Transformer哪家强？Google爸爸辨优良！

原创 Zilong 夕小瑶的卖萌屋 2020-12-11 17:00

文：Zilong

2017年Attention is all you need横空出世，Transformer横扫机器翻译，隔年诞生的BERT建立在层层堆叠的Transformer之上，凭借这个平平无奇的Attention点乘模型一举刷新了各种沉积许久的榜单，一夜间仿佛不懂Transformer，都不敢说自己是NLPer了，曾经最心爱的RNN也瞬间黯然失色。

Transformer有着简易的的结构、SOTA的能力，搭配CUDA矩阵并行运算，不仅效果上比RNN胜出一筹，在运算效率上也遥遥领先。于是，无数论文纷至沓来，留给RNN的时间已经不多了。

然而，Transformer大厦上空依旧有着一朵乌云，让NLPer耿耿于怀，Transformer的核心结构——self attention归根到底依旧是二维矩阵运算，纵使抛弃了RNN中时序运算，得到了极大的运算效率的提升，但是计算机系本科生都知道，矩阵运算的复杂度是丑陋的$O(n^2)$。

$$Attn(Q,K,V)=softmax(\frac{QK^T}{\sqrt{d}})V$$

当Transformer遇到文档分类或者篇章理解之类的任务，随着文档长度增长，计算效率愈发难以忍受。为了解决运算复杂度的问题，NLPer提出了各种改进的版本，xformer家族日渐壮大，一个个都声称自己解决了Transformer的核心问题，试图挑战transformer老大哥的地位。其中不乏佼佼者如：

- Reformer（https://arxiv.org/abs/2001.04451）：通过Locality Sensitive Hashing类似于桶排序，将相似向量归为一类，计算同类向量之间的点积，复杂度为$O(nlog(n))$。
- Linformer（https://arxiv.org/abs/2006.04768）：认为注意力机制是低秩，信息集中在前k大的奇异值中，通过线性映射将复杂度降为$O(nk)$，当k足够小，模型接近线性时间。

- Sinkhorn Transformers（https://arxiv.org/abs/2002.11296.pdf）：将输入分块，并基于Sinkhorn对输入键值对进行重新排序，并应用基于块的局部注意力机制来学习稀疏模式。
- Performers（https://arxiv.org/abs/2009.14794）：通过正交随机特征算法加速注意力计算，改用Positive Orthogonal Random Features对常规softmax注意力进行鲁棒且无偏的估计。
- Synthesizers（https://arxiv.org/abs/2005.00743）：没有保持“token对token”形式的注意力形式，抛弃了原有注意力的动态特点，利用线性变换得到注意力矩阵。
- Linear Transformers（https://arxiv.org/abs/2006.16236）：通过使用核函数并且替换掉SoftMax，来简化Attention的计算过程，使复杂度降至$O(n)$。
- BigBird（https://proceedings.neurips.cc//paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html）：在Longformer的滑动窗口和膨胀窗口的基础上增加了Random attention，当前长序列建模的SOTA，刷新了QA和摘要的SOTA，同时也被证明是图灵完备的。

但是这些文章都是自说自话，用着各式各样的benchmarks、metrics，并没有一个统一的标准比一比Transformer哪家强。于是Google出面提出了Long Range Arena，试图从核心问题场景长文本分析入手，提出评价模型的6个标准、6大任务，逐一比较各个新兴xformer和原始Transformer的表现。

论文题目：

Long Range Arena: A Benchmark for Efficient Transformers

论文链接：

https://arxiv.org/abs/2011.04006

Arxiv访问慢的小伙伴也可以在**【夕小瑶的卖萌屋】**订阅号后台回复关键词**【1211】**下载论文PDF~

6个标准

贴心如Google，纵使坐拥海量资源，依旧心系贫下中农，时时刻刻担心抱着CPU炼丹的码农跑不了他的代码，于是LRA严于律己，树立了6个标准，确保LRA标准适用范围足够广泛。

1. 通用性：所有Transformer都能使
2. 简易性：无需数据增强、预训练等繁琐的准备步骤
3. 挑战性：任务足够难，人人都90%+就没意思了（~~能卷起来~~）
4. 长输入：**Long** Range Arena，输入自然要长一点，测试场景就是长输入下的表现
5. 多方面：方方面面都需要考察到，如长距离依赖、泛化能力等等
6. 轻计算：“妈妈再也不用担心我没有工业级显卡了”

