BERT，全称为Bidirectional Encoder Representations from Transformers，是Google于2018年发布的一种预训练语言模型，被认为是自然语言处理（NLP）领域的里程碑之一。BERT采用了自注意力机制和Transformer编码器，可以将输入的自然语言文本转化为向量表示，用于各种NLP任务。
BERT的出现极大地推动了自然语言处理领域的发展，因为它可以在不同的NLP任务中获得最佳结果。BERT不仅可以进行常见的NLP任务，如情感分析、问答和机器翻译等，还可以解决一些复杂的任务，例如文本生成和文本分类。
BERT的关键创新之一是双向Transformer编码器，这是一种基于神经网络的序列到序列模型，可以有效地捕捉上下文和语义信息。此外，BERT还引入了掩码语言模型和下一句预测两种预训练任务，这使得它可以更好地理解输入文本的含义和语境。
除了BERT，还有一些其他的预训练语言模型，例如GPT、XLNet和RoBERTa等。这些模型也采用了Transformer编码器和其他创新技术，但它们之间的区别在于训练目标、数据集和预训练技巧等方面。因此，选择适合特定任务的预训练语言模型需要进行一些实验和比较。
总之，BERT是自然语言处理领域的一项重要成果，它为各种NLP任务提供了强大的工具和技术。未来，预训练语言模型的发展将继续深入，从而使得计算机可以更加智能地处理和理解自然语言。

BERT：预训练深度双向Tranformers
Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova Google语言模型
摘要
我们介绍一个新的语言模型BERT：基于Transformers的双向编码表示。与最近的语言表示模型不同（Peters et al., 2018a; Rad- ford et al., 2018），BERT可以从未标注的文本中预训练深度双向表示，在Transformer的每一层都可以从左和右两个方向学习。结果表明，预训练模型加上一个输出层来进行微调，在一系列任务上表现优异，比如问答任务、语言推理，而且不需要做模型架构上的大调整。
BERT不仅理论简单，而且实战中表现强大。它在11种NLP任务上均获得了优异的成绩，比如GLUE分数达到了80.5%（提升了7.7%），MultiNLI准确率达到了86.7%（提升了4.6%），SQuAD v1.1问答测试中F1分数达到了93.2（提升了1.5分），SQuAD v2.0测试中F1分数达到了83.1分（提升了5.1分）。
1 引言
语言模型预训练已经被证实可以有效得提升许多NLP任务（Dai and Le, 2015; Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018）。这些任务有句子级（语言推理）、段落级（预测名子间关系）和token级（NER&问答Tjong Kim Sang and De Meulder, 2003; Rajpurkar et al., 2016）。
当前，在预训练模型上训练下游任务有两种策略：基于特征和基于微调。基于特征的方法：将预训练的特征表示添加到任务的特征输入端，比如Elmo。基于微调的方法：在任务中引入少量参数，并在训练任务时，简单微调预训练模型参数，比如Generative Pre-trained Transformer （OpenAI GPT）（Radford et al., 2018）。这两种方法用的都是一样的目标函数，而且都是用的单向语言模型来学习通过语言表示。

我们要说的是：当前的技术限制了预训练模型表示的威力。最大的限制是标准的语言模型是单向的，这限制了预训练模型的架构选择。例如，在PpenAIGPT中，作者使用了从左到右的架构，这样的话，在自注意力层，每个token只能注意到它前面的token(Vaswani et al., 2017)。这样的限制，当微调时，对句子级的任务影响还没那么大，但是对token级的任务影响非常大，比如问答任务，因为问答任务对上下文的理解非常重要。

在本文，我们通过提出BERT（双向编码表示的Transformer）来提升了基于微调的方法效果。BERT通过使用掩码语言模型（MLM）作为预训练目标，降低了之前的提到的单向限制，MLM是受完形填空任务的启发（Taylor, 1953）。掩码语言模型是从输入的句子中随机得盖住一些词，然后通过左右的内容来预测盖住的词。不像从左到右的预训练模型，MLM的目标是融合了内容的两端的信息，这样我们可以训练一个双向的Transformer。除了掩码语言模型，我们还用了“下一个句子预测”任务，两个任务一起训练文本对。我们论文的贡献有以下几点：

