BERT: 预训练的深度双向 Transformer 语言模型

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language

摘要

本文介绍一种称之为 *BERT* 的新语言表征模型，意为来自变换器的双向编码器表征量（*Bidirectional Encoder Representations from Transformers*）。不同于最近的语言表征模型（*Peters* 等人，*2018; Radford* 等人，*2018*），*BERT* 旨在基于所有层的左、右上下文语境来预训练深度双向表征。因此，预训练的 *BERT* 表征可以仅用一个额外的输出层进行微调，进而为很多任务（如问答和语言推理）创建当前最优模型，无需对任务特定架构做出大量修改。

BERT 的概念很简单，但应用效果很强大。它刷新了 *11* 个自然语言处理（*NLP*）任务的当前最优结果，包括将 *GLUE* 基准提升至 *80.5%*（*7.7%* 的绝对改进）、将 *MultiNLI* 的准确率提高到 *86.7%*（*4.6%* 的绝对改进），以及将 *SQuADv1.1* 问答测试 *F1* 的得分提高至 *93.2* 分（*1.5* 分的绝对提高），将 *SQuADv2.0* 测试的 *F1* 得分提高至 *83.1* 分（*5.1* 分的绝对提高）。

1. 介绍

语言模型预训练已被证明可有效改进许多自然语言处理任务（Dai 和 Le，2015；Peters 等人，2018a；Radford 等人，2018；Howard 和 Ruder，2018）。这些任务包括句子级任务，句子级任务如自然语言推理 inference（Bowman 等人，2015; Williams 等人，2018）和释义 paraphrasing（Dolan 和 Brockett，2005），旨在通过整体分析来预测句子之间的关系；以及词符级任务，如命名实体识别（Tjong Kim Sang 和 De Meulder，2003）和 SQuAD 问题回答（Rajpurkar 等人，2016），其中模型需要在词符级别生成细粒度输出。

将预训练语言表征应用于下游任务有两种现有策略：基于特征（feature-based）和微调（fine-tuning）。基于特征的方法，例如 ELMo（Peters 等人，2018a），使用特定于任务的架构，其包括将预训练表征作为附加特征。微调方法，例如 Generative Pre-trained Transformer（OpenAIGPT 生成型预训练变换器）（Radford 等人，2018），引入了最小的任务特定参数，并通过简单地微调预训练参数在下游任务中进行训练。在以前的工作中，两种方法在预训练期间共享相同的目标函数，它们使用单向语言模型来学习通用语言表征。

我们认为，当前技术严重制约了预训练表征的能力，特别是对于微调方法。其主要局限在于标准语言模型是单向的，这限制了可以在预训练期间使用的架构类型。例如，在 OpenAI GPT，作者们用一个从左到右的架构，其中每个词符只能注意变换器自注意层中的前验词符（Vaswani 等人，2017）。这种局限对于句子层面任务而言是次优选择，对于词符级任务的方法，则可能是毁灭性的。在这种任务中应用基于词符级微调法，如 SQuAD 问答（Rajpurkar 等人，2016），结合两个方向语境至关重要。

在本论文，我们通过提出 BERT 模型：来自变换器的双向编码器表征量（Bidirectional Encoder Representations fromTransformers），改进了基于微调的方法。BERT 通过提出一个新的预训练目标：“屏蔽语言模型”（masked language model，MLM），来自 Cloze 任务（Taylor，1953）的启发，来解决前面提到的单向性约束。该屏蔽语言模型随机地从输入中屏蔽一些词符，并且，目标是仅基于该屏蔽词语境语

境来预测其原始词汇 ID。不像从左到右的语言模型预训练，该 MLM 目标允许表征融合左右两侧语境语境，这允许我们预训练一个深度双向变换器。除了该屏蔽语言模型，我们还引入了一个“下一句预测”（nextsentence prediction）任务，该任务联合预训练文本对表征量。我们的论文贡献如下：

