学号：2019124044　　姓名：梅王智汇　　学院：伍伦贡联合研究院

RoBERTa：鲁棒优化的 BERT 预训练方法
翻译自：RoBERTa: A Robustly Optimized BERT Pretraining Approach

Yinhan Liu[§] Myle Ott[§] Naman Goyal[§] Jingfei Du[*§] Mandar Joshi[†] Danqi Chen[§]
Omer Levy[§] Mike Lewis[§] Luke Zettlemoyer[†§] Veselin Stoyanov[§]
[†]Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, WA
[§]Facebook AI

摘　要

语言模型的预训练带来了显着的性能提升，但是不同方法之间的仔细比较却具有挑战性。训练的计算量很大，这通常是在不同大小的私有数据集上进行，而且正如我们将要表明的，超参数的选择对最终结果有重大影响。我们提出了 BERT 预训练的复研究（Devlin 等，2019），该研究仔细衡量了许多关键超参数和训练数据量的影响。我们发现 BERT 的训练不足，并且可以匹配或超过其发布的每个模型的性能。我们最好的模型在 GLUE，RACE 和 SQuAD 上获得了最优的结果。这些结果突出了以前被忽视的设计的重要性，并引起了人们对最近报道改进的来源的质疑。我们发布我们的模型和代码。

1　介绍

自训练方法，例如 ELMo（Peters 等，2018），GPT（Radford 等，2018），BERT（Devlin 等，2019），XLM（Lample 和 Conneau，2019）和 XLNet（Yang 等，2019）取得了显着的性能提升，但要确定方法的哪些方面贡献最大可能是具有挑战性的。训练在计算上是代价较大的，限制了可以完成的调整量，并且经常使用大小不同的私有训练数据来进行，从而限制了我们衡量建模优化效果的能力。

我们提出了 BERT 预训练的复研究（Devlin 等，2019），其中包括对超参数调整和训练集大小的影响的仔细评估。我们发现 BERT 的训练不足，并提出了一种改进的方法来训练 BERT 模型（我们称为 RoBERTa），该模型可以匹敌或超过所有后 BERT 方法的性能。我们的修改很简单，其中包括：（1）在更多数据上以更大的批次训练模型更长的时间；（2）删除下一句预测目标；（3）对更长序列的训练；（4）动态改变应用于训练数据的遮罩模式。我们还收集了一个与其他私有数据集大小相当的大型新数据集（CC-NEWS），以更好地控制训练集大小的效果。

当控制训练数据时，我们改进的训练程序会改进 GLUE 和 SQuAD 上已发布的 BERT 结果。经过更多数据训练后，我们的模型在 GLUE 公开排行榜上得分为 88.5，于 Yang 等得分为 88.4 的报告（2019）匹敌。我们的模型针对 9 个 GLUE 任务中的 4 个建立了新的最新技术：MNLI，QNLI，RTE 和 STS-B。我们还将 SQuAD 和 RACE 上的最新结果进行匹配。总体而言，我们重新确定了 BERT 的遮罩语言模型训练目标可与最近提出的其他训练目标（例如受干扰的自回归语言建模（Yang 等，2019））竞争。

总而言之，本文的贡献是：(1) 提出了一组

重要的 BERT 设计和训练策略，并介绍了可以改善下游任务性能的替代方法；（2）我们使用新的数据集 CC-NEWS，并确认使用更多数据进行预训练可进一步提高下游任务的性能；（3）我们的训练改进表明，在正确的设计选择下，遮罩语言模型预训练与所有其他最近发布的方法相比具有竞争力。我们发布了我们的模型，在 PyTorch 中实现了预训练和微调代码（Paszke 等，2017）。

2 背景

在本节中，我们简要概述了 BERT（Devlin 等，2019）的预训练方法以及我们将在下一节中通过实验测试的一些训练选择。

2.1 配置

BERT 将两个段（token 序列）$x_1,...,x_N$ 和 $y_1,...,y_M$ 的串联作为输入。段通常包含一个以上的自然句子。这两个段作为单个输入序列呈现给 BERT，并用特殊的记号分隔它们：$[CLS],x_1,...,x_N,[SEP],y_1,...,y_M,[EOS]$。$M$ 和 N 被约束为 $M+N<T$，其中 T 是控制训练期间最大序列长度的参数。