1）我们展示了双向预训练语言表示的重要性。与“Radford et al. (2018) Radford et al. (2018)”所使用的单向预训练语言模型不同，BERT使用MLM模型来实现深度双向语言表示。这也和“Peters et al. (2018a)”所使用从左到右且从右到左的浅层网络连接也不同。

2）我们还证明了，预训练模型可以大大减少任务的网络架构工作。BERT是第一个在许多NLP任务上表现出色的预训练模型，包括句子级和token级。

3）BERT在11个NLP任务上表现出色，代码和预训练模型在https://github.com/ google-research/bert。

2 相关工作

预训练通过语言表示有很长的历史了，本小节，我们简要得回顾一下使用最广泛的一些方法。

2.1 无监督基于特征的方法

学习大量的词表示是一个火热的领域，已经存在了几十年了，这包括非神经网络(Brown et al., 1992; Ando and Zhang, 2005; Blitzer et al., 2006)和神经网络(Mikolov et al., 2013; Pennington et al., 2014)的方法。预训练的词向量是现代NLP系统的重要组成部分，相较于从头开始训练的词向量(Turian et al., 2010)，它可以带来显著的改进。为了预训练词向量，有人用了从左至右的语言模型作为目标来训练（(Mnih and Hinton, 2009)），有人用从一段话从区分正确和错的词作为目标来训练(Mikolov et al., 2013)。

这些方法也推广到了更粗的粒度，例如句子向量(Kiros et al., 2015; Logeswaran and Lee,2018)和段落向量(Le and Mikolov,2014)。为了训练句子表示，前人的工作有：对候选句子排名(Jernite et al., 2017; Logeswaran and Lee, 2018)，基于前一句的表示预测下一个句子的词(Kiros et al., 2015)，去噪自编码器目标函数(Hill et al., 2016)。

Elmo和前人(Peters et al., 2017,2018a)在不同的维度推广了词嵌入研究。他们从从左到右和从右到左的语言模型中提取上下文敏感的特征。每个token表示都是从左到右的表示和从右到左的表示的连接。当把基于上下文的词嵌入集成到现有的任务中时，Elmo在一些主要的NLP任务上的基准测试(Peters et al., 2018a)中取得了优异的成绩，包括问答任务(Rajpurkar et al., 2016)，句法分析(Socher et al., 2013)，实体识别(Tjong Kim Sang and De Meulder,2003)。Melamud et al. (2016)等人提出了一种学习上下文表示的方法，即通过LSTMs来学习上下文，进而预测一个单词。与Elmo类似，他们的模

型也是基于特征的，而且不是深度双向的。Fedus et al. (2018)等人指出完形填空任务可以提升文本生成模型的鲁棒性。

2.2 无监督基于微调的方法

和基于特征的方法一样，这个方向最初使用从无标注的文本中预训练的词嵌入参数(Col-lobert and Weston, 2008)。

最近，句子或文档编码器已经可以从无标注文本中进行预训练，并应用于下游的监督任务(Dai and Le, 2015; Howard and Ruder, 2018; Radford et al., 2018)，其中这个编码器可以产生基于上理文的token表示。这样做的好处是只有少量参数需要从头开始学习。OpenAIGPT(Radford et al., 2018)能在GLUE基准测试(Wang et al., 2018a)的许多句子级的任务上达到了优异的成绩，至少有一部分是归功于这个好处。从左到右的语言模型以及自编码器已经用于这些模型的预训练(Howard and Ruder, 2018; Radford et al., 2018; Dai and Le, 2015)。

2.3 从监督数据进行迁移学习

有些工作是可以从预训练模型进行迁移的，比如语言推理和机器翻译，预训练模型是从大量数据中进行监督学习得到。计算机视觉研究已经证明了从大模型迁移学习的重要性，其中一个有效得方法是从ImageNet中预训练的模型进行微调(Deng et al., 2009; Yosinski et al., 2014)。

