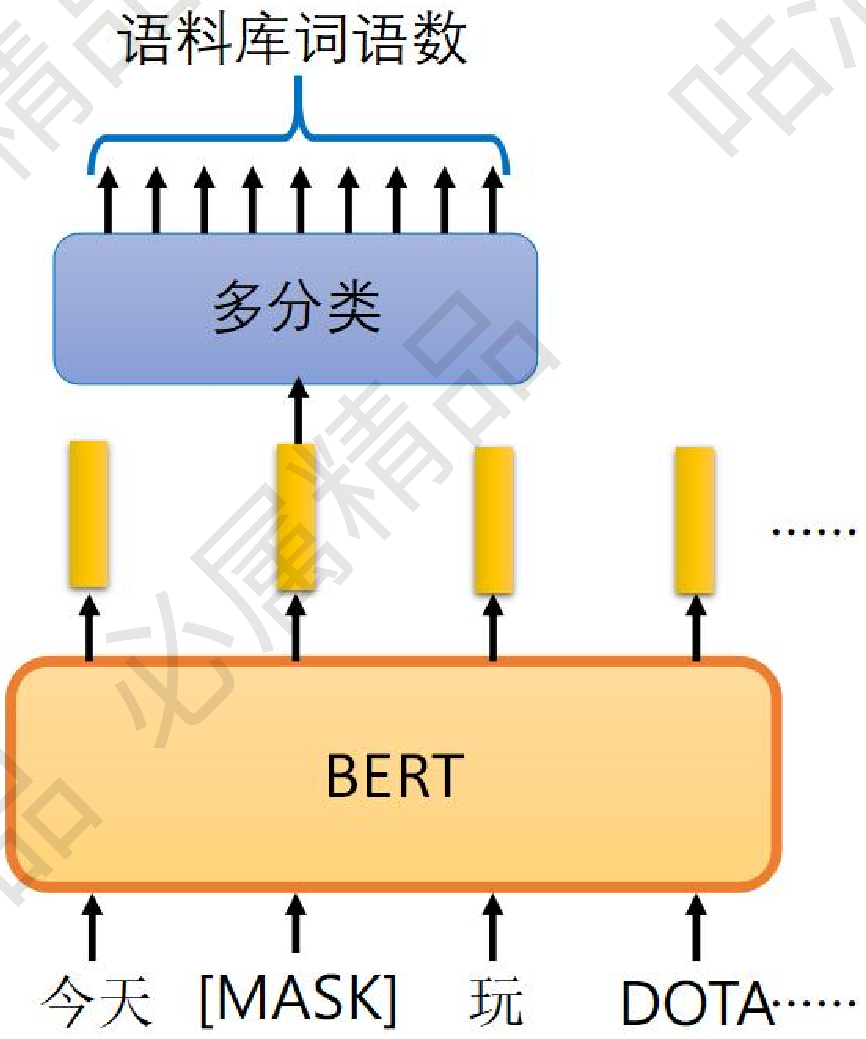
BERT

如何训练BERT

- 方法1：句子中有15%的词汇被随机mask掉
- 交给模型去预测被mask的家伙到底是什么
- 词语的可能性太多了，中文一般是字
- 如果BERT训练的向量好，那分类自然OK