，partopsep

- 我们证明了双向预训练对语言表征量的重要性。与 Radford 等人（2018）不同，其使用单向语言模型进行预训练，BERT 使用屏蔽语言模型来实现预训练的深度双向表征量。这也与 Peters 等人（2018）形成对比，其使用由独立训练的从左到右和从右到左 LM（语言模型）的浅层串联。
- 我们展示了预训练表征量能消除许多重型工程任务特定架构的需求。BERT 是第一个基于微调的表征模型，它在大量的句子级和词符级任务上实现了最先进的性能，优于许多具有任务特定架构的系统。
- BERT 推进了 11 项 NLP 任务的最高水平。因此，我们报告了广泛的 BERT 消融，证明我们模型的双向性质是最重要的新贡献。代码和预训练模型将在https://github.com/google-research/bert上提供。

2. 相关工作

预训练通用语言表征有很长历史，本节我们简要回顾中这些最常用的方法。

2.1. 基于特征的无监督方法

几十年来，学习可广泛应用的单词表示方法一直是一个活跃的研究领域，包括非神经网络方法（Brown 等人，1992；Ando 和 Zhang，2005；Blitzer 等人，2006）和神经网络方法（Mikolov 等人，2013；Pennington 等人，2014）。预训练的词嵌入是现代 NLP 系统不可或缺的一部分，与从零开始学习的嵌入相比有显着改进（Turian 等人，2010）。为了预训练词嵌入向量，已使用从左到右的语言建模目标（Mnih 和 Hinton，2009），以及在左右上下文中区分正确单词和错误单词的目标（Mikolov 等人，2013）。

这些方法已经推广到更粗粒度，例如句子嵌入（Kiros 等人，2015；Logeswaran 和 Lee，2018）或段落嵌入（Le 和 Mikolov，2014）。为了训练句子表示，以前的工作使用对候选的下一句进行评分排序（Jernite 等人，2017；Logeswaran 和 Lee，2018）的目标，给定前一个句子的表示从左到右生成下一个句子的单词（Kiros 等人，2015）或去噪自动编码器派生的目标（Hill 等人，2016）。

ELMo 及其前身（Peters 等人，2017，2018a）沿不同维度推广了传统单词嵌入的研究。他们结合从左到右和从右到左的语言模型提取上下文相关特征。每个词符的上下文表示是从左至右和从右至左表示的首尾相连。当将上下文词嵌入与现有的特定于任务的体系结构集成时，ELMo 改进了几种主要的 NLP 基准的最先进结果（Peters 等人，2018a），包括问题回答（Rajpurkar 等人，2016）、情感分析（Socher 等人，2013）和命名实体识别（Tjong Kim Sang 和 De Meulder，2003）。Melamud 等人（2016）提出通利用 LSTM 从左右上下文中预测单个单词的任务学习上下文表征。与 ELMo 相似，它们的模型是基于特征的并且不是深度双向的。Fedus 等人（2018）表明，cloze 任务可用于提高文本生成模型的鲁棒性。

2.2. 无监督微调方法

与基于特征的方法一样，在这个方向上的早期工作只是在未标记的文本上预训练单词的嵌入参数（Collobert 和 Weston，2008）。

最近，产生上下文词符表示的句子或文档编码器已经从未标记的文本中预训练，并针对有监督的下游任务进行微调（Dai 和 Le，2015；Howard 和 Ruder，2018；Radford 等人，2018）。这些方法的优点是几乎不需要从头学习参数。至少部分由于此优势，OpenAI GPT（Radford 等人，2018）在 GLUE 基准测试中许多句子级任务上获得了以前的最先进结果（Wang 等人，2018a）。从左到右的语言建模和自动编码器目标已被用于预训练此类模型（Howard 和 Ruder，2018；Radford 等人，2018；Dai 和 Le，2015）。