3 BERT

在本小节，我们介绍BERT和它的实现细节。我们的框架包含两步：预训练和微调。在预训练时，模型基于未标注的数据进行多个预训练任务学习。在微调时，BERT首先用预训练参数初始化，然后在下游任务，使用标注的数据进行所有参数的更新。每个下游任务使用单独的微调模型，尽管他们在初始化时预训练参数是一样的。本小节的图1会展示一个问答任务的例子。BERT的一个独特的特点是它在不同的任务上有统一的架构。预训练的架构和最终下游任务的架构差异非常小。

模型架构 BERT的模型架构是一个多层双向Transformer编码器，Transformer技术细节见Vaswani et al. (2017)等人的论文，Transformer的代码已在tensor2tensor的库里发版。因为Transformers已经非常普遍，而且我们的实现细节和它几乎一样，我们就不介绍这个背景知识了，读者有需要可以去阅读Vaswani et al. (2017)等人的论文，也可以阅读指南“The Annotated Transformer”。

在我们的工作中，我们定义层的数量为L，隐藏层大小为H，自注意力头的数量为A。我主要展示两个模型的结果：**BERT-**base(L=12, H=768, A=12, Total Parameters=110M)和**BERT**-large(L=24, H=1024, A=16, Total Parameters=340M)。**BERT**-base是用来和同尺寸的OpenAI GPT来作对比的。重要的是，BERT Transformer使用双向自注意力机，而GPT Transformer使用了限制的自注意力机制，它的每一个token只能关注它之前的内容。

输入/输出表示 为了使模型能够处理多种下游任务，我们的输入可以明确地表示单个句子或句子对（例如，<问题，答案>）。在这个工作中，“句子”可以是任意跨度的连续文本，而不一定是实际语言的名子。输入到BERT的token序列，可以是一个单独的名子或者两个句子。

我们使用WordPiece词嵌入(Wu et al.,2016)，词库总共有30000个token。每句话的开头是一个特殊的分类标识符([CLS])，对应最后的输出可以代表句子的分类。句子对可以合成一个句子，我们用两种方法来区分句子。首先，我们用一个特殊的符号([SEP])隔开两个句子。其次，我们给每一个token添加

了一个学习的embedding，以此来区分这个token是属于句子A还是句子B。如图1所示，我们将输入表示为E，最后输出中对应[CLS]的为C∈ R-H（H代表上标），第i个token的输届表示为 Ti ∈ R-H。

图1：BERT预训练和微调总览。除去输出层，预训练和微调的网络架构是一样的。不同的下游任务使用相同的预训练模型参数。微调时，所有参数进行更新。[CLS]是句前的一个特殊符号，[SEP]是一个句中的特殊符号（例如，区分问题和答案）。

For a given token, its input representation is constructed by summing the corresponding token,segment, and position embeddings. A visualiza-tion of this construction can be seen in Figure 2.

对于给定的一个token，它的输入是由token、句子和位置三方面的embedding相加。图2展示了这个过程。

图2 BERT输入表示。输入由token embedding、句子embedding和位置embedding三方面相加而成。

3.1 BERT预训练

和 Peters et al. (2018a) and Radford et al.(2018)的方法不同，我们不用传统的从左至右和从右至左的语言模型去预训练BERT。相反，我们用了两个无监督任务去预训练BERT。图1的左半部分展示了这个过程。

任务1：Masked LM 直觉上来说，有理由相信深度双向模型，比从左至右的模型或者从左至右及从右至左模型更强大。不幸的是，标准的语言模型只从左至右或从右至左进行训练，因为双向条件将会使每个单词间接地“看到自己“，而且模型可以在多层上文中轻易地预测目标词。

