一、BERT:Pre-training of Deep Bidirectional Transformers for Language Understanding 论文精读

摘要

我们提出了一种新的称为 BERT 的语言表示模型，BERT 代表来自 Transformer 的双向编码器表示（**B**idirectional **E**ncoder **R**epresentations from **T**ransformers）。不同于最近的语言表示模型（Peters et al., 2018，Radford et al., 2018）， BERT 旨在通过联合调节所有层中的左右上下文来预训练深度双向表示。因此，只需要一个额外的输出层，就可以对预训练的 BERT 表示进行微调，从而为广泛的任务（比如回答问题和语言推断任务）创建最先进的模型，而无需对特定于任务进行大量模型结构的修改。

BERT 的概念很简单，但实验效果很强大。它刷新了 11 个 NLP 任务的当前最优结果，包括将 GLUE 基准提升至 80.4%（7.6% 的绝对改进）、将 MultiNLI 的准确率提高到 86.7%（5.6% 的绝对改进），以及将 SQuAD v1.1 的问答测试 F1 得分提高至 93.2 分（提高 1.5 分）——比人类表现还高出 2 分。

1. 介绍

语言模型预训练可以显著提高许多自然语言处理任务的效果（Dai and Le, 2015；Peters et al., 2018；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（Bow-man et al., 2015；Williams et al., 2018）和释义（Dolan and Brockett, 2005），目的是通过对句子的整体分析来预测句子之间的关系，以及标记级任务，如命名实体识别（Tjong Kim Sang and De Meulder, 2003）和 SQuAD 问答（Rajpurkar et al., 2016），模型需要在标记级生成细粒度的输出。

现有的两种方法可以将预训练好的语言模型表示应用到下游任务中：基于特征的和微调。基于特征的方法，如 ELMo （Peters et al., 2018)，使用特定于任务的模型结构，其中包含预训练的表示作为附加特特征。微调方法，如生成预训练 Transformer（OpenAI GPT)（Radford et al., 2018）模型，然后引入最小的特定于任务的参数，并通过简单地微调预训练模型的参数对下游任务进行训练。在之前的工作中，两种方法在预训练任务中都具有相同的目标函数，即使用单向的语言模型来学习通用的语言表达。

我们认为，当前的技术严重地限制了预训练表示的效果，特别是对于微调方法。主要的局限性是标准语言模型是单向的，这就限制了可以在预训练期间可以使用的模型结构的选择。例如，在 OpenAI GPT 中，作者使用了从左到右的模型结构，其中每个标记只能关注 Transformer 的自注意层中该标记前面的标记（Williams et al., 2018）。这些限制对于句子级别的任务来说是次优的（还可以接受），但当把基于微调的方法用来处理标记级别的任务（如 SQuAD 问答）时可能会造成不良的影响（Rajpurkar et al., 2016），因为在标记级别的任务下，从两个方向分析上下文是至关重要的。

在本文中，我们通过提出 BERT 改进了基于微调的方法：来自 Transformer 的双向编码器表示。受完形填空任务的启发，BERT 通过提出一个新的预训练任务来解决前面提到的单向约束：“遮蔽语言模型”（MLM masked language model）（Tay-lor, 1953）。遮蔽语言模型从输入中随机遮蔽一些标记，目的是仅根据被遮蔽标记的上下文来预测它对应的原始词汇的 id。与从左到右的语言模型预训练不同，MLM 目标允许表示融合左右上下文，这允许我们预训练一个深层双向 Transformer。除了遮蔽语言模型之外，我们还提出了一个联合预训练文本对来进行“下一个句子预测”的任务。

本文的贡献如下：

- 我们论证了双向预训练对语言表征的重要性。与 Radford et al., 2018 使用单向语言模型进行预训练不同，BERT 使用遮蔽语言模型来实现预训练深层双向表示。这

也与 Peters et al., 2018 的研究形成了对比，他们使用了一个由左到右和由右到左的独立训练语言模型的浅层连接。

- 我们表明，预训练的表示消除了许多特定于任务的高度工程化的的模型结构的需求。BERT 是第一个基于微调的表示模型，它在大量的句子级和标记级任务上实现了最先进的性能，优于许多特定于任务的结构的模型。
- BERT 为 11 个 NLP 任务提供了最先进的技术。我们还进行大量的消融研究，证明了我们模型的双向本质是最重要的新贡献。代码和预训练模型将可在 goo.gl/language/bert 获取。

