BERT:深度双向transformer的预训练语言理解

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

我们介绍了一种新的语言表示模型**BERT**，它代表双向的编码器表示来自**transformer**。与最近的语言表示模型(Peters et al., 2018a; Radford et al., 2018)不同，BERT旨在通过联合约束所有层的左右上下文，从未标记文本中预训练深度双向表示。因此，预训练的BERT模型可以通过一个额外的输出层进行微调，以创建适用于广泛任务的最先进模型，如问答和语言推理，而无需对特定任务的架构进行大量修改。

BERT是概念上简单，经验上强大的。它在11个自然语言处理任务上获得了最新的结果，包括将胶水分数提高到80.5%(7.7%的绝对提高)，多项精度提高到86.7%(4.6%的绝对提高)，SQuAD v1.1问答测试F1提高到93.2(1.5分的绝对提高)，SQuAD v2.0测试F1提高到83.1(5.1分的绝对提高)。

1 简介

语言模型预训练已被证明对改善许多自然语言处理任务是有效的(Dai and Le, 2015; Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018)。这些任务包括句子级任务，如自然语言推理(Bowman et al., 2015; Williams et al., 2018)和复述(Dolan and Brockett, 2005)，旨在通过整体分析句子之间的关系来预测句子之间的关系，以及标记级任务，如命名实体识别和问答，其中模型需要在标记级生成细粒度输出(Tjong Kim Sang and De Meulder, 2003; Rajpurkar et al., 2016)。

现有两种将预训练语言表示应用于下游任务的策略:基于特征和微调。基于特征的方法，如ELMo (Peters et al., 2018a)，使用特定于任务的架构，其中包括预训练表示作为额外的特征。微调方法，如生成式预训练Transformer (OpenAI GPT) (Radford et al., 2018)，引入了最小的特定于任务的参数，并通过简单微调所有预训练参数在下游任务上进行训练。这两种方法在预训练期间具有相同的目标函数，它们使用单向语言模型来学习通用语言表示。

本文认为，目前的技术限制了预训练表示的能力，特别是微调方法。主要的限制是标准语言模型是单向的，这限制了预训练期间可以使用的架构的选择。例如，在OpenAI GPT中，作者使用从左到右的架构，其中每个标记只能处理Transformer的self-attention层中的前一个标记(Vaswani et al., 2017)。这种限制对于句子级任务是次优的，并且在将基于微调的方法应用于问答等token级任务时可能非常有害，在问答中，从两个方向结合上下文至关重要。

本文通过提出BERT: transformer的双向编码器表示来改进基于微调的方法。BERT受完形填空任务(Taylor, 1953)的启发，通过使用"掩码语言模型"(MLM)预训练目标，缓解了前面提到的单向性约束。掩码语言模型从输入中随机掩码一些标记，目标是仅根据其上下文预测掩码单词的原始词汇id。与从左到右的语言模型预训练不同，MLM目标使表示能够融合左右上下文，这使我们能够预训练深度双向Transformer。除了掩码语言模型之外，还使用了"下一个句子预测"任务，联合预训练文本对表示。本文的贡献如下:

- 本文证明了双向预训练对语言表示的重要性。与Radford et al. (2018)使用单向语言模型进行预训练不同，BERT使用掩码语言模型来实现预训练的深度双向表示。这也与Peters et al. (2018a)形成对比，使用独立训练的从左到右和从右到左的语言模型的浅连接。
- 预训练表示减少了对许多重度工程特定任务架构的需求。BERT是第一个基于微调的表示模型，在大量句子级和标记级任务上实现了最先进的性能，超过了许多特定于任务的体系结构。
- BERT提升了11个NLP任务的技术水平。代码和预训练模型可以在https://github.

com/google-research/bert上找到。

2 相关工作

预训练通用语言表示有很长的历史，本节简要回顾一下最常用的方法。

2.1 基于特征的无监督方法

学习广泛适用的单词表示是几十年来的一个活跃研究领域，包括非神经方法(Brown et al., 1992; Ando and Zhang, 2005; Blitzer et al., 2006)和神经方法(Mikolov et al., 2013; Pennington et al., 2014)。预训练的词嵌入是现代NLP系统的组成部分，比从头学习的嵌入提供了显著的改进(Turian et al., 2010)。在预训练词嵌入向量时，使用了从左到右的语言建模目标(Mnih and 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 et al., 2013)。

这些方法已被推广到更粗的粒度，如句子嵌入(Kiros et al., 2015; Logeswaran and Lee, 2018)或段落嵌入(Le and Mikolov, 2014)。为了训练句子表示，之前的工作使用目标对候选下一个句子进行排序(Jernite et al., 2017; Logeswaran and Lee, 2018)，从左到右生成下一个句子单词，给出上一个句子的表示(Kiros et al., 2015)，或去噪自编码器派生的目标(Hill et al., 2016)。

ELMo及其前身(Peters et al., 2017, 2018a)从一个不同的维度对传统的词嵌入研究进行了概括。它们从从左到右和从右到左的语言模型中提取上下文相关的特征。每个标记的上下文表示形式是由从左到右和从右到左的表示形式拼接而成。当将上下文词嵌入与现有的特定任务架构集成时，ELMo推进了几个主要NLP基准的最先进水平(Peters et al., 2018a)，包括问答(Rajpurkar et al., 2016)，情感分析(Socher et al., 2013)和命名实体识别(Tjong Kim Sang and De Meulder, 2003)。Melamud et al. (2016)提出通过从左右上下文预测单个单词的任务来学习上下文表示使用lstm。与ELMo类似，他们的模型是基于特征的，不是深度双向的。Fedus et al. (2018)表明完形填空任务可以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，第一个方法只从未标记的文本中预训练词嵌入参数(Collobert and Weston, 2008)。