该模型首先在庞大的无标签文本语料库上进行了预训练，然后使用最终任务标记的数据进行了微调。

2.2 架构

BERT 使用了如今无处不在的变换器架构（Vaswani 等，2017），我们将不对其进行详细介绍。我们使用具有 L 层的变换器架构。每个块都使用 A 自感知头和隐藏维 H。

2.3 训练目标

在预训练期间，BERT 使用两个目标：遮罩语言建模和下一句预测。

2.3.1 遮罩语言模型（MLM）

选择输入序列中 token 的随机样本，并将其替换为特殊 token[$MASK$]。MLM 的目标是预测掩盖 token 时的交叉熵损失。BERT 统一选择 15% 的输入 token 以进行替换。在所选 token 中，有 80% 被替换为 [$MASK$]，10% 则保持不变，还有 10% 被随机选择的词典 token 代替。

在最初的实现中，随机遮罩和替换从头开始执行一次，并在训练期间保存，尽管在实践中，数据是重复的，所以每个训练语句的遮罩并不总是相同的（参见第 4.1 节）。

2.3.2 下句预测（NSP）

NSP 是二进制分类损失，用于预测原始文本中两个段是否紧随其后。正样例是通过从文本语料库中选取连续的句子来创建的。负样例是通过将不同文档中的句段配对而创建的。正样本和负样本均以相同的概率采样。

NSP 目标旨在提高下游任务的性能，例如自然语言推理（Bowman 等，2015），这需要对句子对之间的关系进行推理。

2.4 优化

BERT 使用 Adam（Kingma 与 Ba，2015）使用以下参数进行了优化：$\beta_1=0.9,\beta_2=0.999,\epsilon=1e-6$ 和 L_2 权重系数为 0.01。在最初的 10,000 步中，学习率被加热到 $1e-4$ 的峰值，然后线性衰减。BERT 的所有层和注意权重都具有 0.1 的下降，并具有 GELU 激活功能（Hendrycks 和 Gimpel，2016）。对模型进行 $S=1,000,000$ 更新的预训练，其中小批量包含 $B=256$ 个最大长度为 $T=512$ 个 token 的序列。

2.5 数据

BERT 接收了 BOOKCORPUS（Zhu 等，2015）和英语 WIKIPEDIA 的组合训练，总共有

16GB 的未压缩文本。

3 实验配置

在本节中，我们将描述用于 BERT 研究的实验装置。

3.1 实现

我们在 FAIRSEQ 中重新实现 BERT（Ott 等，2019）。我们主要遵循第 2 节中给出的原始 BERT 优化超参数，除了峰值学习速率和预热步数（针对每个设置分别调整）之外。我们还发现训练对 Adam epsilon 术语非常敏感，并且在某些情况下，对其进行调整后可以获得更好的性能或更高的稳定性。类似地，我们发现设置 $\beta_2 = 0.98$ 可以提高大批量训练时的稳定性。

我们使用最多 $T = 512$ 个 token 的序列进行预训练。与 Devlin 等（2019）不同。我们不会随机注入短序列，并且对于前 90% 的更新，我们不会以减少的序列长度进行训练。我们只训练全长序列。

我们在 DGX-1 机器上使用混合精度浮点算术进行训练，每台机器都配有由 Infiniband 互连的 8Œ32GB Nvidia V100 GPU（Micikevicius 等，2018）。

3.2 数据

BERT 样式的预训练至关重要地依赖于大量文本。Baevski 等于 2019 年证明，增加数据大小可提高最终任务性能。有一些在比原始 BERT 更大，更多样化的数据集上进行的努力（Radford 等，2019; Yang 等，2019; Zellers 等，2019）。不幸的是，并非所有其他数据集都可以公开发布。在我们的研究中，我们专注于收集尽可能多的数据以进行实验，从而使我们能够根据每个比较结果来匹配数据的整体质量和数量。