6个任务

Google上先抛出了严格的6个标准，然后将准备好的任务娓娓道来。

- **Long ListOps**

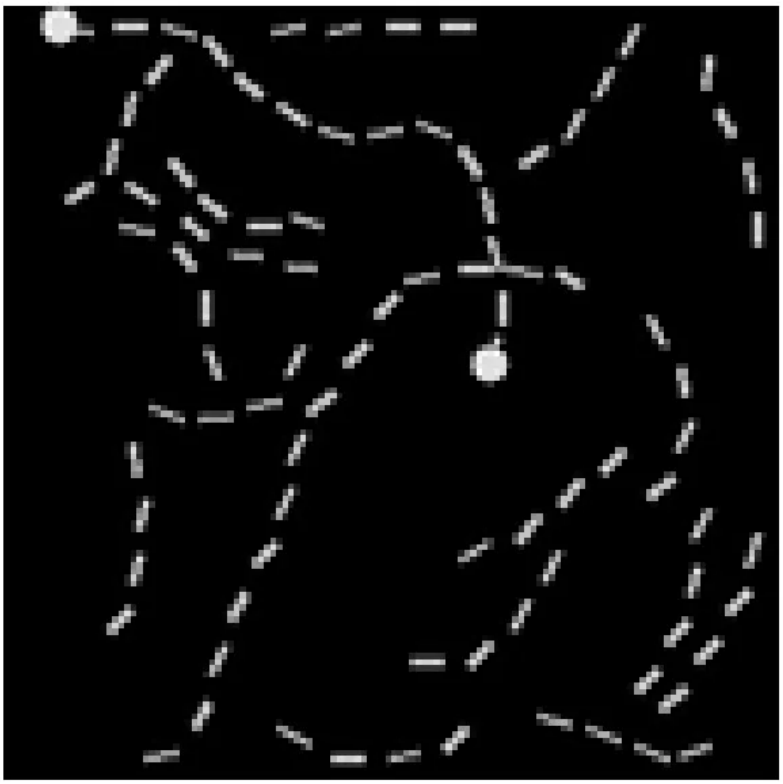
  INPUT: [MAX 4 3 [MIN 2 3] 1 0 [MEDIAN 1 5 8 9, 2]] **OUTPUT:** 5

  这个任务看起来神似前缀表达式，考虑 `max` 、 `min` 、 `median` 、 `sum_mod` 四种运算外带括号形成的hierarchical structure，考察xformer对长序列层次结构的理解能力。

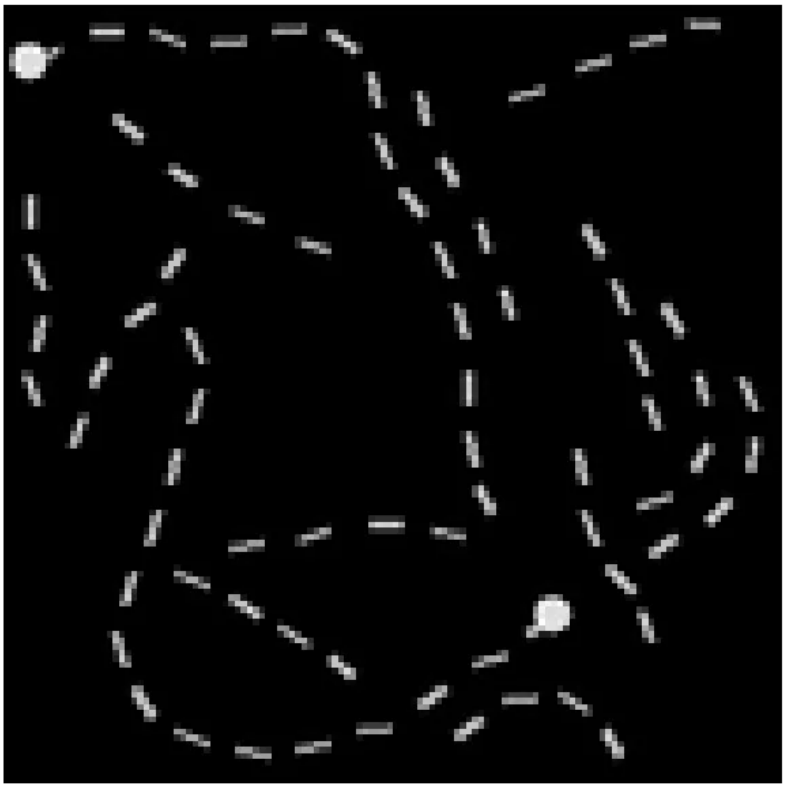
- **Byte-level Text Classification**、**Byte-level Document Retrieval**这两个任务主要关注对长文本的概括能力，测试xformer能否提取到长文本的足够信息量用于分类和匹配，值得注意的是，Google选取了**Byte-level**的输入，即字符级别的输入，轻松构造出长达4k的输入。