最近，生成上下文标记表示的句子或文档编码器已经从未标记的文本中进行了预训练，并为有监督的下游任务进行了微调(Dai and Le, 2015; Howard and Ruder, 2018; Radford et al., 2018)。这些方法的优点是很少有参数需要从头学习。至少部分由于这一优势，OpenAI GPT (Radford et al., 2018)在GLUE基准(Wang et al., 2018a)中的许多句子级别任务上取得了以前最先进的结果。从左到右的语言建模和自动编码器目标已用于预训练此类模型(Howard and Ruder, 2018; Radford et al., 2018; Dai and Le, 2015)。

2.3 监督数据的迁移学习

也有工作表明，从具有大型数据集的监督任务中进行有效的迁移，如自然语言推理(Conneau et al., 2017)和机器翻译(McCann et al., 2017)。计算机视觉研究还证明了从大型预训练模型中进行迁移学习的重要性，其中一个有效的方法是对使用ImageNet (Deng et al., 2009; Yosinski et al., 2014)预训练的模型进行微调。

3 Bert

介绍BERT及其具体实现。该框架有两个步骤:预训练和微调。在预训练期间，模型在不同的预训练任务上的未标记数据上进行训练。对于微调，首先用预训练参数初始化BERT模型，并使用来自下游任务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，即使它们用相同的预训练参数初始化。图1中的问答示例将作为本节的运行示例。

BERT的一个独特特性是它跨不同任务的统一架构。预训练架构和最终的下游架构之间的差异很小。

模型架构 BERT的模型架构是一个多层双向Transformer编码器，基于Vaswani et al. (2017)中描述的原始实现，并在2tensor库中发布。[1]因为Transformer的使用已经变得很普遍，我们的实现几乎与原始版本相同，所以我们将省略对模型架构的详尽背景描述，并请读者参考Vaswani et al. (2017)以及优秀的指南，如"带注释的Transformer"。[2]

本文将层数(即Transformer块)表示为L，隐藏大小表示为H，自注意力头数量表示为A。[3]我们主要报告两种模型大小的结果:**BERT$_{\text{BASE}}$** (L=12, H=768, A=12, 总参数=110M)和**BERT$_{\text{LARGE}}$** (L=24, H=1024, A=16，总参数=340M)。

BERT$_{\text{BASE}}$为了便于比较，选择了与OpenAI GPT相同的模型大小。然而，关键的

[1] https://github.com/tensorflow/tensor2tensor

[2] http://nlp.seas.harvard.edu/2018/04/03/attention.html

[3] 在所有情况下，我们将前馈/滤波器大小设置为$4H$，即$H = 768$为3072,$H = 1024$为4096。

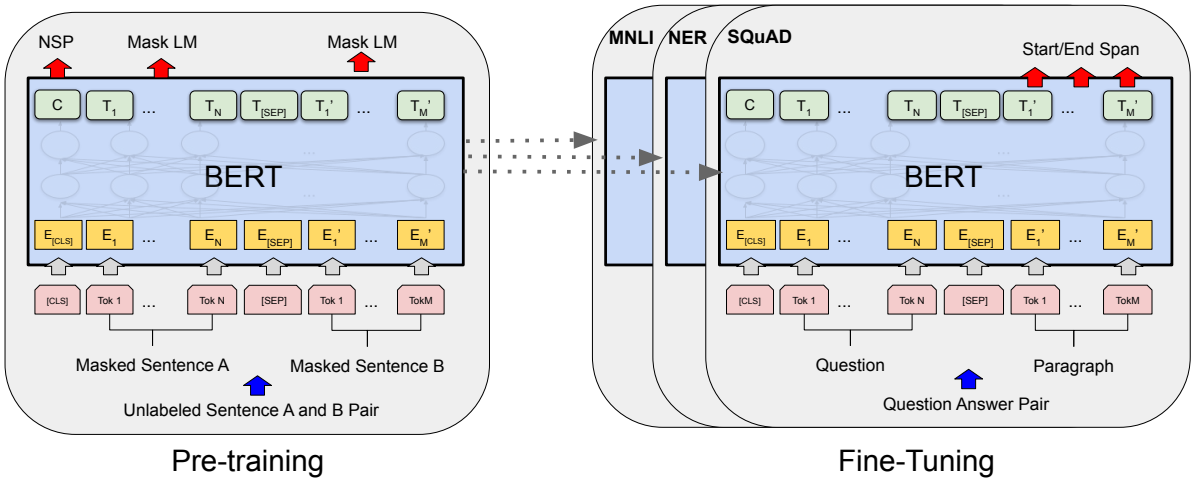


Figure 1: BERT的整体预训练和微调过程。除了输出层，预训练和微调中都使用了相同的架构。相同的预训练模型参数用于为不同的下游任务初始化模型。在微调过程中，对所有参数进行微调。[CLS]是添加的特殊符号在每个输入示例的前面，[SEP]是一个特殊的分隔符(例如，分隔问题/答案)。

是，BERT Transformer使用双向自注意力，而GPT Transformer使用约束自注意力，每个标记只能关注其左侧的上下文。[4]

输入/输出表示法　为了让BERT处理各种下游任务，输入表示能够在一个标记序列中无歧义地表示单个句子和一对句子(例如，⟨问题，答案⟩)。在这项工作中，一个"句子"可以是连续文本的任意跨度，而不是实际的语言句子。"序列"是指BERT的输入token序列，它可能是一个句子或两个句子组合在一起。

我们使用带有30000 token词汇的WordPiece embedding (Wu et al., 2016)。第一个标记总是一个特殊的分类标记([CLS])。与此标记相对应的最终隐藏状态被用作分类任务的聚合序列表示。句子对被打包成一个单独的序列。我们用两种方法来区分这些句子。首先，我们用一个特殊的标记([SEP])将它们分开。其次，我们为每个词添加一个学习到的嵌入，以表示它属于句子还是句子。如图1所示，我们表示输入嵌入为E，特殊[CLS]标记的最终隐藏向量为$C \in \mathbb{R}^H$，i^{th}输入标记的最终隐藏向量为$T_i \in \mathbb{R}^H$。