我们考虑五个大小和域不同的英语语料库，总计超过 160GB 的未压缩文本。我们使用以下文本语料库：

- BOOKCORPUS（Zhu 等，2015）加上英文 WIKIPEDIA。这是用于训练 BERT 的原始数据。（16 GB）
- CC-NEWS，我们是从 CommonCrawl News 数据集的英语部分中收集的（Nagel, 2016）。数据包含 2016 年 9 月至 2019 年 2 月之间爬取的 6300 万篇英语新闻文章（过滤后为 76GB）。
- OPENWEBTEXT（Gokaslan 和 Cohen, 2019）是 Radford 等描述的 WebText 语料库的开源复本（2019）。文本是从 Reddit 上共享的 URL 中提取的至少有 3 个赞的 Web 内容。（共 38GB）
- STORIES 是 Trinh 和 Le（2018）中引入的数据集，其中包含 CommonCrawl 数据的子集，这些数据经过过滤以匹配 Winograd 模式的故事风格。（共 31GB）

3.3 评估

在完成之前的工作之后，我们使用以下三个基准评估了针对下游任务的预训练模型。

3.3.1 GLUE

通用语言理解评估（GLUE）基准（Wang 等，2019）是 9 个数据集的集合，用于评估自然语言理解系统。任务被划分为单句分类或句子对分类任务。GLUE 的组织者提供了训练和开发数据的划分，以及提交服务器和排行榜，使参与者可以根据个人提供的测试数据评估和比较其系统。

对于第 4 节中的复研究，我们在相应的单任务训练数据上对预训练的模型进行了微调（即没有多任务训练或集合）后，报告了开发集的结果。我们的微调程序遵循原始 BERT 论文（Devlin 等，2019）。

在第 5 节中，我们还报告了从公共排行榜获得的测试结果。这些结果取决于几个特定于

任务的修改，我们将在 5.1 节中进行介绍。

3.3.2 SQuAD

斯坦福问答数据集（SQuAD）提供了一段上下文和一个问题。任务是通过从上下文中提取相关范围来回答问题。

我们评估了 SQuAD 的两个版本：V1.1 和 V2.0（Rajpurkar 等，2016,2018）。在 V1.1 中，上下文始终包含一个答案，而在 V2.0 中，某些问题未在提供的上下文中回答，这使任务更具挑战性。

对于 SQuAD V1.1，我们采用与 BERT 相同的跨度预测方法（Devlin 等，2019）。对于 SQuAD V2.0，我们添加了一个附加的二进制分类器来预测问题是否可以解决，我们通过对分类和跨度损失项求和来共同进行训练。在评估期间，我们仅预测分类为可回答的对上的跨度指数。

3.3.3 RACE

考试的理解力（RACE）（Lai 等，2017）任务是一个大规模的阅读理解数据集，包含 28,000 多个段落和将近 100,000 个问题。数据集是从中国的英语考试中收集的，这些考试是针对中学生和高中生设计的。在 RACE 中，每个段落都与多个问题相关联。对于每个问题，任务是从四个选项中选择一个正确的答案。与其他流行的阅读理解数据集相比，RACE 具有更长的上下文，并且需要推理的问题比例非常大。

4 训练过程分析

本节探讨并量化了哪些选择对于成功地预训练 BERT 模型很重要。我们保持模型架构不变。具体来说，我们首先训练与 BERT_{BASE} 具有相同配置的 BERT 模型（$L = 12, H = 768, A = 120$，110M 参数）。

遮罩	**SQuAD 2.0**	**MNLI-m**	**SST-2**
参考	76.3	86.3	92.8
我们的实现			
静态	78.3	84.3	92.5
动态	78.7	84.0	92.9

表 1: BERT_{BASE} 的静态和动态掩码之间的比较。我们报告 SQuAD 的 F1，MNLI-m 和 SST-2 准确性。报告的结果是 5 个随机初始化（种子）的中位数。参考结果来自 Yang 等（2019）

4.1 静态遮罩对比动态遮罩

如第 2 节所述，BERT 依赖于随机遮罩和预测 token。原始 BERT 实现在数据预处理期间执行了一次遮罩，从而产生了单个静态遮罩。为了避免在每个过程中 · 对每个训练实例使用相同的遮罩，将训练数据重复 10 次，以便在 40 个实例中以 10 种不同的方式对每个序列进行遮罩。因此，每个训练序列在训练过程中都用相同的遮罩四次出现。