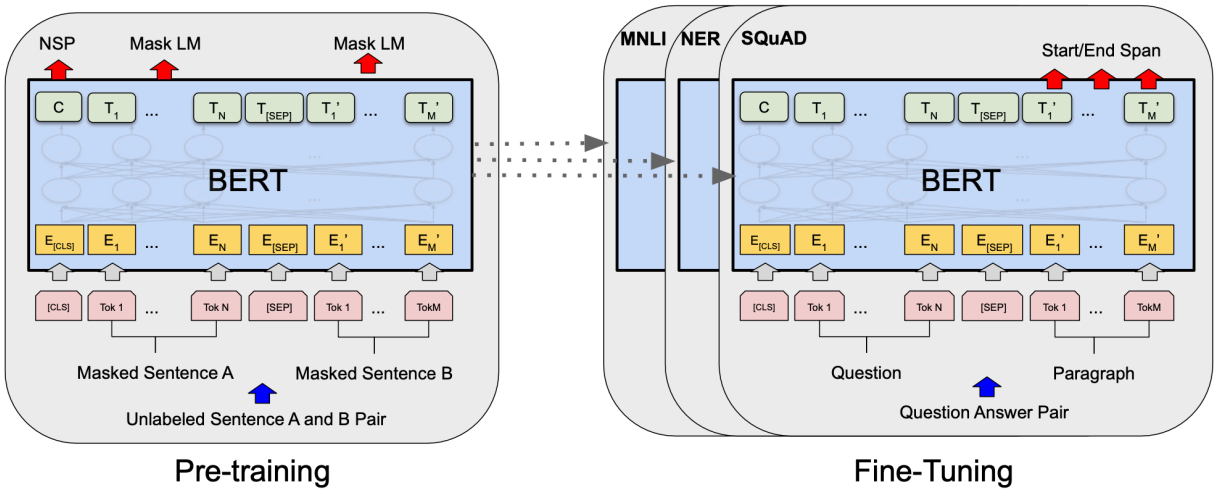

图 1. BERT 的总体预训练和微调程序。除了输出层，在预训练和微调中都使用相同的体系结构。相同的预训练模型参数用于初始化不同下游任务的模型。在微调期间，所有参数都将进行微调。[CLS] 是在每个输入示例前添加的特殊符号，而 [SEP] 是特殊的分隔符（例如，分隔问题/答案）。

2.3. 从监督数据迁移学习

也有工作显示了从大数据集监督任务的有效转移，如自然语言推理（Conneau 等人，2017）和机器翻译（McCann 等人，2017）。计算机视觉研究还证明了从大型预训练模型进行迁移学习的重要性，其中一个有效的方法是微调用 ImageNet 预训练的模型（Deng 等人，2009；Yosinski 等人，2014）。

3. BERT

我们将在本节中介绍 BERT 及其详细实现。我们的框架有两个步骤：预训练和微调。在预训练期间，通过不同的预训练任务在未标记的数据上进行模型训练。对于微调，首先使用预训练的参数初始化 BERT 模型，然后使用下游任务中的标记数据对所有参数进行微调。每个下游任务都有单独的微调模型，即使它们使用相同的预训练参数进行了初始化。图 1 中的问答示例将作为本节的运行示例。

BERT 的一个独有的特征是其跨不同任务的统一结构。预训练的结构和最终的下游结构之间的差异很小。

模型结构 BERT 的模型结构是一种多层 Transformer 编码器，它基于的原始实现的描述位于 Vaswani 等人（2017）并发布在 tensor2tensor 库中。因为 Transformer 的使用已经很普遍以及我们的实现与原始版本几乎相同，我们将省略模型结构的详尽背景说明并请读者参考 Vaswani 等人（2017）以及优秀的指南如 “The Annotated Transformer”。

在这项工作中，我们将网络层（即 Transformer 的网络模块）的数量表示为 L，将隐藏层大小表示为 H，并将自注意头的数量表示为 A。我们主要报告两种模型大小的结果：$\text{BERT}_{\text{BASE}}$（L=12，H=768，A=12，Total Parameters=110M）和 $\text{BERT}_{\text{LARGE}}$（L=24，H=1024，A=16，Total Parameters=340M）。

为了进行比较，选择 $\text{BERT}_{\text{BASE}}$ 具有与 OpenAI GPT 相同的模型大小。但是，至关重要的是，BERT Transformer 使用双向自关注，而 GPT Transformer 使用受限的自关注，其中每个词符只能关注其左侧的上下文。

输入/输出表示 为了使 BERT 处理各种下游任务，我们的输入可以用一个词符序列明确地表示单个句子和一对句子（例如 Question，Answer）。在整个工作中，“句子” 可以是连续文本的任意范围，而不是实际的语言句子。“序列” 是指 BERT 的输入词符序列，它可以是一个句子或两个句子封装在一起。