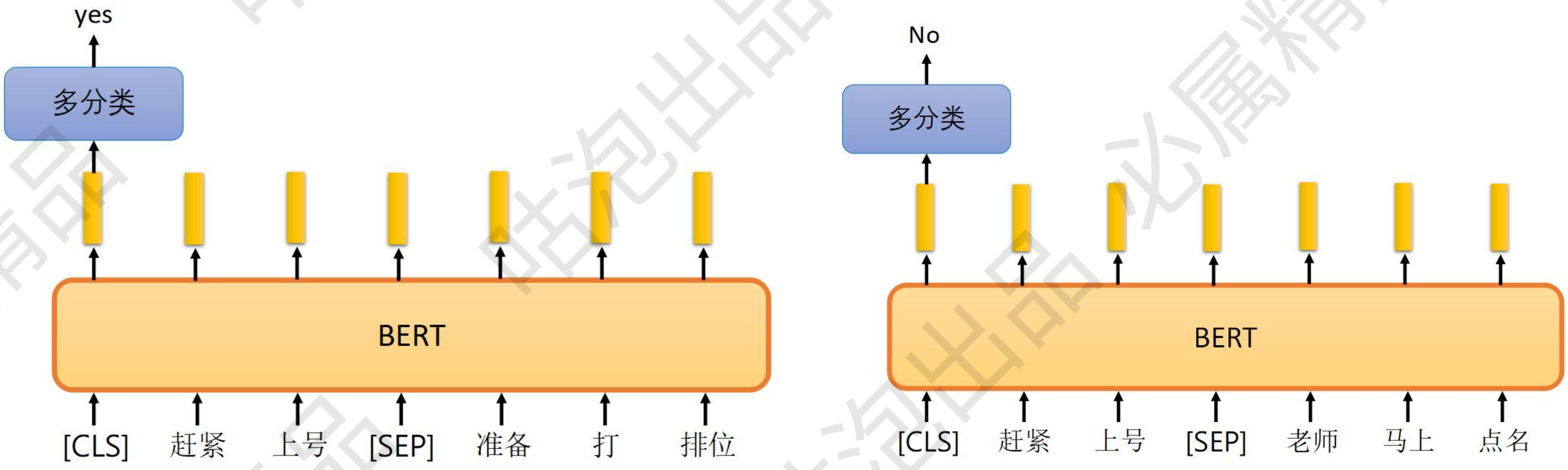


BERT

- 如何训练BERT

  - 方法2：预测两个句子是否应该连在一起

  - [seq]：两个句子之前的连接符，[cls]：表示要做分类的向量



ALBERT

- 要解决的问题（ A Lite BERT，轻量级的BERT）

- 从BERT开始NLP就一直强调一件事，要想效果好，模型就一定得大

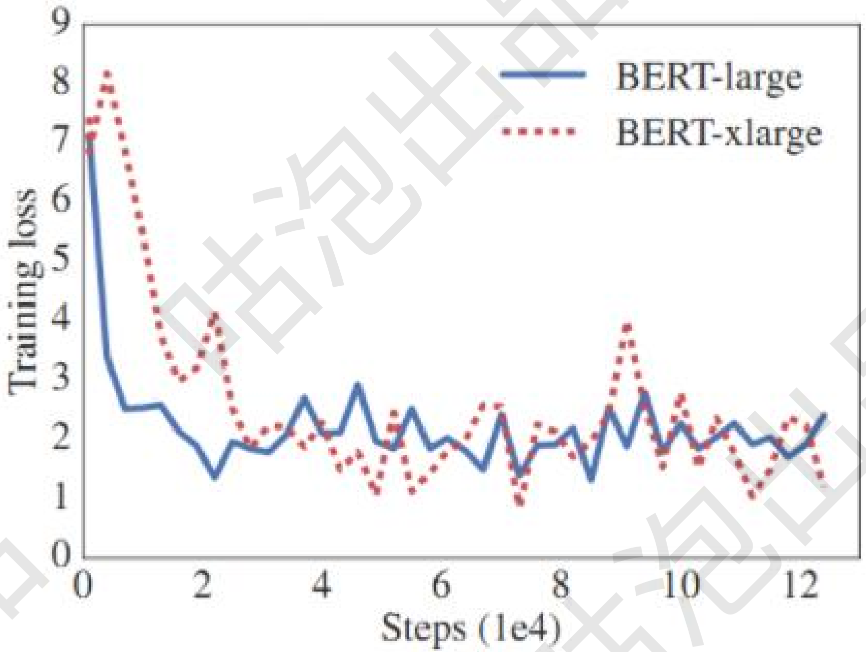
- 但是如果模型很大，权重参数就会非常多，训练是一个大问题(显存都装不下)