2 相关工作

预训练通用语言表示有很长的历史，我们将在本节简要回顾最流行的方法。

2.1 基于特征的方法

几十年来，学习广泛适用的词语表示一直是一个活跃的研究领域，包括非神经网络学领域（Brown et al., 1992;;Blitzer et al., 2006）和神经网络领域（Collobert and Weston, 2008；Mikolov et al., 2013；Pennington et al., 2014）方法。经过预训练的词嵌入被认为是现代 NLP 系统的一个不可分割的部分，词嵌入提供了比从头开始学习的显著改进（Turian et al., 2010）。

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）。与传统的单词嵌入一样，这些学习到的表示通常也用作下游模型的输入特征。

ELMo（Peters et al., 2017）从不同的维度对传统的词嵌入研究进行了概括。他们建议从语言模型中提取上下文敏感的特征。在将上下文嵌入与特定于任务的架构集成时，ELMo 为几个主要的 NLP 标准提供了最先进的技术（Peters et al., 2018)，包括在 SQuAD 上的问答（Rajpurkar et al., 2016），情感分析（Socher et al., 2013），和命名实体识别（jong Kim Sang and De Meul-der, 2003）。

2.2 基于微调的方法

语言模型迁移学习（LMs）的一个最新趋势是，在对受监督的下游任务的模型进行微调之前，先对 LM 目标上的一些模型构造进行预训练（Dai and Le, 2015；Howard and Ruder, 2018；Radford et al., 2018）。这些方法的优点是只有很少的参数需要从头开始学习。至少部分得益于这一优势，OpenAI GPT （Radford et al., 2018）在 GLUE 基准测试的许多句子级任务上取得了此前最先进的结果（Wang et al.(2018)）。

2.3 从有监督的数据中迁移学习

虽然无监督预训练的优点是可用的数据量几乎是无限的，但也有研究表明，从具有大数据集的监督任务中可以进行有效的迁移，如自然语言推理（Con-neau et al., 2017）和机器翻译（McCann et al., 2017）。在NLP之外，计算机视觉研究也证明了从大型预训练模型中进行迁移学习的重要性，有一个有效的方法可以微调在 ImageNet 上预训练的模型（Deng et al., 2009；Yosinski et al., 2014）

3 BERT

本节将介绍 BERT 及其具体实现。首先介绍了 BERT 模型结构和输入表示。然后我们在 3.3 节介绍本文的核心创新——预训练任务。在 3.4 和 3.5 节中分别详细介绍了预训练过程和微调模型过程。最后，在 3.6 节中讨论了 BERT 和 OpenAI GPT 之间的区别。

3.1 模型结构

BERT 的模型结构是一个基于 Vaswani et al.(2017) 描述的原始实现的多层双向 Transformer 编码器，并且 Transformer 编码器发布在 tensor2tensor 代码库中。由于最近 Transformer 的使用已经非常普遍，而且我们的实现与最初的实现实际上是相同的，所以我们将省略对模型结构的详尽的背景描述，并向读者推荐 Vaswani et al.(2017) 以及优秀的指南，如“带注释的 Transformer”。

在这项工作中，我们表示层的数量(即，Transformer 块)为 L，隐藏尺寸为 H，自注意头的个数为 A。在所有例子中，我们将前馈/过滤器的大小设置为 $4H$，即当 $H = 768$ 时是 3072；当 $H = 1024$ 是 4096。我们主要分析两个模型大小的结果:

- $BERT_{BASE}: L=12, H=768, A=12, TotalParameters=110M$
- $BERT_{LARGE}: L=24, H=1024, A=16, TotalParameters=340M$

为了方便比较，$BERT_{BASE}$ 选择了与 OpenAI GPT 一样的模型大小。然而，重要的是，BERT Transformer 使用的是双向的自注意力，而 GPT Transformer 使用的是受限的自注意力，每个标记只能关注其左边的语境。我们注意到，在文献中，双向 Transformer 通常被称为“Transformer 编码器”，而只有标记左侧语境的版本由于可以用于文本生成而被重新定义为“Transformer 解码器”。BERT、OpenAI GPT 和 ELMo 之间的比较如图 1 所示。