- **Image Classification on Sequences of Pixels**Google还企图将Transformer用于CV的任务中，这个任务将$N \times N$的图片拉成N^2的像素序列，当作文本去做分类任务。因为输入直接抹去了二维信息，这个任务不仅考察了xformer对序列特征的捕捉能力，同时考察了对层次结构的感知力。

  PathFinder (Long-Range Spatial Dependency)、**PathFinder-X (Long-Range Spatial Dependencies with Extreme Lengthts)**



(a) A positive example.



(b) A negative example.

最后两个任务依然是建立在图片之上，给定图片上两个圆点和若干条曲线，需要模型判断，这两个圆点是否被某一条曲线连接，xformer的输入依然是将2维图片拉长成一条1维的像素序列。而第二个任务PathFinder-X，意思是“格外的长”，图片大小变为128×128，于是序列长度达到16384，文章说就是想看看同一个问题，序列变长，会不会变难，xformer会不会处理不了，从结果来看，确实都处理不了，结果勉强达到随机分类的50%。

结果！

结果终于揭晓了，究竟Google LRA眼中Transformer哪家强呢？Google不仅报告了各个xformer的表现，同时还分析了时间和空间的消耗。

Model	ListOps	Text	Retrieval	Image	Pathfinder	Path-X	Avg
Transformer	36.37	64.27	57.46	42.44	71.40	FAIL	54.39
Local Attention	15.82	52.98	53.39	41.46	66.63	FAIL	46.06
Sparse Trans.	17.07	63.58	**59.59**	**44.24**	71.71	FAIL	51.24
Longformer	35.63	62.85	56.89	42.22	69.71	FAIL	53.46
Linformer	35.70	53.94	52.27	38.56	76.34	FAIL	51.36
Reformer	**37.27**	56.10	53.40	38.07	68.50	FAIL	50.67
Sinkhorn Trans.	33.67	61.20	53.83	41.23	67.45	FAIL	51.39
Synthesizer	36.99	61.68	54.67	41.61	69.45	FAIL	52.88
BigBird	36.05	64.02	59.29	40.83	74.87	FAIL	**55.01**
Linear Trans.	16.13	**65.90**	53.09	42.34	75.30	FAIL	50.55
Performer	18.01	65.40	53.82	42.77	**77.05**	FAIL	51.41
Task Avg (Std)	29 (9.7)	61 (4.6)	55 (2.6)	41 (1.8)	72 (3.7)	FAIL	52 (2.4)

Table 1: Experimental results on Long-Range Arena benchmark. Best model is in boldface and second best is underlined. All models do not learn anything on Path-X task, contrary to the Pathfinder task and this is denoted by FAIL. This shows that increasing the sequence length can cause seriously difficulties for model training. We leave Path-X on this benchmark for future challengers but do not include it on the Average score as it has no impact on relative performance.

6个任务中，ListOps差距最为明显，Linear Transformer和Performer难以捕捉层次结构信息，比原始Transformer下降了20%左右。其他各个任务中，各有优劣，差距并没有ListOps任务明显。另外，对于PathFinder-X任务，所有xformer都无法得到满意的结果，说明序列长度过长的情况下，Transformer无法很好的理解序列信息。