因此，为了训练深度双向表示，我们从输入中随机得地盖住一定比例的token，然后预测这些被盖住的tokens。我们称这个过程为”masked LM“(MLM)，尽管文献(Taylor, 1953)中经常称它为完形填空。然后将被盖住的这些token最后对应的隐藏层输出向量送进softmax函数，softmax函数的长度是词库的长度。在我们的所有实验中，每个句子，我们随机盖住了WordPiece token的15%。与去噪自编码器 (Vincent et al., 2008)不同，我们仅预测盖住的词，而不是重构输入。

尽管我们得到了一个双向预训练模型，但是有个缺点，训练和微调不匹配，因为[MASK]这个token在微调时不会出现的。为解决这个问题，我们并不都是用实际的[MASK] token来替换这个被盖住的词。当训练数据选择15%的数据进行预测时，如果某一个token被选中了，我们以80%的概率用[MASK]这个词来替换，以10%和概率用随机词来替换，以10%的概率不替换。然后就用交叉熵损失去预测这个被选中的词。我们在附录C.2对比这个过程的各种变化。

任务2： 下一个句子预测(NSP)

许多重要的下游任务，比如问答和自然语言推理是基于对句子间关系的理解，而这个关系是语言模型不能直接捕捉的。为了训练可以理解句子关系的模型，我们构造二分类的下一个句子预测的任务来进行预训练，而且这个任务可以轻易地基于单个的语料库生成。举例来说，当选择两个句子A和B作为一个样本时，有50%的概率B是A的下一个句子（标注为”IsNext“）,而且有50%的概率B是随机的（标注为”NotNext“）。正如图1所示，C就是用来进行下一个句子预测的(NSP)。尽管它很简单，我们实验证明了用这个任务进行预训练对QA和NLI非常有好处，5.1小节显示了这个过程。

NSP任务和 Jernite et al. (2017) and Logeswaran and Lee (2018)研究的”representation-learning objectives“非常相似。然而，在之前的工作中，只有句子的embedding迁移到下游任务中，BERT是将所有参数拿来初始化下游任务的参数。

预训练数据 预训练过程在很大程度上遵循了现有语言模型的文献。我们使用了BooksCorpus (800M words) (Zhu et al., 2015) 和英文维基百科 (2,500M words)作为预训练的语料库。对于维基百科，我们仅抽取了文本段落，去除了列表、表格和标题。使用文档级语料库而不是像Billion Word Benchmark（Chelba等，2013）这样的句级语料库非常重要，这是为了提取长的连续序列。

3.2 BERT微调

微调很简单，因为Transformer的自注意力机制是可以让BERT在许多下游任务上建模的，只需要改变一下输入和输出即可，而且输入既可以是单句也可以是句子对。

对于文本对，一个常见的做法是，先分别对每个文本进行编码操作，然后进行双向交叉注意力操作，比如 Parikh et al.(2016); Seo et al. (2017)等人的做法。

而BERT的做法是使用自注意力机制来合并这两步，因为将文本对连起来一起进行自注意力编码，自然就包括了对两个句子进行双向交叉注意力。

对每个任务，我们只需要简单地将输入和输出放进BERT，然后用端到端的方式微调所有参数。在输入端，句子A和句子B就类似于（1）释义中的句子对，（2）蕴含中的假设前提对，（3）问答中的问题和答案，（4）文本分类或序列标注中的退化文本本对。在输出端，token表示将会被送到输出层来完成token级的任务，例如序列标注或问答，[CLS]表示将会被送到输出层进行分类，例如蕴含或情感分析。

与预训练相比，微调是相对便宜的。论文中的所有结果在TPU最多一个小时可以复现，在GPU上也只需要几个小时。在第四小节我们详细描述了任务的细节。更多的细节详见附录A.5。

4 实验

在本小节，我们展示BERT在11个NLP任务上的微调结果。

4.1 GLUE

通用语言理解评估(General Language Understanding Evaluation-GLUE)基准测试是一系列不同的自然语言理解任务。GLUE数据集的详细描述见附录B.1。

在GLUE上微调时，我们先输入一个序列（一个句子或句子对），然后将C ∈ RH作为第一个输入[CLS]的输出。微调时唯一的新参数是分类层权重W ∈ R-K×H，这里K是标签的数量。最后基于C和W计算标准的分类损失，比如log(softmax(CW-T))。