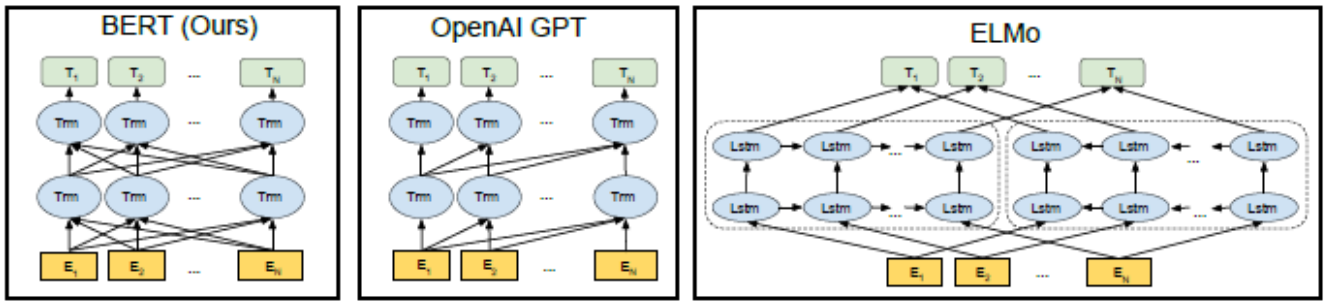


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

> *图 1：预训练模型结构的不同。BERT 使用双向 Transformer。OpenAI GPT 使用从左到右的 Transformer。ELMo 使用独立训练的从左到右和从右到左的 LSTM 的连接来为下游任务生成特征。其中，只有 BERT 表示在所有层中同时受到左右语境的制约。*

3.2 输入表示

我们的输入表示能够在一个标记序列中清楚地表示单个文本句子或一对文本句子(例如，[Question, Answer])。（注释：在整个工作中，“句子”可以是连续的任意跨度的文本，而不是实际语言意义上的句子。“序列”是指输入到 BERT 的标记序列，它可以是单个句子，也可以是两个句子组合在一起。）通过把给定标记对应的标记嵌入、句子嵌入和位置嵌入求和来构造其输入表示。图 2 给出了输入表示的可视化表示。
细节是:

- 我们使用含 3 万个标记词语的 WordPiece 嵌入（Wu et al., 2016）。我们用 ## 表示拆分的单词片段。

- 我们使用学习到的位置嵌入，支持的序列长度最长可达 512 个标记。
- 每个序列的第一个标记始终是特殊分类嵌入（[CLS]）。该特殊标记对应的最终隐藏状态（即，Transformer 的输出）被用作分类任务中该序列的总表示。对于非分类任务，这个最终隐藏状态将被忽略。
- 句子对被打包在一起形成一个单独的序列。我们用两种方法区分这些句子。方法一，我们用一个特殊标记（[SEP]）将它们分开。方法二，我们给第一个句子的每个标记添加一个可训练的句子 A 嵌入，给第二个句子的每个标记添加一个可训练的句子 B 嵌入。
- 对于单句输入，我们只使用句子 A 嵌入。