我们使用 WordPiece 嵌入（Wu 等人，2016），具

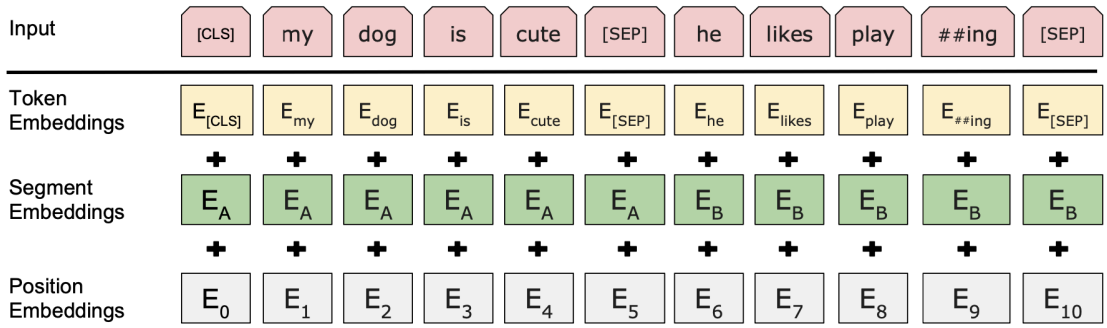

图 2. BERT 的总体预训练和微调程序。除了输出层，在预训练和微调中都使用相同的体系结构。相同的预训练模型参数用于初始化不同下游任务的模型。在微调期间，所有参数都将进行微调。[CLS] 是在每个输入示例前添加的特殊符号，而 [SEP] 是特殊的分隔符（例如，分隔问题/答案）。

有 30，000 个词符的词汇表。每个序列的第一个标记始终是特殊分类标记（[CLS]）。与此标记对应的最终隐藏状态用作分类任务的聚合序列表示。句子对封装在一起形成单个序列。我们通过两种方式区分句子。首先，我们使用特殊词符（[SEP]）将它们分开。其次，我们向每个词符添加一个学习型嵌入，表明它是属于句子 A 还是句子 B。如图 1 所示，我们将输入嵌入表示为 E，将特殊的 [CLS] 词符的最终隐藏向量表示为 $C \in \mathbb{R}^H$，第 i 个输入词符为的最终隐藏向量为 $T_i \in \mathbb{R}^H$。

对于给定的词符，其输入表示是通过将相应的词符嵌入、分段嵌入和位置嵌入相加来构造的。在图 2 中可以看到这种构造的可视化。

3.1. 预训练 BERT

不同于 Peters 等人（2018a）和 Radford 等人（2018），我们没有使用传统的从左到右或从右到左的语言模型对 BERT 进行预训练。相反，我们使用本节中描述的两个无监督任务对 BERT 进行预训练。该步骤显示在图 1 的左侧。

任务 #1：屏蔽 LM 直观地讲，我们有理由相信，一个深层的双向模型严格来说比从左到右的模型或从左到右和从右到左的浅层模型连接更强大。不幸的是，标准的条件语言模型只能从左至右或从右至左进行训练，因为双向条件会让每个词间接地"看到自己"，从而模型在多层上下文中可以十分容易地预测目标词。

为了训练深度双向表示，我们简单地随机屏蔽一定百分比的输入词符，然后预测这些屏蔽的词符。尽管此过程在文献中通常被称为 Cloze 任务（Taylor，1953），但我们将该过程称为“屏蔽 LM”（MLM）。这种情况下，和在标准 LM 中一样，与屏蔽词符相对应的最终隐藏向量被送入词汇表上的输出 softmax 中。在所有实验中，我们随机屏蔽每个序列中所有 WordPiece 词符的 15%。与去噪自动编码器（Vincent 等人，2008）不同，我们只预测被屏蔽的单词，而不重构整个输入。

尽管这可以使我们获得双向的预训练模型，但缺点是我们在预训练和微调之间造成了不一致，因为 [MASK] 词符不会在微调期间出现。为了缓解这种情况，实际上我们并不总是用 [MASK] 词符替换“被屏蔽”的单词。训练数据生成器随机选择词符位置的 15%进行预测。如果选择第 i 个词符，我们替换这第 i 个词符为（1）80%的时间用 [MASK] 词符（2）10%的时间用随机词符（3）10%的时间维持第 i 词符不变。然后，将用 T_i 以交叉熵损失预测原始词符。我们在附录 C.2 中比较了此过程的各种变体。