我们将这种策略与动态遮罩进行了比较，在动态遮罩中，每次将序列输入模型时都会生成遮罩模式。当进行更多步骤或更大数据集的预训练时，这变得至关重要。

4.1.1 结果

表 1 比较了 Devlin 等发表的 BERT_{BASE} 结果与静态或动态遮罩重新实现（2019）。我们发现，使用静态遮罩进行的重新实现与原始 BERT 模型的性能类似，并且动态遮罩与静态遮罩具有可比性或稍好于静态遮罩。鉴于这些结果以及动态遮罩的其他效率优势，我们在其余实验中使用动态遮罩。

4.2 模型输入格式和下句预测

在原始 BERT 预训练过程中，模型观察到两个串联的文档段，它们是从同一文档（$p = 0.5$）或不同文档中连续采样的。除了遮罩的语言建模目标外，还训练该模型以通过辅助的下

模型	**SQuAD 1.1/2.0**	**MNLI-m**	**SST-2**	**RACE**
我们的实现（带 NSP 损失）				
段对	90.4/78.7	84.0	92.9	64.2
句对	88.7/76.2	82.9	92.1	63.0
我们的实现（不带 NSP 损失）				
静态	90.4/79.1	84.7	92.5	64.8
动态	90.6/79.7	84.7	92.7	65.6
BERT_{BASE}	88.5/76.3	84.3	92.8	64.3
$XLNet_{BASE}(K = 7)$	-/81.3	85.8	92.7	66.1
$XLNet_{BASE}(K = 6)$	-/81.0	85.6	93.4	66.7

表 2: 经过 BOOKCORPUS 和 WIKIPEDIA 预训练的基本模型的开发集结果。所有模型都经过了 1M 步的训练，批量大小为 256 个序列。我们报告 SQuAD 为 F1，MNLI-m，SST-2 和 RACE 为准确性。报告的结果是五个随机初始化（种子）的中位数。BERT_{BASE} 和 XLNetBASE 的结果来自 Yang 等（2019）。

一句预测（NSP）损失来预测观察到的文档片段是来自同一文档还是来自不同文档。

NSP 损失被认为是训练原始 BERT 模型的重要因素。Devlin 等（2019）观察到，删除 NSP 会损害性能，并且会在 QNLI，MNLI 和 SQuAD 1.1 上显着降低性能。但是，最近的一些工作质疑 NSP 损失的必要性（Lample 和 Conneau，2019 年; Yang 等，2019 年; Joshi 等，2019）。

为了更好地理解这种差异，我们比较了几种替代的训练格式：

- SEGMENT-PAIR + NSP：这遵循 BERT（Devlin 等，2019）中使用的原始输入格式，但 NSP 丢失。每个输入都有一对段，每个段可以包含多个自然语句，但是总的组合长度必须少于 512 个标记。
- SENTENCE-PAIR + NSP：每个输入都包含一对自然句子，它们从一个文档的连续部分或单独的文档中采样。由于这些输入明显少于 512 个 token，因此我们增加了批量大小，以便 token 的总数保持与 SEGMENT-PAIR + NSP 相似。我们保留 NSP 损失。
- FULL-SENTENCES：每个输入都包含从一个或多个文档中连续采样的完整句子，因此总长度最多为 512 个 token。输入内容可能会跨越文档边界。当我们到达一个文档的末尾时，我们开始从下一个文档中抽取句子，并在文档之间添加一个额外的分隔符。我们消除了 NSP 损失。
- DOC-SENTENCES：输入的构造与 FULL-SENTENCES 相似，不同之处在于它们可能不会跨越文档边界。在文档末尾附近采样的输入内容可能少于 512 个 token，因此在这种情况下，我们会动态增加批处理大小，以实现与 FULL SENTENCES 相似的 token 总数。我们消除了 NSP 损失。

结果

表 2 显示了四种不同设置的结果。我们首先比较 Devlin 等的原始 SEGMENT-PAIR 输入格式（2019）为 SENTENCE-PAIR 格式；两种格式都保留了 NSP 损失，但后者使用了单句。我们发现**使用单个句子会损害下游任务的性能**，我们推测这是因为该模型无法学习远程依赖关系。