对所有的GLUE任务，我们使用的参数batch size为32，参数epochs为3。每个微调任务，我们基于验证集选择了最好的学习率(5e-5, 4e-5, 3e-5, 或2e-5)。此外，对于BERTlarge模型，我们发现微调有时候在小数据集上不稳定，因此我们进行了几轮随机重启实验并基于验证集选择了最好的模型。对于随机重启，我们使用了相同的预训练参数，但是微调的数据和分类层的参数是不一样的。

结果展示在了表1。BERTbase和BERTlarge在所有任务上都大幅优于现有的模型，平均准确率比之前最好的水平分别提升了4.5%和7.0%。这里，除了注意力掩码，BERTbase和OpenAI GPT在模型架构上完全一样。对于最大和最广泛报告的GLUE任务，MNLI，BERT绝对准确率提升了4.6%。在GLUE的官方排行榜上，BERTlarge取得了80.5分，而OpenAI GPT当前为72.8。

表1 GLUE测试结果，分数由评估网站(https://gluebenchmark.com/leaderboard)生成。任务下面的数字代表样本数，“平均”这一列和官方的GLUE分数略有不同，因为我们去掉了有问题的数据集WNLI。BERT和OpenAI GPT是单模型，单任务。QQP和MRPC用的是F1分数，STS-B用的是Spearman相关性，其它任务用的是准确率。那些用BERT作为组件的模型我们没有列出来。
我们发现BERTlarge比BERTbase的表现要好，尤其是对于小数据集，关于模型大小的影响将在5.2节进行详细的探讨。

4.2 问答任务-SQuAD v1.1

斯坦幅问答数据集 (SQuAD v1.1)是由10万个众包问/答对组成的(Rajpurkar et al., 2016)。该任务是，给定一个问题和一个包含答案的段落，从这个段落里面预测答案所在文本片段。
如图1，在问答任务中，我们将输入的问题和段落合成一个序列，问题使用嵌入A，段落使用嵌入B。在微调时，我们引入了一个开始向量S ∈ R-H和结束向量E ∈ R-H。计算S和所有单词的内积，然后执行softmax操作，得到第i个单词属于开头部分的概率：Pi = eS·Ti / ∑ eS·Tj。同样的地方法可以得到第i个单词属于结尾的概率。候选答案的得分可以表示为S·Ti + E·Tj，其中j>=i，得分最大对应的i和j即是预测到的答案范围。训练目标函数就是正确答案的开始和结束位置的内积对数似然之和，即log(S·Ti) + log(E·Tj)。我们微调时使用的参数为，epochs=3，leraning rate=5e-5，batch size=32。（没有理解，待研究）
表2展示了顶级排行榜以及顶级系统的结果(Seo et al., 2017; Clark and Gardner, 2018; Peters et al., 2018a; Hu et al., 2018)。 这些顶级的结果没有最新的有效系统信息，而且训练时允许使用任何公开数据，因此我们进行了数据增强，具体来说，就是在微调时，先使用数据TriviaQA(Joshi et al., 2017)，再使用数据SQuAD。

表2 SQuAD 1.1结果。BERT集成是用了7个系统，每个系统使用不同的检查点和微调种子。我们最好的系统比集成方法的最好系统F1高1.5，比最好的单个模型的系统F1高1.3。实际上，就F1来说，我们单个BERT模型就打败了集成方法。即使不用数据集TriviaQA来微调，我们F1也仅仅输了0.1到0.4，仍然超越了现有系统一大截。