任务 #2：下一句预测（NSP） 许多重要的下游任务，例如问答（QA）和自然语言推理（NLI）是是基于理解两个句子之间的关系，而这不是语言建模直接捕获的。为了训练能够理解句子关系的模型，我们预训练了可以从任何单语语料库很容易生成的

二值下一个句子预测任务。具体来说，当为每个预训练样本选择句子 A 和 B 时，50%的时间 B 是紧随其后的实际下一个句子 A（标记为 IsNext），50%的时间是语料库中的随机句子（标记为 NotNext）。如图 1 所示，C 用于下一句预测（NSP）。尽管非常简单，我们在 5.1 节中展示此任务的预训练对 QA 和 NLI 都非常有益。NSP 任务与表示学习目标很接近，参见 Jernite 等人（2017）和 Logeswaran 和 Lee（2018）。但是，在之前的工作中，只有句子嵌入被传输到下游任务，而 BERT 传输所有参数以初始化最终任务模型的参数。

预训练数据 预训练过程在很大程度上遵循有关语言模型预训练的现有文献。对于预训练语料库，我们使用 BooksCorpus（800 万个单词）（Zhu 等人，2015）和英语 Wikipedia（25 亿个单词）。对于 Wikipedia，我们仅提取文本段落，而忽略列表、表格和标题。为了提取长的连续序列，使用文档级语料库而不是洗乱的句子级语料库如 Billion Word Benchmark（Chelba 等人，2013）至关重要。

3.2. 微调 BERT

微调很简单，因为 Transformer 中的自我关注机制允许 BERT 通过交换适当的输入和输出来建模许多下游任务（无论它们涉及单个文本还是文本对）。对于涉及文本对的应用，常见的模式是在应用双向交叉注意之前对文本对进行独立编码，例如 Parikh 等人（2016）；Seo 等人（2017）。BERT 使用自我注意机制来统一这两个阶段，因为使用自我注意编码连接的文本对行实际上包括了两个句子之间的双向交叉注意。

对于每个任务，我们只需将特定于任务的输入和输出送入 BERT，并端到端微调所有参数。在输入处，来自预训练的句子 A 和句子 B 类比于（1）paraphrasing 中的句子对（2）entailment 中的假设-前提对（3）question answering 中使用的 question-passage 对，以及（4）在文本分类或序列标记中使用退化的 text-$\varnothing$ 对。在输出处，将词符表示输入到输出层中以进行词符级任务如序列标记或问题回答，将 [CLS] 表示输入到输出层中进行分类如蕴含或情感分析。

与预训练相比，微调代价相对较小。从完全相同的预训练模型开始，论文中的所有结果都可以复现，在单个 Cloud TPU 上最多 1 个小时，在 GPU 上最多需要几个小时。我们在第 4 节的相应小节中描述特定任务的细节。更多详细信息，请参见附录 A.5。

4. 实验

在本节中，我们介绍了 11 个 NLP 任务的 BERT 微调结果。

4.1. GLUE

通用语言理解评估（GLUE）基准（Wang 等人，2018a）是各种自然语言理解任务的集合。GLUE 数据集的详细说明包含在附录 B.1 中。

为了对 GLUE 进行微调，我们按照第 3 节中的描述表示输入序列（针对单个句子或句子对），并使用对应于第一个输入词符（[CLS]）的最终的隐藏向量 $C \in \mathbb{R}^H$ 作为聚合表示。微调期间引入的唯一新参数是分类层权重 $W \in \mathbb{R}^{K \times H}$，其中 K 是标签数。我们用 C 和 W 计算标准分类损失，即 log（softmax (CW^T)）。

对于所有 GLUE 任务，我们使用的 batch size 为 32，在数据上进行 3 个 epoch 的微调。对于每个任务，我们在开发集上选择最佳的微调学习率（在 5e-5、4e-5、3e-5 和 2e-5 中）。另外，对于 $\text{BERT}_{\text{LARGE}}$，我们发现微调有时在小型数据集上不稳定，因此我们并在开发集上进行了几次随机重跑，选择最佳模型。对于随机重跑，我们使用相同的预训练检查点，但执行不同的微调数据洗乱和分类器层初始化。