Model	Steps per second				Peak Memory Usage (GB)			
	1K	2K	3K	4K	1K	2K	3K	4K
Transformer	8.1	4.9	2.3	1.4	0.85	2.65	5.51	9.48
Local Attention	9.2 (1.1x)	8.4 (1.7x)	7.4 (3.2x)	7.4 (5.3x)	0.42	0.76	1.06	1.37
Linformer	9.3 (1.2x)	9.1 (1.9x)	8.5 (3.7x)	7.7 (5.5x)	**0.37**	**0.55**	0.99	**0.99**
Reformer	4.4 (0.5x)	2.2 (0.4x)	1.5 (0.7x)	1.1 (0.8x)	0.48	0.99	1.53	2.28
Sinkhorn Trans	9.1 (1.1x)	7.9 (1.6x)	6.6 (2.9x)	5.3 (3.8x)	0.47	0.83	1.13	1.48
Synthesizer	8.7 (1.1x)	5.7 (1.2x)	6.6 (2.9x)	1.9 (1.4x)	0.65	1.98	4.09	6.99
BigBird	7.4 (0.9x)	3.9 (0.8x)	2.7 (1.2x)	1.5 (1.1x)	0.77	1.49	2.18	2.88
Linear Trans.	9.1 (1.1x)	9.3 (1.9x)	8.6 (3.7x)	7.8 (5.6x)	**0.37**	0.57	**0.80**	1.03
Performer	**9.5** (1.2x)	**9.4** (1.9x)	**8.7** (3.8x)	**8.0** (5.7x)	**0.37**	0.59	0.82	1.06

Table 2: Benchmark results of all Xformer models with a consistent batch size of 32 across all models. We report relative speed increase/decrease in comparison with the vanilla Transformer in brackets besides the steps per second. Memory usage refers to per device memory usage across each TPU device. Benchmarks are run on 4x4 TPU V3 Chips.

速度上，利用核函数的方法整体速度提升明显，表现最优的是Performer，不少模型提出时声称达到线性复杂度，但是在LRA上速度却没有明显提高，甚至有所下降，如Reformer，在4组不同长度输入的实验中，速度始终慢于原始Transformer。

考虑各个xformer的表现、速度、占用内存，排名情况如图所示：

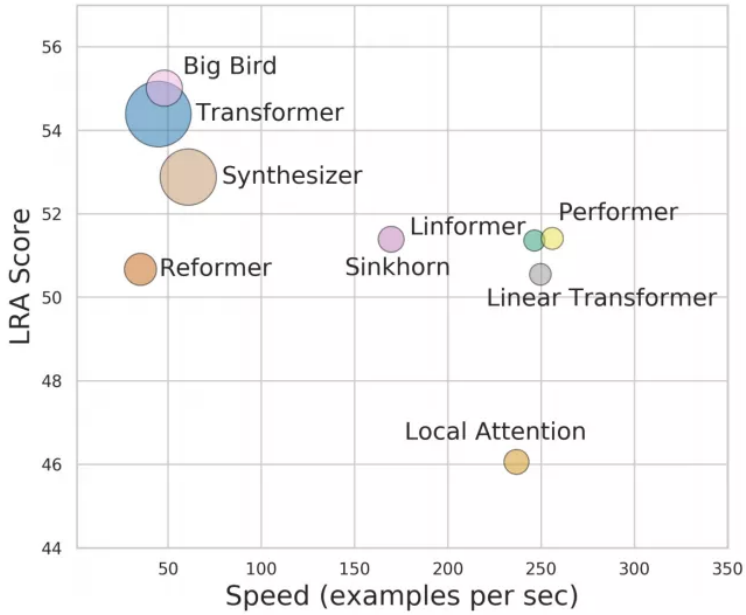


Figure 3: Performance (y axis), speed (x axis), and memory footprint (size of the circles) of different models.

可见xformer百花齐放，各有千秋，总之一句话"No one-size-fits-all"：

- 有的模型在个别任务上表现惊艳，却无法兼顾所有，例如Performers和Linear Transformers，虽然个别任务上相比原始Transformer有所下降，但是速度上提升极大
- 有的模型在各个任务上平均成绩出色，却一个第一也拿不到，例如BigBird，虽然号称线性复杂度，但是在实际测试环境中速度和原始Transformer差不多，性能却得到了一定的加强。
- 有的模型利用了复杂的技巧，但是速度却没有优势，性能可能有明显的下降，例如reformer、synthesizer。

权衡之下，似乎基于核函数的模型，如Performer、Linformer、linear transformer是兼顾各个方面的较优解。可能并不存在十全十美的模型，某个模型一统江湖的注定是乏味无趣的，根据自己任务合理选择模型，设计结构，最终得到针对当前问题的SOTA才是程序员朴实无华的快乐吧~

喜欢此内容的人还喜欢

我不看好data2vec这类多模态融合的研究

夕小瑶的卖萌屋