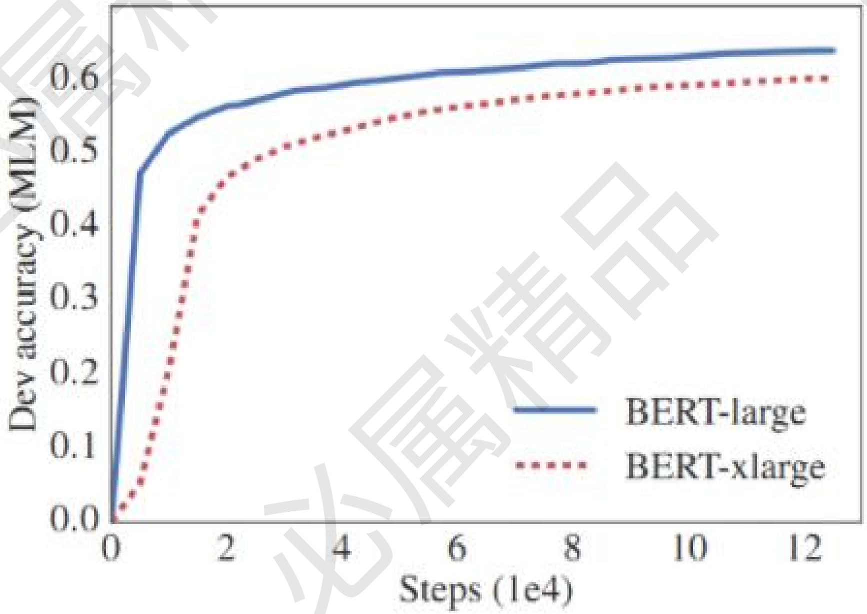
- 训练速度也是一个事，现在大厂模型都要以月为单位，速度巨慢

- 能不能简化下BERT，让他训练的更快更容易一些呢?
  （Transformer中Embedding占20%参数，Attetntion占80%）

ALBERT

- 隐层特征越多，效果一定越好吗？





Model	Hidden Size	Parameters	RACE (Accuracy)
BERT-large (Devlin et al., 2019)	1024	334M	72.0%
BERT-large (ours)	1024	334M	73.9%
BERT-xlarge (ours)	2048	1270M	54.3%

ALBERT

- 先记住这几个字母
  - E：词嵌入大小，也就是第一层Embedding后得到向量的维度
  - H：隐藏层大小，比如经过attention后得到768维向量
  - V：语料库中词的个数，比如咱们的字典中一共有20000个词
  - 然后咱们想一想，是不是Transformer中一般E都跟H一样大小的（768）

ALBERT

- 嵌入向量参数化的因式分解
  - 通过一个中介，将一层转换为两层，但是参数量可以大幅降低
  - 参数量：（V×H）降低到（V × E + E × H）
  - 此时如果H>>E，就达到了咱们的目的（E越小可能会效果越差）
  - 但是Embedding层只是第一步，Attention如何简化才是重头戏

ALBERT

- 嵌入向量参数化的因式分解

不同E值对结果的影响，E小一些会影响结果，但是不大

Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared	64	10M	88.7/81.4	77.5/74.8	80.8	89.4	63.5	79.0
	128	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1
	256	16M	88.8/81.5	79.1/76.3	81.5	90.3	63.4	79.6
	768	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8

ALBERT

- 跨层参数共享

  - 共享的方法有很多，ALBERT选择了全部共享，FFN和ATTENTION的都共享

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base E=768	all-shared	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8
	shared-attention	83M	89.9/82.7	80.0/77.2	84.0	91.4	67.7	81.6
	shared-FFN	57M	89.2/82.1	78.2/75.4	81.5	90.8	62.6	79.5
	not-shared	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base E=128	all-shared	12M	89.3/82.3	80.0/77.1	82.0	90.3	64.0	80.1
	shared-attention	64M	89.9/82.8	80.7/77.9	83.4	91.9	67.6	81.7
	shared-FFN	38M	88.9/81.6	78.6/75.6	82.3	91.7	64.4	80.2
	not-shared	89M	89.9/82.8	80.3/77.3	83.2	91.5	67.9	81.6

ALBERT

实验中还告诉我们的故事

层数一定越多越好嘛，目前来看是的

Number of layers	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
1	18M	31.1/22.9	50.1/50.1	66.4	80.8	40.1	52.9
3	18M	79.8/69.7	64.4/61.7	77.7	86.7	54.0	71.2
6	18M	86.4/78.4	73.8/71.1	81.2	88.9	60.9	77.2
12	18M	89.8/83.3	80.7/77.9	83.3	91.7	66.7	81.5
24	18M	**90.3/83.3**	**81.8/79.0**	83.3	91.5	**68.7**	**82.1**
48	18M	90.0/83.1	**81.8/78.9**	**83.4**	**91.9**	66.9	81.8

隐层特征要越大越好嘛，目前来看是的

Hidden size	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
1024	18M	79.8/69.7	64.4/61.7	77.7	86.7	54.0	71.2
2048	60M	83.3/74.1	69.1/66.6	79.7	88.6	58.2	74.6
4096	225M	**85.0/76.4**	**71.0/68.1**	**80.3**	**90.4**	**60.4**	**76.3**
6144	499M	84.7/75.8	67.8/65.4	78.1	89.1	56.0	74.0

RoBERTa

- Robustly optimized BERT approach
  - 基本就是说训练过程可以再优化优化
  - 最核心的就是如何在语言模型中设计mask:

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

  - 动态mask光听感觉肯定都比静态的要强，也就是这篇论文的核心
  - 取消NSP任务（Next Sentence Prediction）后效果反而好

RoBERTa

✓ 优化点

BatchSize基本也是大家公认的：

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

用了更多的数据集，训练了更久，提升了一点效果

分词方式做了一点改进，让英文拆的更细致（与中文无关）

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**

RoBERTa

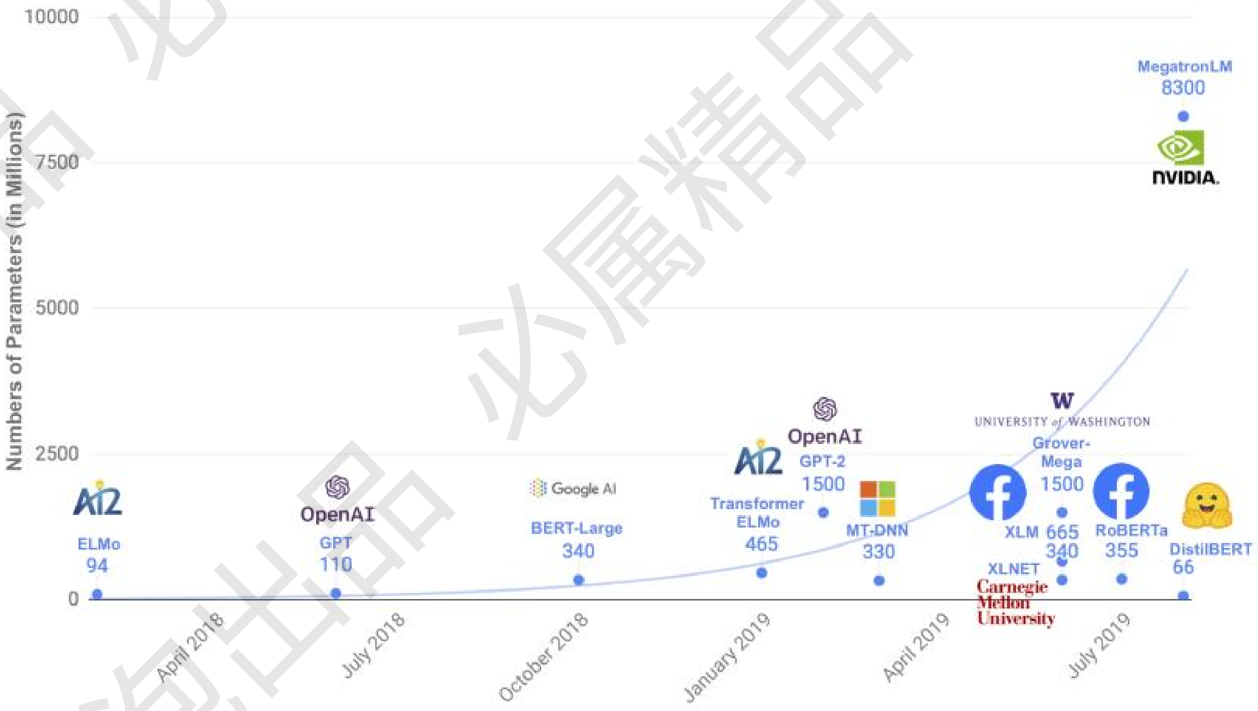
✓ RoBERTa-wwm

- wwm就是whole word mask，全词掩码
- 这个挺重要，1.我喜欢吃XXX正宗烤冷面；2.我喜欢吃哈X滨正宗烤冷面
- 对中文场景的训练来说肯定定wwm是比较重要的

说明	样例
原始文本	我喜欢吃西瓜，还喜欢跑步
原始Mask	我喜欢吃Mask瓜，还喜欢跑Mask
分词文本	我 喜欢 吃 西瓜 ， 还 喜欢 跑步
全词Mask	我喜欢吃Mask Mask，还喜欢Mask Mask

DistilBERT

- A distilled version of BERT: smaller,faster, cheaper and lighter
- 梦回2019，当年大家就发现模型越来越大这个趋势了
- 学术上一贯是可暴力出奇迹
- 工程上该怎么办，还得小一些
- 既小效果还得保障，怎么办呢



DistilBERT

- A distilled version of BERT: smaller,faster, cheaper and lighter
  - 差不多减少了40%的参数，主要是预测速度快
  - 蒸馏后效果还能保持97%，但是却被大大瘦身了

Model	# param. (Millions)	Inf. time (seconds)
ELMo	180	895
BERT-base	110	668
DistilBERT	66	410

Table 1: **DistilBERT retains 97% of BERT performance.** Comparison on the dev sets of the GLUE benchmark. ELMo results as reported by the authors. BERT and DistilBERT results are the medians of 5 runs with different seeds.

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3