接下来，我们将比较不损失 NSP 的训练和使用单个文档中的文本块进行训练的情况（DOC-SENTENCES）。我们发现，与 Devlin 等相反（2019），此设置优于最初发布的 BERT_{BASE} 结果，并且**消除了 NSP 损失匹敌或略微提高了下游任务性能**。原始 BERT 实现可能只删除了

损失项，同时仍保留 SEGMENT-PAIR 输入格式。

最后，我们发现来自单个文档的限制序列（DOC-SENTENCES）的效果比来自多个文档的打包序列（FULL-SENTENCES）的效果略好。但是，由于 DOC-SENTENCES 格式会导致批量大小可变，因此在其余的实验中我们将使用 FULL SENTENCES，以便与相关工作进行比较。

4.3 大批量训练

过去神经机器翻译的工作表明，适当提高学习率时，使用非常大的小批处理进行的训练可以提高优化速度和最终任务性能（Ott 等，2018）。最近的工作表明 BERT 也适合进行大批量训练（You 等，2019）。

Devlin 等（2019）最初训练 BERT_{BASE} 进行 1M 步，批量大小为 256 个序列。通过梯度累加，这在计算成本上等同于训练批量为 2K 序列的 125K 步或批量为 8K 的 31K 步。

在表 3 中，我们比较了 BERT_{BASE} 的困惑度和最终任务性能，因为我们增加了批次大小，并控制了训练数据的通过次数。我们观察到，大批量训练可以提高遮罩语言建模目标的复杂性以及最终任务的准确性。大批量也可以通过分布式数据并行训练进行并行化，在以后的实验中，我们使用 8K 序列进行训练。

值得注意的是，You 等（2019）训练 BERT 甚至具有更大的批量，最多 32K 序列。我们将进一步探索大批量训练对未来工作的局限性。

4.4 文本编码

字节对编码（BPE）（Sennrich 等，2016）是字符级和单词级表示的混合体，可以处理自然语言语料库中常见的大词汇。BPE 代替子词，依靠子词单元，这些子词是通过对训练语料库进行统计分析而提取的。

BPE 词汇量的大小通常在 10K-100K 子词单位之间。但是，在为大型和多样的公司（例如本工作中考虑的公司）建模时，unicode 字符可以占该词汇表的很大一部分。Radford 等（2019）引入了一种 BPE 的巧妙实现，它使用字节而不是 Unicode 字符作为基本子字单元。使用字节可以学习适度大小（50K 单位）的子词，该词仍可以对任何输入文本进行编码，而无需引入任何“未知”标记。

bsz	步数	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

表 3: 对于通过 BOOKCORPUS 和 WIKIPEDIA 训练的基本模型，批处理数据（pps）的保留训练数据（ppl）的困惑和开发集的准确性。我们调整每个设置的学习率（lr）。模型在数据（历元）上进行相同次数的传递，并且具有相同的计算成本。

最初的 BERT 实现（Devlin 等，2019）使用大小为 30K 的字符级 BPE 词汇表，该词汇表是在使用启发式合并规则对输入进行预处理之后才学习的。继 Radford 等（2019），我们取而代之的是使用更大的字节级 BPE 词汇表来训练 BERT，该词汇表包含 5 万个子词单元，而无需对输入进行任何额外的预处理或标记化。这分别为 BERT_{BASE} 和 BERT_{LARGE} 添加了大约 15M 和 20M 的附加参数。

早期的实验表明在这些编码之间只有细微的差别，BPE 在某些任务上的最终任务性能稍差（Radford 等，2019）。不管怎样，我们认为通用编码方案的优势胜过性能上的细微差别，在其余的实验中都使用这种编码。这些编码的更详细的比较留给以后的工作。

5 RoBERTa

在上一节中，我们建议对 BERT 预训练过程进行修改，以提高最终任务的性能。现在，我们汇总这些改进并评估它们的综合影响。我们将此配置称为 **RoBERTa**，以实现稳健（**R**obust）