结果显示在表 1 中。$\text{BERT}_{\text{BASE}}$ 和 $\text{BERT}_{\text{LARGE}}$ 在所有任务上的性能均优于所有系统，分别比之前的最先进结果提高了 4.5%和 7.0%。请注意，除了注意力屏蔽之外，$\text{BERT}_{\text{BASE}}$ 和 OpenAI GPT 在模型架构方面几乎相同。对于最大且报告最广的 GLUE 任务 MNLI，BERT 获得 4.6%的绝对准确度改进。在官方 GLUE 排行榜上，$\text{BERT}_{\text{LARGE}}$ 得分为 80.5，而截至撰写之日 OpenAI GPT 得分为 72.8 。

我们发现在所有任务中，尤其是训练数据很少的任务，$\text{BERT}_{\text{LARGE}}$ 明显优于 $\text{BERT}_{\text{BASE}}$。在 5.2

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

表 1. GLUE 测试结果，由评估服务器（https://gluebenchmark.com/leaderboard）评分。每个任务下方的数字表示训练样本的数量。“平均” 列与官方 GLUE 得分略有不同，因为我们排除了有问题的 WNLI 集。BERT 和 OpenAI GPT 是单模型、单任务。QQP 和 MRPC 报告的是 F1 分数，STS-B 报告的是 Spearman 相关性，其他任务报告的是准确率分数。我们不包括使用 BERT 作为其组件之一的条目。

部分中更全面地探讨了模型大小的影响。

4.2. SQuAD v1.1

斯坦福问答数据集（SQuAD v1.1）是 10 万个众包问题/答案对的集合（Rajpurkar 等人，2016）。给定一个问题以及 Wikipedia 中包含答案的段落，任务是预测段落中的答案文本范围。

如图 1 所示，在问题回答任务中，我们将输入的问题和段落表示封装成单个序列，其中问题使用 A 嵌入，而段落使用 B 嵌入。在微调期间我们只引入一个开始向量 $S \in \mathbb{R}^H$ 和一个结束向量 $E \in \mathbb{R}^H$。单词 i 是答案范围的开始的概率计算为 T_i 和 S 之间的点积，后面是一个段落上单词的 softmax：$P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$。答案范围的结束用类似的公式。候选范围位置 i 到位置 j 的分值定义为 $S \cdot T_i + E \cdot T_j$，分值最大的范围作为预测，其中 $j\ i$。训练目标是正确的开始和结束位置的对数似然率的总和。我们以 5e-5 的学习率和 32 的批处理大小微调 3 个 epoch。

表 2 显示顶级排行榜和发表的顶级系统（Seo 等人，2017；Clark 和 Gardner，2018；Peters 等人，2018a；Hu 等人，2018）。SQuAD 排行榜的最高结果没有公开最新系统的描述，11 ，允许使用任何公共数据训练他们的系统。因此，我们首先在 TriviaQA 上进行微调（Joshi 等人，2017），然后在 SQuAD 上进行微调，从而在系统中使用适度的数据增强。

我们表现最好的系统在集成时比排行榜最高的系统高出 +1.5 个 F1 值，在单一系统时高出 +1.3 个 F1 值。实际上，就 F1 分数而言，我们的单一 BERT 模型优于最好的集成系统。如果没有 TriviaQA 的微调数据，我们只会损失 0.1-0.4 个 F1 值，仍然远远胜过所有现有系统。

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

表 2. SQuAD 1.1 结果。BERT ensemble 是使用不同的预训练检查点和微调种子的 7 个系统。

4.3. SQuAD v2.0

SQuAD 2.0 任务通过允许在提供的段落中不存在简短答案的可能性扩展了 SQuAD 1.1 问题定义，从而使问题更加实际。

我们使用一种简单的方法来扩展 SQuAD v1.1 BERT 模型以完成此任务。我们将没有答案的问题视为答案范围的开始和结束都位于 [CLS] 词符。答案范围开始和结束位置的概率空间扩展为包括 [CLS] 词符的位置。对于预测，我们比较非答案的范围 $s_null =$

$S \cdot C + E \cdot C$ 和最高的非空范围 $s_{\hat{i}\,\hat{j}} = \max_{j \geq i} S \cdot T_i + E \cdot T_j$ 得分。我们预测非空答案满足 $s_{\hat{i}\,\hat{j}} > s_{\text{null}} + \tau$，其中阈值 在开发集上选择以最大化 F1 值。我们没有为此模型使用 TriviaQA 数据。我们微调了 2 个 epoch，学习率为 5e-5，批次大小为 48。