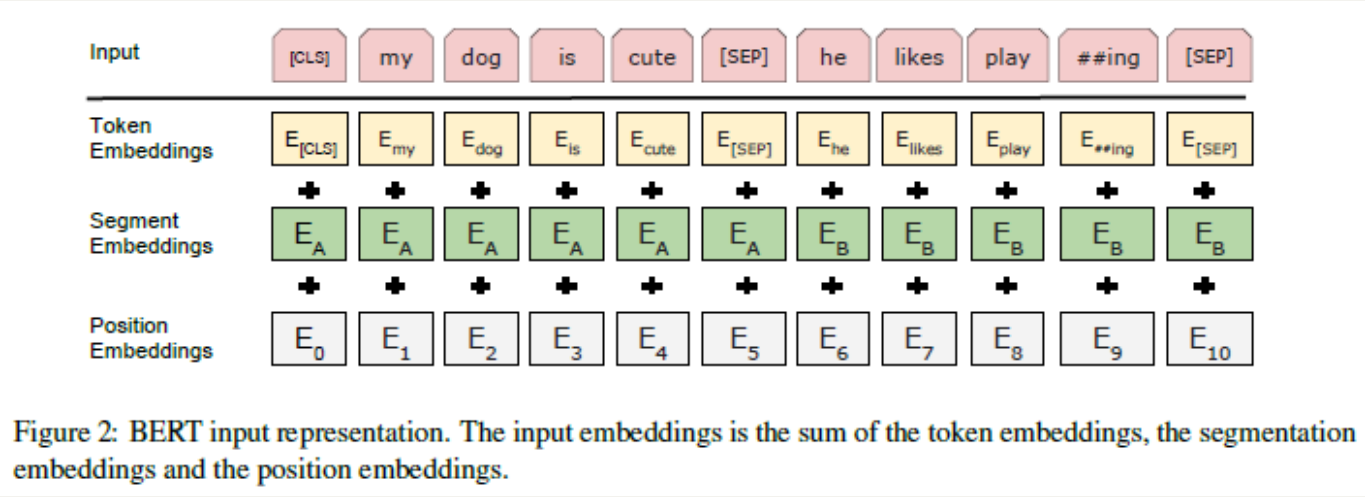


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

> *图 2：BERT 的输入表示。输入嵌入是标记嵌入（词嵌入）、句子嵌入和位置嵌入的总和。*

3.3.1 任务一#：遮蔽语言模型

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模型结合从右到左模型的浅层连接更强大。不幸的是，标准条件语言模型只能从左到右或从右到左进行训练，因为双向条件作用将允许每个单词在多层上下文中间接地“看到自己”。

为了训练深度双向表示，我们采用了一种简单的方法，即随机遮蔽一定比例的输入标记，然后仅预测那些被遮蔽的标记。我们将这个过程称为“遮蔽语言模型”（MLM），尽管在文献中它通常被称为完形填词任务（Taylor, 1953）。在这种情况下，就像在标准语言模型中一样，与遮蔽标记相对应的最终隐藏向量被输入到与词汇表对应的输出 softmax 中（也就是要把被遮蔽的标记对应为词汇表中的一个词语）。在我们所有的实验中，我们在每个序列中

随机遮蔽 15% 的标记。与去噪的自动编码器（Vincent et al., 2008）不同的是，我们只是让模型预测被遮蔽的标记，而不是要求模型重建整个输入。

虽然这确实允许我们获得一个双向预训练模型，但这种方法有两个缺点。第一个缺点是，我们在预训练和微调之间造成了不匹配，因为 [MASK] 标记在微调期间从未出现过。为了缓和这种情况，我们并不总是用真的用 [MASK] 标记替换被选择的单词。而是，训练数据生成器随机选择 15% 的标记，例如，在my dog is hairy 这句话中，它选择 hairy。然后执行以下步骤:

- 数据生成不会总是用 [MASK] 替换被选择的单词，而是执行以下操作:
- 80% 的情况下：用 [MASK] 替换被选择的单词，例如，my dog is hairy → my dog is [MASK]
- 10% 的情况下：用一个随机单词替换被选择的单词，例如，my dog is hairy → my dog is apple
- 10% 的情况下：保持被选择的单词不变，例如，my dog is hairy → my dog is hairy。这样做的目的是使表示偏向于实际观察到的词。

Transformer 编码器不知道它将被要求预测哪些单词，或者哪些单词已经被随机单词替换，因此它被迫保持每个输入标记的分布的上下文表示。另外，因为随机替换只发生在 1.5% 的标记（即，15% 的 10%）这似乎不会损害模型的语言理解能力。

第二个缺点是，使用 Transformer 的每批次数据中只有 15% 的标记被预测，这意味着模型可能需要更多的预训练步骤来收敛。在 5.3 节中，我们证明了 Transformer 确实比从左到右的模型（预测每个标记）稍微慢一点，但是 Transformer 模型的实验效果远远超过了它增加的预训练模型的成本。