模型	数据	bsz	步数	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
带有 BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ 额外数据	160GB	8K	100K	94.0/87.7	89.3	95.6
+ 预训练较长	160GB	8K	300K	94.4/88.7	90.0	96.1
+ 预训练更长	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
BERT_{LARGE}						
带有 BOOKS+ WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$XLNET_{LARGE}$						
带有 BOOKS+ WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ 额外数据	126GB	2K	500K	94.5/88.8	89.8	95.6

表 4: 当我们预训练更多数据（16GB160GB 文本）并预训练更长的数据（100K300K500K 步长）时，RoBERTa 的开发设置结果。每行都累积了以上行的改进。RoBERTa 符合 BERTLARGE 的体系结构和训练目标。BERTLARGE 和 XLNetLARGE 的结果来自 Devlin 等（2019）和 Yang 等（2019）。有关所有 GLUE 任务的完整结果，请参见附录。

优化（**o**ptimized）的 BERT 方法（**a**pproach）。具体来说，RoBERTa 接受了动态屏蔽（第 4.1 节），无 NSP 丢失的全检测（第 4.2 节），大型迷你批处理（第 4.3 节）和较大的字节级 BPE（第 4.4 节）的训练。

此外，我们调查了先前工作中未充分强调的另外两个重要因素：（1）用于预训练的数据，以及（2）通过数据的训练次数。例如，最近提出的 XLNet 架构（Yang 等，2019）使用的数据比原始 BERT（Devlin，2019）多了近 10 倍。还以八倍大的批量大小对其进行了训练，优化步骤的数量减少了一半，因此与 BERT 相比，预训练中的序列数量增加了四倍。

为了帮助将这些因素的重要性与其他建模选择（例如，训练前的目标）区分开来，我们首先遵循 BERTLARGE 体系结构（L = 24，H = 1024，A = 16，355M 参数）训练 RoBERTa。我们在 Devlin 等使用（2019）的 BOOK-CORPUS 加上 WIKIPEDIA 数据集上预训练了 100K 步，使用 1024 个 V100 GPU 预训练模型大约一天。

结果

我们在表 4 中给出了结果。在控制训练数据时，我们观察到 RoBERTa 相对于最初报告的 BERTLARGE 结果有很大的改进，印证了我们在第 4 节中探讨的设计选择的重要性。

接下来，我们将这些数据与第 3.2 节中描述的三个附加数据集结合起来。我们使用与以前相同的训练步骤数（100K）对组合数据进行 RoBERTa 训练。我们总共预训练了 160GB 以上的文本。我们观察到所有下游任务的性能都有进一步的提高，从而验证了数据大小和多样性在预训练中的重要性。

最终，我们对 RoBERTa 进行了更长的预训练，将预训练步骤的数量从 100K 增加到 300K，然后进一步增加到 500K。我们再次观察到下游任务性能的显着提高，并且在大多数任务中，300K 和 500K 步进模块的性能优于 XLNetLARGE。我们注意到，即使是经过最长训练的模型也似乎并不太过适合我们的数据，并且可能会从其他训练中受益。

在本文的其余部分，我们在三种不同的基准上评估我们最好的 RoBERTa 模型：GLUE，

SQuaD 和 RACE。具体来说，我们认为 RoBERTa 在第 3.2 节中介绍的所有五个数据集上训练了 500K 步。

5.1 GLUE 结果

当我们预训练更多数据（16GB→160GB 文本）并预训练更长的数据（100K→300K→500K 步长）时，RoBERTa 的开发设置结果每行都累积了以上行的改进。RoBERTa 符合 BERTLARGE 的体系结构和训练目标。BERTLARGE 和 XLNetLARGE 的结果来自 Devlin 等（2019）和 Yang 等（2019）。有关所有 GLUE 任务的完整结果，请参见附录。

在第二种设置（集合，测试）中，我们通过 GLUE 排行榜将 RoBERTa 与测试集上的其他方法进行了比较。虽然 GLUE 排行榜的许多内容取决于多任务微调，但我们**提交的内容仅取决于单任务微调**。对于 RTE，STS 和 MRPC，我们发现从 MNLI 单任务模型而不是基线预训练的 RoBERTa 开始进行微调很有帮助。我们探究了附录中描述的稍宽的超参数空间，并且每个任务在 5 至 7 个模型之间集成。