对于给定的标记，其输入表示是通过对相应的标记、段和位置嵌入求和来构建的。这种结构的可视化可以在图2中看到。

3.1　预训练BERT

与Peters et al. (2018a)和Radford et al. (2018)不同，我们没有使用传统的从左到右或从右到左的语言模型来预训练BERT。相反，我们使用两个无监督任务对BERT进行预训练，如本节所述。这一步显示在图1的左侧。

任务#**1:**蒙面**LM**　直觉上，我们有理由相信，深度双向模型严格来说比从左到右模型或从左到右和从右到左模型的浅层连接更强大。不幸的是，标准的条件语言模型只能被训练为从左到右或从右到左，因为双向条件可以允许每个词间接地"看到自己"，并且该模型可以简单地在多层上下文中预测目标词。

为了训练深度的双向表示，我们简单地随机掩码一定比例的输入标记，然后预测这些掩码标记。我们将这个过程称为"掩码语言模型"(MLM)，尽管它在文献(Taylor, 1953)中经常被称为任务。在这种情况下，与掩码标记相对应的最终隐藏向量被输入到词汇表上的输出softmax中，就像在标准LM中一样。在所有实验中，随机屏蔽了每个序列中15

尽管这允许我们获得双向预训练模型，但缺点是我们在预训练和微调之间创建了不匹配，因为在微调期间没有出现[MASK]令牌。为了缓解这个问题，我们并不总是用实际的[MASK]标记替换"掩码"单词。训练数据生成器随机选择15%的token位置进行预测。如果选择i -th令牌，我们将i -th令牌替换为(1)[MASK]令牌(80%的时间)(2)随机令牌(10%的时间)(3)未改变的i -th令牌(10%的时间)。然后，T_i将用于预测具有交叉熵损失的原始token。我们在附录C.2中比较了这个过程的变体。

任务**2:**下一个句子预测**(NSP)**　许多重要的下游任务，如问答(Question Answering, QA)和自

[4]我们注意到，在文献中，双向Transformer通常被称为"Transformer编码器"，而仅左上下文的版本被称为"Transformer解码器"，因为它可以用于文本生成。

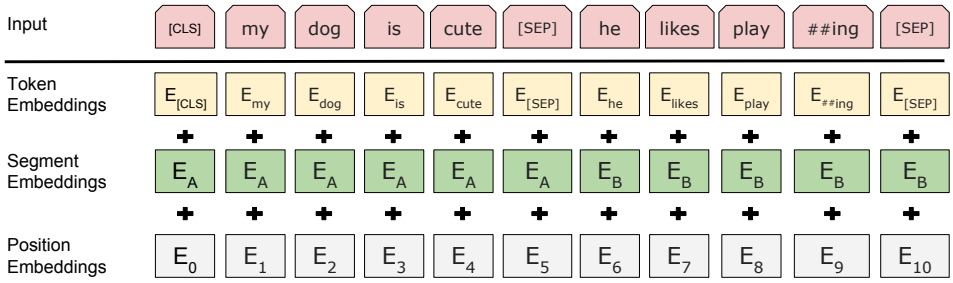


Figure 2: BERT输入表示。输入嵌入是标记嵌入、分割嵌入和位置嵌入的总和。

然语言推理(Natural Language Inference, NLI)，都是基于理解两个句子之间的关系，而语言建模无法直接捕捉到这些关系。为了训练一个理解句子关系的模型，我们对一个二值化的下一个句子预测任务进行了预训练，该任务可以从任何单语语料库中轻松生成。具体来说，当为每个预训练示例选择句子和时，50%的时间是A之后的实际下一个句子(标记为如)，50%的时间它是语料库中的随机句子(标记为NotNext)。如图1所示，C用于下一个句子预测(NSP)。[5]尽管很简单，但我们在5.1节中证明，针对这项任务的预训练对QA和NLI都非常有益。[6] NSP任务与Jernite et al. (2017)和Logeswaran and Lee (2018)中使用的表示学习目标密切相关。然而，在之前的工作中，只有句子嵌入被迁移到下游任务中，其中BERT迁移所有参数来初始化最终任务模型参数。

预训练数据 预训练过程在很大程度上遵循了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(8亿单词)(Zhu et al., 2015)和英语维基百科(2500亿单词)。对于维基百科，我们只提取文本段落，忽略列表、表格和标题。为了提取长连续序列，使用文档级语料库而不是混淆的句子级语料库(如十亿单词基准(Chelba et al., 2013))至关重要。

3.2 微调BERT

微调是直接的，因为Transformer中的自注意力机制允许BERT通过交换适当的输入和输出来对许多下游任务进行建模——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程序，一个常见的模式是在应用双向交叉注意力之前独立编码文本对，例如Parikh et al. (2016); Seo et al. (2017)。相反，BERT使用自注意力机制将这两个阶段统一为编码具有自注意力的连接文本对有效地包含两个句子之间的双向交叉注意力。

对于每个任务，我们只需将特定于任务的输入和输出插入BERT中，并对所有参数进行端到端的微调。在输入时，来自预训练的句子和句子类似于(1)复述中的句子对，(2)蕴含中的假设-前提对，(3)问答中的问题-段落对，以及(4)文本分类或序列标注中的退化文本- ∅对。在输出中，token表示被输入到输出层以完成token级任务，如序列标记或问题回答，而[CLS]表示被输入到输出层以进行分类，如蕴含或情感分析。

与预训练相比，微调相对便宜。从完全相同的预训练模型开始，论文中的所有结果可以在单个云TPU上最多1小时内复制，或在GPU上最多几个小时。[7] 我们将在4节的相应小节中描述特定于任务的细节。更多细节请参见附录A.5。

4 实验

在本节中，我们将展示BERT在11个NLP任务上的微调结果。

4.1 胶水

通用语言理解评估(GLUE)基准(Wang et al., 2018a)是各种自然语言理解任务的集合。GLUE数据集的详细描述见附录B.1。

为了对GLUE进行微调，我们表示输入序列(对于单个句子或句子对)，如3节所述，并使用对应于第一个输入标记的最终隐藏向量$C \in \mathbb{R}^H$作为聚合表示。微调过程中引入的唯一新参数是分类层权重$W \in \mathbb{R}^{K \times H}$，其中$K$是标签的数量。我们用$C$和$W$计算标准分类损失，即$\log(\text{softmax}(CW^T))$。