4.3 SQuAD v2.0

SQuAD 2.0任务延伸了SQuAD 1.1的问题定义，允许答案不在给定的段落里，这样使得问题更加灵活。We use a simple approach to extend the SQuAD v1.1 BERT model for this task. We treat ques-tions that do not have an answer as having an an-swer span with start and end at the [CLS] to-ken. The probability space for the start and end answer span positions is extended to include the position of the [CLS] token. For prediction, we compare the score of the no-answer span: snull =S·C + E·C to the score of the best non-null span si,j = maxj≥iS·Ti + E·Tj. We predict a non-null answer when ˆsi,j > snull + τ, where the thresh-old τ is selected on the dev set to maximize F1.We did not use TriviaQA data for this model. We fine-tuned for 2 epochs with a learning rate of 5e-5 and a batch size of 48。（没有理解，待翻译）
表3列出了结果，并与这前的排行榜及顶级公开工作进行了对比，其中去掉了使用BERT作为组件的模型。与之前最好的系统相比，我们的F1有5.1分的提升。

表3 SQuAD 2.0结果。我们去掉了把BERT作为组件的模型

4.4 SWAG(The Situations With Adversarial Generations)

The Situations With Adversarial Generations (SWAG) dataset contains 113k sentence-pair com- pletion examples that evaluate grounded common- sense inference (Zellers et al., 2018). Given a sen- tence, the task is to choose the most plausible con- tinuation among four choices.

“对抗生成情景”(SWAG)数据集包含11.3万个句子对，评估基于常识的推理 (Zellers et al., 2018)。给定一个句子，任务从四个选项中选择一个最可能的延续。

微调时，我们构建了四个输入，每一个包含了给定的句子（句子A）和一个可能的延续（句子B）。任务新增的参数就是一个二维向量W(H*4)，[CLS]的输出表示C与W矩阵相乘得到一个包含4个元素的一维向量，然后用softmax归一化得到每个选项的概率。

我们微调这个模型使有的参数有，epochs=3、learning_rate=2e-5、batch_size=16。结果展示见表4。BERTlarge击败了作者的基线模型ESIM+ELMo，提升了27.1%，相较于OpenAI GPT提升了8.3%。

表4：SWAG验证和测试准确率。专家表现是基于100个样本测算的，具体见SWAG论文。

5 消融实验

在这一小节，我们对BERT的不同部分进行了消融实验，以更好地理解它们的相对重要性。更多的消融实验见附录C。

5.1 预训练任务的影响

我们通过评估两个预训练的目标来展示了BERT的深度双向性的重要性，其中两个预训练的数据、微调方案及超参都是和BERTbase一样的。两个训练目标如下：

No NSP：就是使用了MLM，但是没有使用NSP的双向模型。

LTR & No NSP：我们训练了一个只考虑左文本的模型，用的是标准的从左至右语言模型，而不是MLM。同样地，在微调时出使用了左文本限制，因为如果不使用左文本限制的话，将会带来预训练和微调不一致的情况，进而降低下游任务的表现。此外，这个模型也没有用NSP任务。这样是可以直接和OpenAI的GPT对比的，区别是我们用得是更大的数据集，自己的输入表示和微调方案。

我们首先研究了NSP的任务的影响。在表5中，我们展示了去除NSP在QNLI、MNLI和SQuAD上带来的影响。接来我们评估了双向表示的影响，主要是通过比较“NO NSP”和“LTR & No NSP”来进行的，也就是比较MLM和单向语言模型的差异。结果显示，LTR模型的表示在所有任务上均差于MLM，其中在MRPC和SQuAD差距特别大。

表5：基本BERTbase模型架构进行消融实验。“NO NSP”是指不使用下一个句子任务。“LTR & NSP”是指从左至右的模型，而且不使用NSP任务，这像OpenAI的GPT一样。“+BiLSTM”是指在微调时在“LTR+No NSP”的顶部增加了一个随机初始化的BiLSTM。

对于SQuAD，直觉上看，可以很清晰地了解到，LTR模型在token预测上会很差，因为token级的隐藏层状态没有右侧文本的的信息。为了提升LTR系统，我们在顶部增加了一个随机初始化的BiLSTM。这个改进确实提升了SQuAD的效果，但是结果和预训练的双向模型还是有差距的。

