文本分块的天花板来了~

原创 热爱AI的 NLP前沿 2024年11月05日 11:55 湖北

论文笔记分享，标题：Meta-Chunking: Learning Efficient Text Segmentation via Logical Perception，代码开源:https://github.com/IAAR-Shanghai/Meta-Chunking/tree/386dc29b9cfe87da691fd4b0bd4ba7c352f8e4ed

切块切的好，对下游任务是很有帮助的。这个工作主要就是介绍2个文本分块策略，部分细节还是有点意思的。

如果已经在用bert做分类或者相似度了，可以考虑用Qwen-1.5B了，性能和耗时综合最优，如下图：

Table 1: Main experimental results are presented in five QA datasets. The first four datasets are sourced from LongBench. *sent.* indicates whether it is suitable to separate two sentences, while *chunk* signifies whether the latter sentence is appropriate to be merged with the preceding chunk. *comb.* refers to the process of first segmenting the text using PPL Chunking with a threshold of 0, followed by dynamic combination.

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

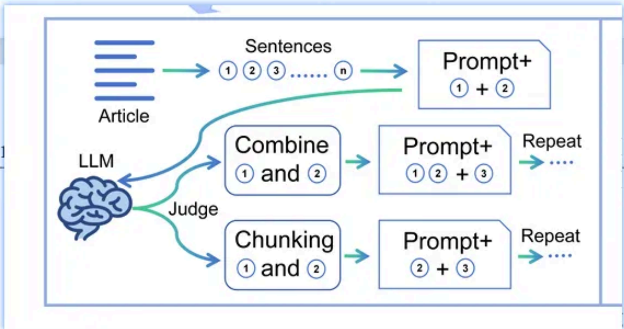
常见用模型除了用bert之类的做分类或者相似度区分，也有用大模型来做的，如LumberChunker，主要是靠prompt来实现，他的prompt如下

```
prompt = '''你是一位文本分块专家，将给定的文本进行分块处理。我需要你遵守以下4个条件：
1. 尽量使每个分块的大小保持在190个汉字左右。
2. 只能按照逻辑结构和语义结构进行文本分块。
3. 不要改变原文的词汇和结构。
4. 不要添加新的词汇或符号。
通过仅判断文本分块边界的方式，对原文进行文本分块，并逐个输出分块好的文本，分块之间用'---分块分隔符---'清晰分隔，其他任何解释都不要输出。如果你理解了，请对
```


下图对应的是这个工作中提到的第一种分块，称为Margin Sampling Chunking，大概思路是让LLM来做二分类，大模型输出是个词表的概率分布，这里他们做了一个对“是” 、 “否”的概率差，判断是否符合阈值。

$$\text{Margin}_M(x_i) = P_M\left(y = k_1 | \text{Prompt}(x_i, X')\right) - P_M\left(y = k_2 | \text{Prompt}(x_i, X')\right) \quad (1)$$

Preprint. Under



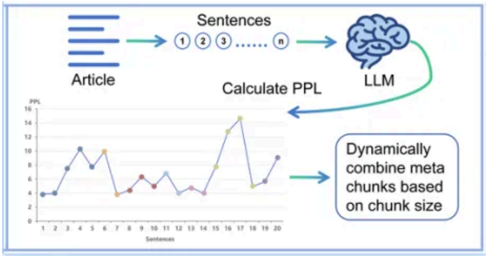
Chunking Prompt

This is a text chunking task. You are a text analysis expert. Please choose one of the following two options based on the logical structure and semantic content of the provided sentence:

1. Split *sentence1+sentence2* into *sentence1* and *sentence2* two parts;
2. Keep *sentence1+sentence2* unsplit in its original form;

Please answer 1 or 2.

下图为第二种分块，称为Perplexity Chunking，计算每个句子在上下文下的困惑度（如果困惑度高，说明模型对这段文本比较懵逼，所以不建议切分）。每次找到序列中困惑度最小的句子，并且如果这个句子前后2句都小于当前这个句子，那就可以切分了。算困惑度可以利用固定长度的kv-cache，来保证显存问题。



Hits@6	Hits@4	Hits@2	MAP@10	MRR@10
0.5180	0.4523	0.3499	0.1512	0.3507
0.5406	0.4741	0.3379	0.1486	0.3391
0.5521	**0.5055**	**0.4102**	**0.1849**	**0.4147**
0.6244	0.5503	0.4151	0.1954	0.4195
0.6435	0.5721	0.4381	0.2075	0.4413

Perplexity Chunking: Similarly, we split the text into sentences and use the model to calculate the PPL of each sentence x_i based on the preceding sentences:

$$\text{PPL}_M(x_i) = \frac{\sum_{k=1}^{K} \text{PPL}_M(t_k^i | t_{<k}^i, t_{<i})}{K} \quad (2)$$

where K represents the total number of tokens in x_i, t_k^i denotes the k-th token in x_i, and $t_{<i}$ signifies all tokens that precede x_i. To locate the key points of text segmentation, the algorithm further analyzes the distribution characteristics of $\text{PPL}_{seq} = (\text{PPL}_M(x_1), \text{PPL}_M(x_2), \ldots, \text{PPL}_M(x_n))$, particularly focusing on identifying minima:

$$\text{Minima}_{index}(\text{PPL}_{seq}) = \Big\{ i \;\Big|\; \min(\text{PPL}_M(x_{i-1}), \text{PPL}_M(x_{i+1})) - \text{PPL}_M(x_i) > \theta,$$
$$\textit{or}\ \text{PPL}_M(x_{i-1}) - \text{PPL}_M(x_i) > \theta\ \textit{and}\ \text{PPL}_M(x_{i+1}) = \text{PPL}_M(x_i) \Big\} \quad (3)$$

这些都可以后处理拼接，进行适量的拼接，让最终长度满足要求。

最终结果下来，Meta-Chunking能够有效提升RAG的单跳和多跳问答任务的性能。比如说，在2WikiMultihopQA数据集上，在时间消耗仅为相似性分块的45.8%的情况下，性能提升了1.32。