我们使用32的批量大小，并对所有GLUE任务的数据进行3个epoch的微调。对于每个任务，我们在开发集上选择了最佳的微调学习率(在5e-5、4e-5、3e-5和2e-5中)。此外，对于$\text{BERT}_{\text{LARGE}}$，我们发现微调在小型数据集

[5]最终模型在NSP上取得了97%～98%的准确率。

[6]向量C在没有微调的情况下不是有意义的句子表示，因为它是用NSP训练的。

[7]例如，BERT SQuAD模型可以在单个云TPU上在大约30分钟内进行训练，以达到91.0%的Dev F1分数。

[8]见https://gluebenchmark.com/faq(10)。

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**

Table 1: 胶水测试结果，由评估服务器(https://gluebenchmark.com/leaderboard)评分。每个任务下面的数字表示训练示例的数量。'Average '列与官方的GLUE分数略有不同，因为我们排除了有问题的WNLI集。[8] BERT和OpenAI GPT都是单模型、单任务。QQP和MRPC报告了F1分数，STS-B报告了Spearman相关性，准确性分数报告了其他任务。我们排除了使用BERT作为组件之一的条目。

上有时是不稳定的，因此我们运行了几次随机重启，并在开发集上选择了最佳模型。通过随机重启，我们使用相同的预训练检查点，但执行不同的微调数据洗牌和分类器层初始化。[9]

结果如表1所示。$\text{BERT}_{\text{BASE}}$和$\text{BERT}_{\text{LARGE}}$在所有任务上的表现都大大超过了所有系统，与之前的技术水平相比分别获得了4.5%和7.0%的平均精度提高。请注意，除了注意力掩盖之外，$\text{BERT}_{\text{BASE}}$和OpenAI GPT在模型架构方面几乎相同。对于最大和报道最广泛的GLUE任务MNLI, BERT获得了4.6%的绝对精度提升。在官方的GLUE排行榜[10]上，$\text{BERT}_{\text{LARGE}}$的得分为80.5，而OpenAI GPT的得分为72.8。

$\text{BERT}_{\text{LARGE}}$在所有任务中明显优于$\text{BERT}_{\text{BASE}}$，特别是在训练数据很少的任务中。模型大小的影响在5.2节中进行了更深入的探讨。

4.2 SQuAD v1.1

斯坦福问答数据集(SQuAD v1.1)是一个10万组众包问题/答案对的集合(Rajpurkar et al., 2016)。给定一个问题和一篇来自维基百科的包含答案的文章，任务是预测文章中答案文本的跨度。

如图1所示，在问答任务中，我们将输入问题和文章表示为单个打包序列，问题使用嵌入，文章使用嵌入。在微调过程中，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作为答案范围开始的概率计算为T_i和S之间的点积，然后是段落中所有单词的softmax: $P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$。类似的公式用于答案范围的末尾。候选人从位置$i$到位置$j$的得分被定义为$S \cdot T_i + E \cdot T_j$，最大得分范围$j \geq i$被用作预测。训练目标是正确开始位置和结束位置的对数似然的和。对3个epoch进行微调，学习率为5e-5，批处理大小为32。

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**

Table 2: 阵容1.1结果。BERT集成是7x系统，使用不同的预训练检查点和微调种子。

表2显示了顶级排行榜条目以及来自顶级发布系统(Seo et al., 2017; Clark and Gardner, 2018; Peters et al., 2018a; Hu et al., 2018)的结果。排名第一的结果没有最新的公共系统描述，[11]，并且允许在训练他们的系统时使用任何公共数据。因此，我们在系统中使用适度的数据增强，首先在TriviaQA (Joshi et al., 2017)上进行微调，然后再对SQuAD进行微调。

我们表现最好的系统在集成上比顶级排行榜系统高出1.5 F1，在单个系统上高出1.3 F1。事实上，我们的单个BERT模型在F1分数方面优于顶级集成系统。在没有TriviaQA微调数据的情况下，我们只损失了0.1-0.4 F1，仍然大大超过了所有现有系统。[12]

[9] GLUE数据集分布不包括测试标签，我们仅为$\text{BERT}_{\text{BASE}}$和$\text{BERT}_{\text{LARGE}}$分别提交了单个GLUE评估服务器。

[10] https://gluebenchmark.com/leaderboard

[11] QANet的描述在Yu et al. (2018)，但该系统在发布后有了很大的改进。

[12] 我们使用的TriviaQA数据由TriviaQA- wiki文档中的前400个token组成的段落组成，这些段落至少包含提供的一个可能的答案。

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

Table 3: 救援队2.0的结果。我们排除了使用BERT作为组件之一的条目。

4.3 SQuAD v2.0

SQuAD 2.0任务扩展了SQuAD 1.1问题的定义，考虑到所提供的段落中可能不存在简短的答案，使问题更加现实。

为此，我们使用一种简单的方法来扩展SQuAD v1.1 BERT模型。我们将没有答案的问题视为具有起始和结束于[CLS]标记的答案范围。开始和结束答案跨度位置的概率空间被扩展到包括[CLS]标记的位置。为了进行预测，我们将无答案跨度的分数:$s_{\text{null}} = S{\cdot}C + E{\cdot}C$与最佳非空跨度的分数$\hat{s_{i,j}} = \max_{j \geq i} S{\cdot}T_i + E{\cdot}T_j$进行比较。当$\hat{s_{i,j}} > s_{\text{null}} + \tau$，其中在dev集上选择阈值$\tau$以最大化F1时，我们预测一个非空答案。我们没有在这个模型中使用TriviaQA数据。我们对两个epoch进行了微调，学习率为5e-5，批量大小为48。

表3中显示了与之前的排行榜条目和顶级发布工作(Sun et al., 2018; Wang et al., 2018b)的比较结果，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，F1值提高了5.1。