3.3.2 任务2#：下一句预测

许多重要的下游任务，如问题回答（QA）和自然语言推理（NLI），都是建立在理解两个文本句子之间的关系的基础上的，而这并不是语言建模直接捕捉到的。为了训练一个理解句子关系的模型，我们预训练了一个下一句预测的二元分类任务，这个任务可以从任何单语语料库中简单地归纳出来。具体来说，在为每个训练前的例子选择句子 A 和 B 时，50% 的情况下 B 是真的在 A 后面的下一个句子，50% 的情况下是来自语料库的随机句子。比如说:

```
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
```

我们完全随机选择不是下一句的句子，最终的预训练模型在这个任务中达到了 97%-98% 的准确率。尽管这个任务很简单，但是我们在 5.1 节中展示了针对此任务的预训练对 QA 和 NLI 都非常有益。

3.4 预训练过程

预训练过程大体上遵循以往文献中语言模型预训练过程。对于预训练语料库，我们使用 BooksCorpus（800M 单词）（Zhu et al., 2015）和英语维基百科（2,500M 单词）。对于维基百科，我们只提取文本段落，而忽略列表、表格和标题。为了提取长的连续序列，使用文档级别的语料库，而不是使用像 Billion Word Benchmark （Chelba et al., 2013）那样使用打乱顺序的句子级别语料库是至关重要的。

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，我们将其称为“句子”，尽管它们通常比单个句子长得多（但也可以短一些）。第一个句子添加 A 嵌入，第二个句子添加 B 嵌入。50% 的情况下 B 确实是 A 后面的实际下一句，50% 的情况下它是随机选取的一个的句子，这是为“下一句预测”任务所做的。两句话合起来的长度要小于等于 512 个标记。语言模型遮蔽过程是在使用 WordPiece 序列化句子后，以均匀的 15% 的概率遮蔽标记，不考虑部分词片的影响（那些含有被 WordPiece 拆分，以##为前缀的标记）。

我们使用 256 个序列（256 个序列 * 512 个标记= 128,000 个标记/批次）的批大小进行 1,000,000 步的训练，这大约是在 33 亿词的语料库中训练 40 个周期。我们用Adam 优化算法并设置其学习率为 $1e-4$，$\beta 1 = 0.9, \beta 2 = 0.999$，$L2$ 的权重衰减是 0.01，并且在前 10000 步学习率热身（learning rate warmup），然后学习率开始线性衰减。我们在所有层上使用 0.1 概率的 dropout。像 OpenAI GPT 一样，我们使用 gelu 激活（Hendrycks and Gimpel, 2016）而不是标准 relu。训练损失是遮蔽语言模型似然值与下一句预测似然值的平均值。

在 4 块 Cloud TPU（共含有 16 块 TPU）上训练 $BERT_{BASE}$。在 16 块 Cloud TPU（共含有 64 块 TPU）训练 $BERT_{LARGE}$。每次训练前需要 4 天的时间。

3.5 微调过程

对于序列级别的分类任务，BERT 微调非常简单。为了获得输入序列的固定维度的表示，我们取特殊标记（[CLS]）构造相关的嵌入对应的最终的隐藏状态(即，为 Transformer 的输出)的池化后输出。我们把这个向量表示为 $C \in \mathbb{R}^H$，在微调期间唯一需要的新增加的参数是分类层的参数矩阵 $W \in \mathbb{R}^{K \times H}$，其中 K 是要分类标签的数量。分类标签的概率 $P \in \mathbb{R}^K$ 由一个标准的 softmax 来计算，$P = softmax(CW^T)$。对 BERT 的参数矩阵 W 的所有参数进行了联合微调，使正确标签的对数概率最大化。对于区间级和标记级预测任务，必须以特定于任务的方式稍微修改上述过程。具体过程见第 4 节的相关内容。

对于微调，除了批量大小、学习率和训练次数外，大多数模型超参数与预训练期间相同。Dropout 概率总是使用 0.1。最优超参数值是特定于任务的，但我们发现以下可能值的范围可以很好地在所有任务中工作:

- Batch size: 16, 32
- Learning rate (Adam): 5e-5, 3e-5, 2e-5
- Number of epochs: 3, 4

我们还观察到大数据集（例如 100k+ 标记的训练集）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，因此只需对上述参数进行完全搜索，并选择在验证集上性能最好的模型即可。