我们意识到，也可以先训练一个LTR和一个RTL模型，然后将二者连接起来，就是ELMo一样。然而：(a)这将比双向模型成本增加一倍；(b)这也不符合直觉，因为RTL模型无法基于问题来确定答案；(c)严格来说这没有深度双向模型强大，因为后者是可以在多层都使用上下文信息。

5.2 模型尺寸的影响

在本小节，我们探索了模型尺寸对微调任务的准确率影响。我们使用不同的层数、隐藏层的长度（每个与token相应的向量长度）和注意力头数训练了一系列BERT模型，同时使用的是前面提过的相同的超参和训练过程。

表6展示了GLUE任务的结果。在这个表中，我们展示的是5次随机重启的验证集平均准确率。我们可以发现更大的模型在四个数据集上都得到了准确率的提升，即使像MRPC这样只有3600个标记样本，且与预训练任务很大不同的数据集也不例外。也许令我们惊讶的是，我们也可以超越现有文献中所提及的大模型。例如，Vaswani et al. (2017)所提及的最大的Transformer模型是（L=6, H=1024, A=16）1亿个参数，在文献中发现的最大的Transformer是(L=64, H=512, A=2)2.35亿个参数(Al-Rfou et al., 2018)。相比，BERTbase是1.1亿个参数，BERTlarge是3.4亿个参数。

表6：模型尺寸的消融实验。#L=层数；#H=隐层大小；#A=头数。“LM(ppl)”是训练数据中MLM任务的困惑度。

长久以来，我们似乎已经知道，提升模型大小将会在大规模的任务上得到持续提升，例如机器翻译和语言模型，这已经在表6的LM困惑度上得到了证明。然而，我们相信这是第一篇文章来充分证明，即使在小规模的任务上，模型尺寸增长仍带来性能的快速提升，当然这个前提是模型做了充分的预训练。Peters et al. (2018b)等提升将预训练的Bi-LM模型的层数从2提升4对下游任务的影响有好有坏，Melamud et al. (2016)等人提出将隐层维度尺寸从200增到600是可能提升性能，但是增加1000就不行。这两种工作都是使用的基于特征的方法。本文的假设是当模型直接在下游任务上微调，而且仅使用随机的一些参数，这个时候，更大的尺寸的，更具表现力的预训练表示将会有利于下游任务，即使下游任务数据量很小也是这个效果。

5.3 BERT用于基于特征的方法

目前为止所有的BERT结果都是使用微调的方法，这个方法就是将一个分类层添加到预训练模型上，然后所有参数一起训练。然而，基于特征的方法，即从预训练模型中提取固定特征的方法，也有它的优点。首先，不是所有任务都可以用Transformer的架构来表示的，因此需要添加特定任务的模型架构。其次，是有计算优势的：在训练数据中一次性得到昂贵表示后，在昂贵的表示之上可以运行相对便宜的模型。

在本小节，我们BERT应用到CoNLL-2003实体识别任务上 (Tjong Kim Sang and De Meulder, 2003)，以此来对比基于特征和基于微调的差异。在BERT的输入端，我们使用了一个区分大小的WordPiece模型，而且包含了最大文档上下文。基于标准的做法，我们将此作为标注任务，但是没有使用CRF作为输出。我们使用了第一个子token表示来作为token级任务分类的输入。

为了消除微调的影响，我们在不微调BERT任何参数的情况下，提取了各层的激活层输出。这些基于上下文的embeddings被 送进两层的768维的BiLSTM，然后进入分类层。

结果见表7.BERTlarge表现最好。最好的基于特征的方法是将后四层的输出连接起来，比微调整个模型F1也就低了0.3分。这证明了BERT在基于微调和基于特征两种方法上都表现优异。

6 结论

最近的一些NLP研究中的性能提升，说明了基于大量，无监督的预训练模型进行迁移学习是许多NLP任务的重要组成部分。特别是，在深度单向架构中，即使低资源消耗的任务，也可以提升性能。我们的贡献是将这一发现进一步推广到深度双向网络，将同样的预训练模型能够成功得处理大量的NLP任务