4.4 饰物

对抗性生成(SWAG)数据集包含113k句对补全示例，评估基础常识推理(Zellers et al., 2018)。给定一个句子，任务是在四个选项中选择最合理的延续。

当对SWAG数据集进行微调时，我们构建了四个输入序列，每个序列都包含给定句子(句子)和可能的延续(句子)的连接。引入的唯一特定于任务的参数是一个向量，其与[CLS] token表示的点积C表示每个选择的分数，该分数用softmax层进行归一化。

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。结果如表4所示。$\text{BERT}_{\text{LARGE}}$的性能比作者的基线ESIM+ELMo系统提高了27.1%，比OpenAI GPT系统提高了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

Table 4: SWAG开发和测试准确性。†正如SWAG论文中报道的那样，我们用100个样本来测量人类的表现。

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

Table 5: 消融使用$\text{BERT}_{\text{BASE}}$架构的预训练任务。"No NSP"在没有下一个句子预测任务的情况下进行训练。'LTR &No NSP '被训练为一个没有下一个句子预测的从左到右的LM，就像OpenAI GPT一样。"+ BiLSTM"在微调过程中，在"LTR + No NSP '模型的基础上添加一个随机初始化的BiLSTM。

5 消融研究

在本节中，我们对BERT的多个方面进行消融实验，以更好地理解它们的相对重要性。其他消融研究见附录C。

5.1 预训练任务的效果

本文通过使用与$\text{BERT}_{\text{BASE}}$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:

无**NSP**:使用"掩码LM"(MLM)但没有"下一个句子预测"(NSP)任务训练的双向模型。

LTR &No NSP:仅左上下文模型，使用标准的从左到右(LTR) LM而不是MLM进行训练。仅左约束也适用于微调，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，该模型是在没有NSP任务的情况下进行预训练的。这与OpenAI GPT直接相当，但使用的是更大的训练数据集、输入表示和微调方案。

首先考察了NSP任务带来的影响。在表5中，我们显示删除NSP会显著影响QNLI、MNLI和SQuAD 1.1的性能。通过

将"无NSP "与" LTR & 无NSP "进行比较，评估了训练双向表示的影响。LTR模型在所有任务上的表现都比MLM模型差，在MRPC和SQuAD上有很大的下降。

对于SQuAD来说，很明显，LTR模型在token预测方面表现不佳，因为token级隐藏状态没有右侧上下文。为了加强LTR系统，我们在上面添加了一个随机初始化的BiLSTM。这确实显著提高了SQuAD的结果，但结果仍然比预训练的双向模型差得多。BiLSTM会影响GLUE任务的性能。

我们认识到，也可以训练单独的LTR和RTL模型，并将每个token表示为两个模型的连接，就像ELMo所做的那样。然而:(a)这是单一双向模型的两倍;(b)对于QA这样的任务来说，这是不直观的，因为RTL模型无法根据问题来设置答案;(c)严格来说，它没有深度双向模型强大，因为它在每一层都可以使用左右上下文。

5.2 模型尺寸效应

在本节中，我们将探讨模型大小对微调任务精度的影响。我们使用不同数量的层数、隐藏单元和注意力头训练了一些BERT模型，而使用相同的超参数和训练过程(如前所述)。

所选胶水任务的结果如表6所示。在这个表中，我们报告了微调的5次随机重启的平均开发集精度。我们可以看到，更大的模型在所有四个数据集上都带来了严格的精度提高，即使是只有3600个带标签的训练样本的MRPC，也与预训练任务有本质上的不同。也可能令人惊讶的是，我们能够在相对于现有文献已经相当大的模型之上实现如此显著的改进。例如，在Vaswani et al. (2017)中探索的最大的Transformer是(L=6, H=1024, A=16)，编码器参数为100M，而我们在文献中发现的最大的Transformer是(L=64, H=512, A=2)，参数为235M (Al-Rfou et al., 2018)。其中$\text{BERT}_{\text{BASE}}$包含110M个参数，$\text{BERT}_{\text{LARGE}}$包含340M个参数。

众所周知，增加模型大小将导致机器翻译和语言建模等大规模任务的持续改进，表6所示的保留训练数据的LM困惑证明了这一点。然而，我们相信，这是第一项工作，令人信服地证明，如果模型已经充分预训练，扩展到极端的模型尺寸也会导致在非常小的规模任务上有很大的改进。Peters et al. (2018b)提出了将预训练的bi-LM大小从2层增加到4层对下游任务影响的混合结果，Melamud et al. (2016)在文中提到，将隐藏维度大小从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。之前的这两项工作都使用了一种基于特征的方法——假设当模型直接在下游任务上进行微调，并仅使用少量随机初始化的额外参数时，特定于任务的模型可以从更大、更有表现力的预训练表示中受益，即使下游任务数据非常少。

5.3 BERT基于特征的方法

到目前为止，所有BERT结果都使用了微调方法，即在预训练模型中添加一个简单的分类层，并在下游任务上联合微调所有参数。然而，基于特征的方法从预训练模型中提取固定特征，具有一定的优势。首先，不是全部任务可以很容易地由Transformer编码器架构表示，因此需要添加特定于任务的模型架构。其次，这样做在计算上有很大的好处预先计算一次昂贵的训练数据表示，然后运行多次实验更便宜在这种表示之上的模型。

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 and De Meulder, 2003)来比较这两种方法。在BERT的输入中，我们使用保留大小写的WordPiece模型，并包含数据提供的最大文档上下文。按照标准实践，我们将其表述为标记任务，但在输出中不使用CRF层。我们使用第一个子标记的表示作为NER标签集上标记级分类器的输入。

为了削弱微调方法，我们应用基于特征的方法，从一个或多个层中提取激活值，而无需微调BERT的任何参数。这些上下文嵌入被用作分类层之前随机初始化的两层768维BiLSTM的输入。

结果如表7所示。$\text{BERT}_{\text{LARGE}}$与最先进的方法进行竞争。表现最好的方法连接了预训练Transformer的前四个隐藏层的token表示，与微调整个模型相比，这仅需要0.3 F1。这表明BERT对于微调和基于特征的方法都是有效的。