3.6 BERT 和 OpenAI GPT 的比较

在现有预训练方法中，与 BERT 最相似的是 OpenAI GPT，它在一个大的文本语料库中训练从左到右的 Transformer 语言模型。事实上，BERT 中的许多设计决策都是有意选择尽可能接近 GPT 的，这样两种方法就可以更加直接地进行比较。我们工作的核心论点是，在 3.3 节中提出的两项新的预训练语言模型任务占了实验效果改进的大部分，但是我们注意到 BERT 和 GPT 在如何训练方面还有其他几个不同之处:

- GPT 是在 BooksCorpus（800M 词）上训练出来的；BERT 是在 BooksCor-pus（800M 词）和 Wikipedia（2,500M 词）上训练出来的。

- GPT 仅在微调时使用句子分隔符（[SEP]）和分类标记（[CLS]）；BERT 在预训练时使用 [SEP]，[CLS] 和 A/B 句嵌入。
- GPT 在每批次含 32,000 词上训练了 1M 步；BERT 在每批次含 128,000 词上训练了 1M 步。
- GPT 在所有微调实验中学习速率均为 5e-5；BERT 选择特定于任务的在验证集中表现最好的微调学习率。

为了分清楚这些差异的带来的影响，我们在 5.1 节中的进行每一种差异的消融实验表明，大多数的实验效果的改善实际上来自新的预训练任务（遮蔽语言模型和下一句预测任务）。