特定任务的修改

GLUE 任务中的两项需要特定于任务的微调方法，以获得竞争性的排行榜结果。

QNLI：GLUE 排行榜上的最新提交文件采用了 QNLI 任务的成对排名公式，其中从训练集中提取候选答案并相互比较，然后将单个（问题，候选）对分类为肯定（Liu 等，2019；Yang 等，2019）。这种表述显着简化了任务，但与 BERT 没有直接可比性（Devlin 等，2019）。在最近的工作之后，我们对测试提交采用了排名方法，但是为了与 BERT 直接比较，我们报告了基于纯分类方法的开发集结果。

WNLI：我们发现提供的 NLI 格式数据很难使用。取而代之的是，我们使用 Super-GLUE（Wang 等，2019）中重新格式化的 WNLI 数据，该数据指示查询代词和所指对象的跨度。我们使用 Kocijan 等的边界排位损失来调整 RoBERTa（2019）。对于给定的输入句子，我们使用 spaCy（Honnibal 和 Montani，2017）从句子中提取其他候选名词短语，并对模型进行微调，以使其对正向指称短语分配的得分高于任何生成的负向候选短语。这种表述的一个副作用是，我们只能使用积极的训练示例，而其占所提供的训练示例的一半以下。

结果

我们将结果显示在表 5 中。在第一个设置（单任务，开发）中，RoBERTa 在所有 9 个 GLUE 任务开发集中都获得了最新的结果。至关重要的是，RoBERTa 使用与 BERTLARGE 相同的遮罩语言建模预训练目标和体系结构，但始终优于 BERTLARGE 和 XLNetLARGE。与我们在这项工作中所展示的更普遍的细节（例如数据集大小和训练时间）相比，这提出了关于模型架构和训练前目标的相对重要性的问题。

在第二种设置（装配，测试）中，我们将 RoBERTa 提交给 GLUE 排行榜，并在 9 项任务中的 4 项中取得了最新的成绩，并且是迄今为止最高的平均分数。这尤其令人兴奋，因为 RoBERTa 不依赖于多任务微调，这与大多数其他顶级提交文件不同。我们希望未来的工作可以通过合并更复杂的多任务微调程序来进一步改善这些结果。

5.2 SQuAD 结果

与过去的工作相比，我们为 SQuAD 采用的方法要简单得多。特别是，虽然 BERT（Devlin 等，2019）和 XL-Net（Yang 等，2019）都使用其他 QA 数据集来扩充其训练数据，但我们仅使用提供的 SQuAD 训练数据对 RoBERTa 进行了微调。杨等（2019）还采用了自定义的逐层学习率计划来微调 XLNet，而我们对所有层使用相同的学习率。

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
单任务单模型开发										
BERT$_{LARGE}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet$_{LARGE}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
装配测试（来自 2019 年七月 25 日的排行榜）										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

表 5: GLUE 上的结果。所有结果均基于 24 层架构。BERTLARGE 和 XLNetLARGE 结果来自 Devlin 等（2019）和 Yang 等（2019）。RoBERTa 在开发集上的结果是五次运行的中位数。测试集上的 RoBERTa 结果是单任务模型的集合。对于 RTE，STS 和 MRPC，我们从 MNLI 模型而不是基线预训练模型开始进行微调。从 GLUE 排行榜获得平均值。

模型	SQuAD 1.1		SQuAD 2.0	
	EM	F1	EM	F1
单模型开发，无数据扩充				
BERT$_{LARGE}$	84.1	90.9	79.0	81.8
XLNet$_{LARGE}$	**89.0**	94.5	86.1	88.8
RoBERTa	88.9	**94.6**	**86.5**	**89.4**
单模型测试（2019.7.25）				
XLNet$_{LARGE}$			86.3§	89.1§
RoBERTa			86.8	89.8
XLNet + SG-Net Verifier			**87.0**§	**89.9**§