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
	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

Table 6: 消融BERT模型大小。#L =图层数;#H =隐藏大小;#A =注意力头的数量" LM (ppl)"是保留的训练数据的掩码LM困惑。

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	-

Table 7: CoNLL-2003命名实体识别结果。使用Dev集选择超参数。报告的开发和测试分数是使用这些超参数在5次随机重启后的平均分数。

6 结论

由于语言模型的迁移学习，最近的实证改进表明，丰富的、无监督的预训练是许多语言理解系统的组成部分。特别是，这些结果使即使是低资源任务也能从深度单向架构中受益。本文的主要贡献是将这些发现进一步推广到深度双向架构，使相同的预训练模型能够成功地解决广泛的NLP任务。

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2018. Character-level language modeling with deeper self-attention. *arXiv preprint arXiv:1808.04444*.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The fifth PASCAL recognizing textual entailment challenge. In *TAC*. NIST.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*. Association for Computational Linguistics.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Z. Chen, H. Zhang, X. Zhang, and L. Zhao. 2018. Quora question pairs.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *ACL*.

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*. Association for Computational Linguistics.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *IJCAI*.

Yacine Jernite, Samuel R. Bowman, and David Sontag. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. *CoRR*, abs/1705.00557.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The winograd schema challenge. In *Aaai spring symposium: Logical formalizations of commonsense reasoning*, volume 46, page 47.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *International Conference on Learning Representations*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *CoNLL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1081–1088. Curran Associates, Inc.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *NAACL*.

Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, OpenAI.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Fu Sun, Linyang Li, Xipeng Qiu, and Yang Liu. 2018. U-net: Machine reading comprehension with unanswerable questions. *arXiv preprint arXiv:1810.06638*.

Wilson L Taylor. 1953. "Cloze procedure": A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018a. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Wei Wang, Ming Yan, and Chen Wu. 2018b. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Appendix for "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

我们将附录分为三个部分。

- BERT的其他实现细节见附录A;
- 实验的更多细节见附录B;而且
- 其他消融研究见附录C。

  本文提出其他针对BERT的消融研究，包括:

  - 训练步数的影响;而且
  - 消融不同的掩蔽程序。

A BERT的更多细节

A.1 预训练任务的说明

我们在下面提供预训练任务的示例。

掩码**LM**和掩码程序 假设未标记的句子是dog is hairy，在随机掩码过程中，我们选择了第4个token(对应于hairy)，我们的掩码过程可以进一步说明吗

- 80%的情况:将单词替换为[MASK] token，例如，dog is hairy →我的狗是[MASK]
- 10%的概率:用一个随机的单词替换这个单词，例如，dog is hairy →我的狗是apple
- 10%的概率:保持单词不变，例如:dog is hairy →我的狗是毛茸茸的。这样做的目的是使表示偏向于实际观察到的单词。

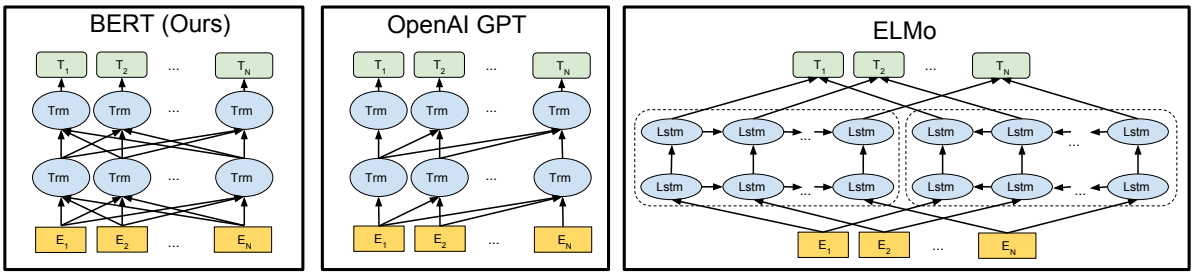


Figure 3: 预训练模型架构的差异。BERT使用双向Transformer。OpenAI GPT使用从左到右的Transformer。ELMo使用连接独立训练的从左到右和从右到左的lstm来为下游任务生成特征。在这三种表示中，只有BERT表示在所有层中同时以左上下文和右上下文为条件。除了架构上的差异，BERT和OpenAI GPT是微调方法，而ELMo是基于特征的方法。

这个过程的优点是，Transformer编码器不知道将被要求预测哪些单词，或者哪些单词已被随机单词替换，因此它被迫保持每个输入标记的分布上下文表示。此外，由于随机替换只发生在所有标记的1.5%(即15%的10%)，这似乎不会损害模型的语言理解能力。在C.2部分，我们评估这个程序的影响。

与标准的语言模型训练相比，掩码的LM只有对每个批次中15%的token进行预测，这表明模型可能需要更多的预训练步骤才能收敛。在C.1节中，我们证明了MLM的收敛速度确实比从左到右的模型(预测每个token)稍慢，但MLM模型的经验改进远远超过了增加的训练成本。

下一句预测　下一个句子预测任务可以在以下示例中说明。

Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Label = `IsNext`

Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Label = `NotNext`

A.2 预训练过程

为了生成每个训练输入序列，我们从语料库中采样两段文本，我们称之为“句子”，尽管它们通常比单个句子长得多(但也可以短得多)。第一个句子接收到嵌入，第二个接收到嵌入。50%的情况下是A之后的实际下一个句子，50%的情况下它是随机句子，这是为“下一个句子预测”任务完成的。对它们进行采样，使组合长度为≤ 512 token。WordPiece分词后采用LM掩模，掩模率为15%，对部分词块不作特别考虑。

用256个序列(256个序列* 512个token = 128,000个token /batch)的批大小进行训练，共100 000 000步，这在33亿单词语料库上大约是40个epoch。我们使用Adam，学习率为1e-4，$\beta_1 = 0.9$, $\beta_2 = 0.999$, L2权重衰减为0.01，学习率在前10000步中预热，学习率线性衰减。我们在所有层上使用0.1的退出概率。我们使用激活(Hendrycks and Gimpel, 2016)而不是标准的，遵循OpenAI GPT。训练损失是平均掩码LM似然和平均下一句话预测似然的总和。