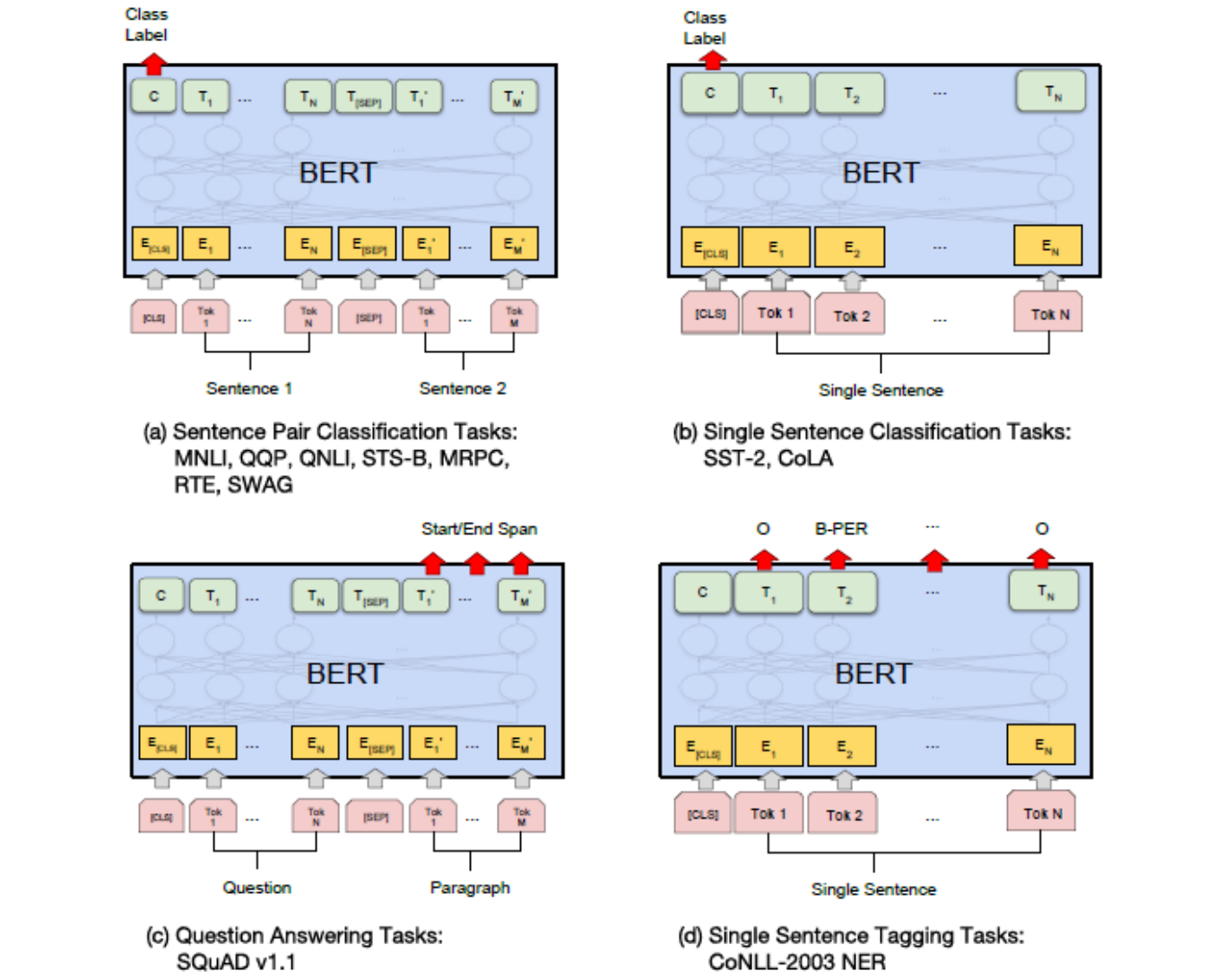


Figure 3: Our task specific models are formed by incorporating BERT with one additional output layer, so a minimal number of parameters need to be learned from scratch. Among the tasks, (a) and (b) are sequence-level tasks while (c) and (d) are token-level tasks. In the figure, E represents the input embedding, T_i represents the contextual representation of token i, [CLS] is the special symbol for classification output, and [SEP] is the special symbol to separate non-consecutive token sequences.

> *图 3：我们具体于特定任务的模型是通过给 BERT 加一个额外的输出层构成，所以仅需要从头学习最小数量的参数。其中（a）和（b）是序列级任务，（c）和（d）是标记级任务。图中 E 表示嵌入的输入，Ti 表示第 i 个标记的上下文表示，[CLS] 是分类输出的特殊符号，[SEP] 是分离非连续标记（分离两个句子）序列的特殊符号。*

4. 实验

在这一节，我们将展示 BERT 在 11 项自然语言处理任务中的微调结果。

4.1 GLUE 数据集

通用语言理解评价 (GLUE General Language Understanding Evaluation) 基准（Wang et al. (2018)）是对多种自然语言理解任务的集合。大多数 GLUE 数据集已经存在多年，但 GLUE 的用途是（1）以分离的训练集、验证集和测试集的标准形式发布这些数据集；并且（2）建立一个评估服务器来缓解评估不一致和过度拟合测试集的问题。GLUE 不发布测试集的标签，用户必须将他们的预测上传到 GLUE 服务器进行评估，并对提交的数量进行限制。

GLUE 基准包括以下数据集，其描述最初在 Wang et al.(2018)中总结:

MNLI 多类型的自然语言推理（Multi-Genre Natural Language Inference）是一项大规模的、众包的蕴含分类任务（Williams et al.， 2018）。给定一对句子，目的是预测第二个句子相对于第一个句子是暗含的、矛盾的还是中立的关系。

QQP Quora问题对（Quora Question Pairs）是一个二元分类任务，目的是确定两个问题在Quora上问的语义是否相等 （Chen et al., 2018）。

QNLI 问题自然语言推理（Question Natural Language Inference）是斯坦福问题回答数据集（Rajpurkar et al., 2016）已经转换为二进制分类任务的一个版本 Wang et al.(2018)。正类的例子是（问题，句子）对，句子中包含正确的答案，和负类的例子是来自同一段的（问题，句子）对，句子中不包含正确的答案。

SST-2 斯坦福情感语义树（Stanford Sentiment Treebank）数据集是一个二元单句分类任务，数据由电影评论中提取的句子组成，并对由人工对这些句子进行标注（Socher et al., 2013）。

CoLA 语言可接受性单句二元分类任务语料库（Corpus of Linguistic Acceptability），它的目的是预测一个英语句子在语言学上是否“可接受”（Warstadt et al., 2018）。

STS-B 文本语义相似度基准（Semantic Textual Similarity Bench-mark ）是从新闻标题中和其它来源里提取的句子对的集合（Cer et al., 2017）。他们用从 1 到 5 的分数标注，表示这两个句子在语义上是多么相似。

MRPC 微软研究释义语料库（Microsoft Research Paraphrase Corpus）从在线新闻中自动提取的句子对组成，并用人工注解来说明这两个句子在语义上是否相等（Dolan and Brockett, 2005.）。