表 6: SQuAD 上的结果。§ 表示取决于其他外部训练数据的结果。RoBERTa 在开发和测试设置中仅使用提供的 SQuAD 数据。BERTLARGE 和 XLNet-LARGE 结果来自 Devlin 等（2019）和 Yang 等（2019 年）

模型	准确率	中阶	高阶
单模型测试（2019.7.25）			
BERT$_{LARGE}$	72.0	76.6	70.1
XLNet$_{LARGE}$	81.7	85.4	80.2
RoBERTa	**83.2**	**86.5**	**81.3**

表 7: RACE 测试集上的结果。BERTLARGE 和 XL-NetLARGE 结果来自 Yang 等（2019）

对于 SQuAD v1.1，我们遵循与 Devlin 等相同的优化程序（2019）。对于 SQuAD v2.0，我们还对给定问题是否可以回答进行分类。我们通过对分类和跨度损失项求和，与跨度预测器一起训练该分类器。

结果

我们在表 6 中给出了结果。在 SQuAD v1.1 开发集上，RoBERTa 与 XLNet 的最新集匹配。在 SQuAD v2.0 开发套件上，RoBERTa 设置了新的最新技术，与 XLNet 相比分别提高了 0.4 点（EM）和 0.6 点（F1）。

我们还将 RoBERTa 提交给 SQuAD 2.0 公开排行榜，并评估其相对于其他系统的性能。大多数顶级系统都基于 BERT（Devlin 等，2019）或 XLNet（Yang 等，2019）构建，这两个系统都依赖于其他外部训练数据。相反，我们的提

交没有使用任何其他数据。

我们的单一 RoBERTa 模型的表现优于单一模型提交中的所有模型，并且是不依赖数据扩充的模型中得分最高的系统。

5.3 RACE 结果

在 RACE 中，系统提供了一段文本，一个相关的问题以及四个候选答案。需要系统对四个候选答案中的哪个进行正确分类。

我们通过将每个候选答案与相应的问题和段落连接起来来修改 RoBERTa。然后，我们对这四个序列中的每个序列进行编码，并将所得的 $[CLS]$ 表示形式通过完全连接的层，该层用于预测正确的答案。我们会截断长度超过 128 个 token 的问题 - 答案对，并在需要时截断段落，以使总长度最多为 512 个 token。

RACE 测试集的结果列于表 7。RoBERTa 在中阶和高阶环境中均达到了最新水平。

6 相关工作

预培训方法已针对不同的培训目标进行了设计，包括语言建模（Dai 和 Le，2015；Peters 等，2018；Howard 和 Ruder，2018），机器翻译（McCann 等，2017）和掩盖语言建模（Devlin 等，2019; Lample 和 Conneau，2019）。

最近的许多论文都为每个最终任务使用了微调模型的基本方法（Howard 和 Ruder，2018 年; Radford 等，2018），并使用某种掩盖的语言模型目标进行了预训练。但是，较新的方法通过多任务微调（Dong 等，2019），合并实体嵌入（Sun 等，2019），跨度预测（Joshi 等，2019）和改进了性能。自回归预训练的多种变体（Song 等，2019; Chan 等，2019; Yang 等，2019）。

通常还可以通过在更多数据上训练更大的模型来提高性能（Devlin 等，2019; Baevski 等，2019; Yang 等，2019; Radford 等，2019）。我们的目标是复制，简化和更好地调整 BERT 的训练，作为更好地了解所有这些方法的相对性能的参考点。

7 结论

在预训练 BERT 模型时，我们会仔细评估许多设计决策。我们发现，通过对模型进行较长时间的训练，使用更多的批次处理更多的数据，可以显着提高性能。删除下一个情感预测目标；培训更长的顺序；并动态更改应用于训练数据的掩蔽模式。我们改进的预训练程序（称为 RoBERTa）可在 GLUE，RACE 和 SQuAD 上获得最新的结果，而无需为 GLUE 进行多任务微调或为 SQuAD 进行其他数据调整。这些结果说明了这些以前被忽略的设计决策的重要性，并表明 BERT 的预训练目标与最近提出的替代方案仍然具有竞争力。

我们还使用一个新颖的数据集 CC-NEWS，并在以下位置发布我们的模型和代码以进行预训练和微调：https://github.com/pytorch/fairseq