在Pod配置下对4个云TPU(共16个TPU芯片)进行$\text{BERT}_{\text{BASE}}$培训。[13]$\text{BERT}_{\text{LARGE}}$在16块云TPU(共64块TPU芯片)上进行训练。每次预训练需要4天才能完成。

较长的序列的开销更大，因为注意力是序列长度的二次型。为了加快预训练速度，在实验中，对90%的步骤进行了序列长度为128的预训练模型。然后，我们训练剩余的10%的512序列步骤来学习位置嵌入。

A.3 微调过程

对于微调，除了批量大小、学习率和训练周期数外，大多数模型超参数与预训练时相同。退出概率始终保持在0.1。最佳超参数值是特定于任务的，但我们发现以下可能的值范围可以适用于所有任务:

- 批量大小:16,32
- 学习率(**Adam**): 5e-5, 3e-5, 2e-5
- **epoch**的数量:2、3、4

我们还观察到，大数据集(例如，100k+带标签的训练样本)对超参数选择的敏感性远远低于小数据集。微调通常非常快，因此合理的做法是简单地对上述参数进行穷举搜索，并选择在开发集上表现最好的模型。

[13]https://cloudplatform.googleblog.com/2018/06/Cloud-TPU-now-offers-preemptible-pricing-and-global-availability.html

A.4 BERT、ELMo和OpenAI GPT的比较

本文研究了最近流行的表示学习的差异模型包括ELMo, OpenAI GPT和BERT。模型之间的比较架构如图3所示。注意除了架构上的差异，BERT和OpenAI GPT是微调方法，而ELMo是基于特征的方法。

与BERT最相似的现有预训练方法是OpenAI GPT，它在大型文本语料库上训练一个从左到右的Transformer LM。事实上，BERT中的许多设计决策都是为了使其尽可能接近GPT，这样两种方法的比较就可以最小化。这项工作的核心论点是，3.1节中提出的双向性和两个预训练任务占了大多数实证改进，但我们注意到BERT和GPT的训练方式之间还有其他几个差异:

- GPT是在BooksCorpus(8亿字)上训练的;BERT在BooksCorpus(8亿单词)和Wikipedia(2500亿单词)上进行训练。
- GPT使用句子分隔符(`[SEP]`)和分类器标记(`[CLS]`)，它们只在微调时引入;BERT在预训练期间学习`[SEP]`，[CLS]和句子A/ B嵌入。
- GPT训练了1M步，batch大小为3.2万单词;BERT训练了100万步，batch大小为12.8万单词。
- GPT在所有微调实验中使用相同的5e-5学习率;BERT选择一个特定于任务的微调学习率，该学习率在开发集上表现最好。

为隔离这些差异的影响，在5.1节中进行了消融实验，表明大多数改进实际上来自两个预训练任务及其实现的双向性。

A.5 不同任务的微调说明

在不同任务中对BERT进行微调的说明可以参见图4。我们的特定于任务的模型是通过将BERT与一个额外的输出层结合起来形成的，因此需要从头学习最少数量的参数。在这些任务中，(a)和(b)是序列级任务，(c)和(d)是标记级任务。在图中，E表示输入嵌入，T_i表示标记i的上下文表示，[CLS]是用于分类输出的特殊符号，[SEP]是用于分隔非连续标记序列的特殊符号。

B 详细实验装置

B.1 胶水基准实验的详细描述。

我们在表1中的胶水结果来自`https://gluebenchmark.com/leaderboard`和`https://blog.openai.com/language-unsupervised`。GLUE基准包括以下数据集，其描述最初总结在Wang et al. (2018a):

mnli 多流派自然语言推理是一项大规模的众包蕴含分类任务(Williams et al., 2018)。给定一对句子，目标是预测第二个句子相对于第一个句子是蕴含、矛盾还是中立。

QQP Quora问题对是一个二分类任务，目标是确定Quora上提出的两个问题在语义上是否等价(Chen et al., 2018)。

qnli 问题自然语言推理是斯坦福问答数据集(Rajpurkar et al., 2016)的一个版本，该数据集已转换为二分类任务(Wang et al., 2018a)。肯定的例子是(问题，句子)对，它们包含正确的答案，否定的例子是来自同一段的(问题，句子)，但没有包含答案。

sst-2 斯坦福情感树库是一个二值单句分类任务，由从电影评论中提取的句子以及人工标注的情感(Socher et al., 2013)组成。

可乐 语言可接受性语料库是一个二元单句分类任务，其目标是预测一个英语句子在语言上是否“可接受”(Warstadt et al., 2018)。

sts-b 语义文本相似度基准是一个句子对的集合，这些句子对来自新闻标题和其他来源(Cer et al., 2017)。它们被标注了从1到5的分数，表示这两个句子在语义方面的相似程度。

