组合。最初，用稠密检索器筛选出前10组相关文本，然后对比两种重排序策略：(1) BgeRerank方法，使用了bge-reranker-large模型，和(2) 使用Qwen2-1.5B模型的PPLRerank方法，采用Jiang等（2023）在粗粒度压缩章节中提到的重排序方法。



实验结果（见图6）表明，PPL分块和PPLRerank在所有指标上实现了最佳综合性能。进一步分析显示，与传统分块相比，PPL分块不仅能单独提供性能提升，还显著增强了后续重排序的效果。值得注意的是，尽管传统分块和重排序策略已经提供了性能改进，PPL分块结果在重排序中获得了更大的增益。例如，在 Hits @8指标中，原始分块下的PPLRerank提升了1.42%，而在PPL分块下该提升达到了3.59%。

结论

本文提出了元分块的概念及其实现策略，即边缘采样分块和困惑度分块，能够更精确地捕捉文本内在的逻辑结构，从而为优化RAG流程中的文本分割提供了一种有力的工具。为了在细粒度和粗粒度文本分割的有效性之间实现平衡，提出了结合元分块的动态组合方法来应对处理不同文本时的限制。通过对十一组数据集进行多个指标的全面评估，表明**元分块显著优于基于规则和相似度的分块，同时在性能、时间成本和计算成本方面优于当前的LLMs方法。**

rag 7 llm 16 ai 19

rag · 目录

上一篇
RAPTOR：树形组织的递归检索，大模型RAG性能提高20%

下一篇
GT&英伟达RankRAG：上下文排序和答案生成微调框架，医学数据集上与GPT-4表现相当