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
$\text{BERT}_{\text{LARGE}}$ (Single)	78.7	81.9	80.0	83.1

表 3. SQuAD 2.0 结果。我们排除使用 BERT 作为其组件之一的条目。

与之前的排行榜和已发表的最好工作（Sun 等人，2018；Wang 等人，2018b）的比较结果显示在表 3 中，不包括使用 BERT 作为其组件之一的系统。与之前的最佳系统相比，我们观察到 +5.1 F1 的改进。

4.4. SWAG

Situations With Adversarial Generations（SWAG）数据集包含 113k 个句子对补全样本，用于评估基础常识推理（Zellers 等人，2018）。这个任务是在给定一个句子的情况下，在四个选项中选择最合理的选项来延续这个句子。

在 SWAG 数据集上进行微调时，我们构造了四个输入序列，每个输入序列包含给定句子（句子 A）和可能的延续词（句子 B）的连接。引入的唯一特定于任务的参数是一个向量，它与 [CLS] 词符表示 C 的点积表示每个选项的得分，并使用 softmax 层对其进行归一化。

我们对模型进行了 3 个 epoch 的微调，学习率为 2e-5，批处理大小为 16。结果显示在表 4 中。$\text{BERT}_{\text{LARGE}}$ 比作者的基准 ESIM + ELMo 系统高出 +27.1%，比 OpenAI GPT 高出 8.3%。

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

表 4. SWAG 开发集和测试集准确率。† 在 SWAG 论文的报告中，人类的表现是用 100 个样本测量的。

5. 细分研究

在本节中，我们将对 BERT 的多个方面进行细分实验，以更好地了解它们的相对重要性。其它细分研究可在附录 C 中找到。

5.1. 预训练任务的效果

我们通过使用与 BERTBASE 完全相同的预训练数据、微调方案和超参数来评估两个预训练目标，证明了 BERT 深度双向性的重要性。

No NSP：使用" 屏蔽 LM"（MLM）训练的双向模型，但没有" 下一句预测"（NSP）任务。

LTR & No NSP：一个仅有左侧上下文的模型，它是用标准的从左到右（LTR）LM，而不是 MLM 训练。左约束也应用于微调，因为删除它会引入预训练/微调不匹配，从而降低下游性能。此外，该模型无需 NSP 任务即可进行预训练。这可以直接与 OpenAI GPT 相提并论，但要使用更大的训练数据集，输入表示形式和微调方案。

我们首先研究 NSP 任务带来的影响。在表 5 中，我们显示删除 NSP 会严重损害 QNLI、MNLI 和 SQuAD 1.1 的性能。接下来，我们通过比较 “No NSP” 与 “LTR & No NSP” 来评估训练双向表示的影响。在所有任务上，LTR 模型的性能都比 MLM 模型差，而 MRPC 和 SQuAD 的性能下降很大。

对于 SQuAD，直观上很清楚，因为词符级别的隐藏状态没有右侧上下文，所以 LTR 模型在词符预测时的性能会很差。为了使 LTR 系统得到加强，我们在上面添加了一个随机初始化的 BiLSTM。这确

Tasks	Dev Set				
	MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

表 5. 使用 BERTBASE 架构细分预训练任务。“No NSP”使用没有下一个句子预测任务进行训练。“LTR & No NSP”以从左到右的 LM 训练，没有下一个句子预测，和 OpenAI GPT 一样。在微调期间，“+ BiLSTM”在“LTR + No NSP”模型的顶部添加一个随机初始化的 BiLSTM。

实可以显着改善 SQuAD 上的结果，但结果仍然比预训练的双向模型的结果差很多。BiLSTM 损害了 GLUE 任务的性能。

我们认识到，也有可能像 ELMo 一样训练单独的 LTR 和 RTL 模型并将每个词符表示为两个模型的连接。但是：（a）代价是单个双向模型的两倍；（b）对于 QA 这样的任务，这是不直观的，因为 RTL 模型将无法确定问题的答案；（c）这绝对不像深度双向模型那么强大，因为它可以在每一层使用左右上下文。