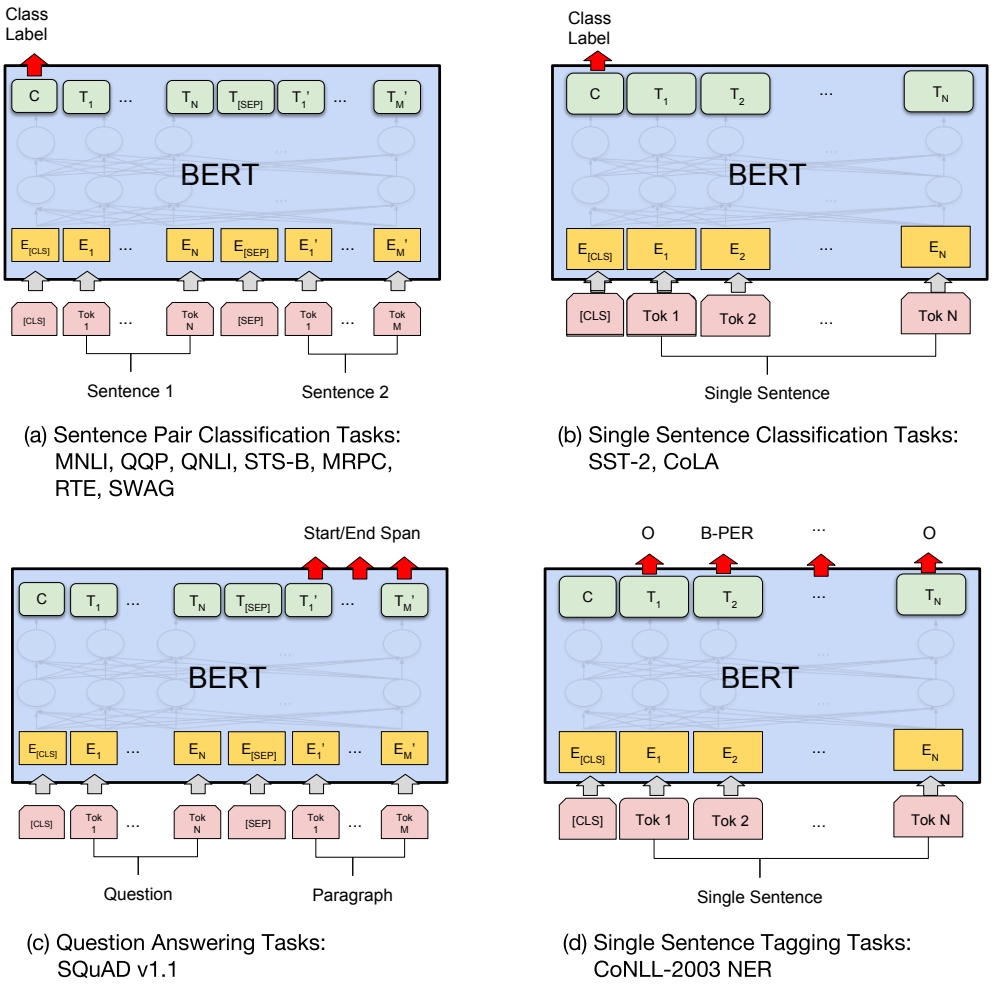
MRPC Microsoft Research的复述语料库由自动从在线新闻源中提取的句子对组成，人工标注了句子对中的句子是否语义等价(Dolan and Brockett, 2005)。

rte 识别文本蕴含是一个类似于MNLI的二进制蕴含任务，但训练数据要少得多(Bentivogli et al., 2009)。[14]

wnli Winograd NLI是一个小型的自然语言推理数据集(Levesque et al., 2011)。GLUE的网页指出，这个数据集的构建存在问题，[15]并且提交给GLUE的每个经过训练的系统的表现都低于预测大多数类别的65.1基线精度。因此，为了对OpenAI GPT公平，我们排除了这个集合。对于我们的GLUE提交，我们总是预测大多数类别。

[14]请注意，本文只报告单任务微调结果。多任务微调方法可能会推动表演更进一步。例如，我们确实观察到使用MNLI进行多任务训练对RTE的显著改善。

[15]`https://gluebenchmark.com/faq`



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 4: 不同任务上对BERT进行微调的示例。

C 额外的消融研究

C.1 训练步数的影响

图5展示了经过k步骤预训练的检查点微调后MNLI Dev的准确性。这允许我们回答以下问题。

1. 问题:BERT真的需要如此大量的预训练(128,000字/批* 1,000,000步)来实现高微调精度吗?
   回答:是的，$\text{BERT}_{\text{BASE}}$在训练100万步时比训练50万步时在MNLI上获得了近1.0%的额外精度。

2. 问:MLM预训练是否比LTR预训练收敛慢，因为在每个批次中只有15%的单词被预测，而不是每个单词?
   答案:MLM模型的收敛速度确实比LTR模型稍慢。然而，就绝对精度而言，MLM模型几乎立即开始优于LTR模型。

C.2 消融不同的掩蔽程序

在3.1节中，我们提到BERT在使用掩码语言进行预训练时使用混合策略来掩码目标标记模型(MLM)目标。下面是一项消融研究，以评估不同掩蔽策略的效果。

请注意，掩码策略的目的是减少不匹配在预训练和微调之间，因为[MASK]符号在微调阶段不会出现。我们报告两者的开发结果MNLI和NER。对于NER，我们报告了微调和基于特征的方法，正如我们预期的那样，对于基于特征的方法，不匹配将被放大模型将没有机会调整表示。

结果如表8所示。在表格中，掩码意味着我们将目标标记替换为[Mask]符号对于传销;相同的意思是保持目标token不变;RND 意味着我们将目标token替换为另一个随机token。

表左侧的数字表示在MLM预训练期间使用的特定策略的概率(BERT使用80%，10%，10%)。论文的右边是开发集的结果。对于基于特征的方法，我们

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI Fine-tune	NER Fine-tune	NER Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

Table 8: 消融不同的掩蔽策略。

将BERT的最后4层连接起来作为特征在5.3部分被证明是最好的方法。

从表中可以看出，微调的鲁棒性令人惊讶不同的掩蔽策略。不过，不出所料，只使用了掩码策略在将基于特征的方法应用于NER时存在问题。有趣的是，只使用RND策略的性能也比我们的策略差得多。

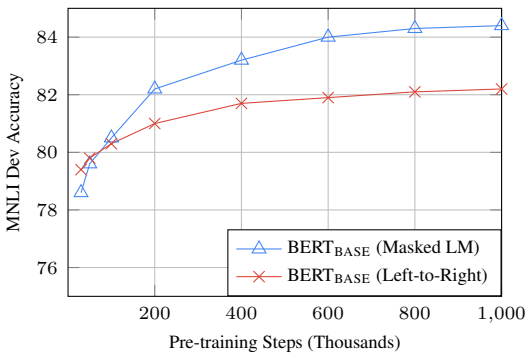


Figure 5: 消融训练步数。这显示了微调后的MNLI精度，从已为k步骤预训练的模型参数开始。x轴是k的值。