5.2. 模型大小的影响

在本节中，我们探索模型大小对微调任务准确性的影响。我们训练了许多具有不同层数，隐藏单元和注意头的 BERT 模型，而其他方面则使用了与之前所述相同的超参数和训练过程。

表 6 中显示了选定的 GLUE 任务的结果。在此表中，我们报告了 5 次随机微调重新启动后的平均 Dev Set 精度。我们可以看到较大的模型使得所有四个数据集的准确性提高，即使对于只有 3，600 个带标签的训练样本且与预训练任务大不相同的 MRPC 也是一样。我们能够在相对于现有文献而言已经相当大的模型的基础上实现如此显着的改进，这也许也令人惊讶。例如，在 Vaswani 等人（2017）（L = 6、H = 1024、A = 16）参数为 100M 的编码器，我们在文献中找到的最大 Transformer 是（L = 64、H = 512、A = 2）具有 235M 个参数（Al-Rfou 等人，2018）。相比之下，BERTBASE 包含 110M 参数，而 BERTLARGE 包含 340M 参数。

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

表 6. BERT 模型大小的分解。#L = 层数；#H = 隐藏大小；#A = 注意头的数量。“LM（ppl）”是保留的训练数据的屏蔽 LM 困惑度。

众所周知，增加模型大小将导致大规模任务如机器翻译和语言建模的不断改进，表 6 所示的 LM 对持有的训练数据的迷惑性就证明了这一点。但是，我们认为，这是第一个有说服力的工作，证明只要模型已经过充分的预训练，将模型缩放到极端的模型大小也可以在非常小的规模的任务上带来很大的改进。Peters 等人（2018b）公开了将预训练的 bi-LM 大小从两层增加到四层在下游任务上的混合结果以及 Melamud 等人（2016）顺便提到了将隐藏大小从 200 增加到 600 有帮助，但进一步增加到 1000 并没有带来进一步的改进。这两个先前的工作都使用基于特征的方法-我们假设当直接在下游任务上微调模型并且仅使用很少数量的随机初始化的附加参数时，特定于任务的模型可以从较大的模型中受益，即使下游任务数据非常小，也可以提供更具表现力的预训练表示形式。

5.3. 基于特征的 **BERT** 方法

到目前为止，所有提出的 BERT 结果都使用微调方法，即在预训练模型中添加一个简单的分类层，在下游任务上对所有参数进行共同微调。但是，基于特征的方法，即从预训练的模型中提取固定的特征，也有一定的优势。首先，并非所有任务都可以由 Transformer 编码器结构轻松表示，因此需要添加特定于任务的模型结构。其次，预计算一次训练数据的昂贵表示，然后在这个表示之上用更便宜的模型运行许多实验，有很大的计算优势。

在本节中，我们通过将 BERT 应用于 CoNLL-2003 命名实体识别（NER）任务（Tjong Kim Sang 和 De Meulder，2003）来比较这两种方法。在 BERT 的输入中，我们使用一个大小写保留的 WordPiece 模型，并且最大程度包含数据提供的文档上下文。按照标准惯例，我们将其公式化为标记任务，但在输出中不使用 CRF 层。我们使用第一个子词符的表示作为 NER 标签集上词符级分类器的输入。

为了详细分解研究微调方法，我们应用基于特征的方法，在没有微调 BERT 任何参数的情况下，提取一个或多个层的激活。这些上下文嵌入用作分类层之前的随机初始化的两层 768 维 BiLSTM 的输入。

结果显示在表 7 中。BERT LARGE 与最先进的方法相比具有竞争力。表现最好的方法将来自预训练的 Transformer 的顶部四个隐藏层的词符表示连接起来，这仅比微调整个模型低 0.3 F1。这表明 BERT 对于微调和基于特征的方法均有效。

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

表 7. CoNLL-2003 命名实体识别结果。超参数的选择使用开发集。报告的开发集和测试集得分是用这些超参数在 5 次随机重启后的平均值。

6. 结论

最近由于语言模型的迁移学习经验所带来的改进表明，丰富的、无监督的预训练是许多语言理解系统中不可或缺的一部分。尤其是，这些结果使即使是资源很少的任务也可以从深度单向结构中受益。我们的主要贡献是将这些发现进一步推广到更深的双向体系结构中，从而使相同的预训练模型能够成功解决各种 NLP 任务。