RTE 识别文本蕴含（Recognizing Textual Entailment）是一个与 MNLI 相似的二元蕴含任务，只是 RTE 的训练数据更少 Bentivogli et al., 2009。

WNLI 威诺格拉德自然语言推理（Winograd NLI）是一个来自（Levesque et al., 2011) ）的小型自然语言推理数据集。GLUE网页提示到这个数据集的构造存在问题，每一个被提交给 GLUE 的经过训练的系统在预测多数类时都低于 65.1 这个基线准确度。因此，出于对 OpenAI GPT 的公平考虑，我们排除了这一数据集。对于我们的 GLUE 提交，我们总是预测多数类。

4.1.1 GLUE 结果

为了在 GLUE 上微调模型，我们按照本文第 3 节中描述的那样表示输入的句子或者句子对，并且使用最后一层的隐藏向量 $C \in \mathbb{R}^H$ 中的第一个输入标记（[CLS]）作为句子总的表示。如图3 （a）和（b）所示。在微调期间唯一引入的新的参数是一个分类层参数矩阵 $W \in \mathbb{R}^{K \times H}$，其中 K 是要分类的数量。我们用 C 和 W 计算一个标准的分类损失，换句话说是 $log(softmax(CW^T))$。

我们在 GLUE 所有的任务中使用 32 的批次大小和 3 个周期。对于每个任务我们使用 $5e-5, 4e-5, 3e-5, 2e-5$ 的学习率来微调，然后在验证集中选择表现最好的学习率。此外，对于 $BERT_{LARGE}$ 我们发现它有时在小数据集上微调时不稳定（换句话说是，有时运行时会使结果更差），因此，我们进行了几次随机重启，并选择了在验证集上表现最好的模型。对于随机重启，我们使用相同的预训练检查点，但执行不同的数据打乱和分类器层初始化来微调模型。我们注意到，GLUE 发布的数据集不包括测试的标签，所以我们分别将 $BERT_{BASE}$ 和 $BERT_{LARGE}$ 向 GLUE 评估服务器提交结果。

结果如表 1 所示。在所有的任务上，$BERT_{BASE}$ 和 $BERT_{LARGE}$ 都比现有的系统更加出色，与先进水平相比，分别取得 4.4% 及 6.7% 的平均改善。请注意，除了 $BERT_{BASE}$ 含有注意力屏蔽（attention masking），$BERT_{BASE}$ 和 OpenAI GPT 的模型结构方面几乎是相同的。对于最大和最广泛使用的 GLUE 任务 MNLI，BERT 比当前最优模型获得了 4.7% 的绝对提升。在 GLUE 官方的排行榜上，$BERT_{LARGE}$ 获得了 80.4 的分数，与原榜首的 OpenAI GPT 相比截止本文写作时只获得了 72.8 分。

有趣的是，$BERT_{LARGE}$ 在所有任务中都显著优于 $BERT_{BASE}$，即使是在那些只有很少训练数据的任务上。BERT 模型大小的影响在本文 5.2 节有更深入的探讨。

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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$BERT_{BASE}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Table 1: GLUE Test results, scored by the GLUE evaluation server. The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set. OpenAI GPT = (L=12, H=768, A=12); $BERT_{BASE}$ = (L=12, H=768, A=12); $BERT_{LARGE}$ = (L=24, H=1024, A=16). BERT and OpenAI GPT are single-model, single task. All results obtained from https://gluebenchmark.com/leaderboard and https://blog.openai.com/language-unsupervised/.

> *表 1：GLUE 测试结果，由 GLUE 评估服务器评分。每个任务下面的数字表示训练示例的数量。"Average"列与官方 GLUE 评分略有不同，因为我们排除了有问题的 WNLI 数据集。OpenAI GPT = (L=12, H=768, A=12); BERTBASE = (L=12, H=768, A=12); BERTLARGE = (L=24, H=1024, A=16)。BERT 和 OpenAI GPT 都是单模型，单任务。所有结果可以从 https://gluebenchmark.com/leaderboard 和 https://blog.openai.com/language-unsupervised/ 获得。*

4.2 SQuAD v1.1

斯坦福问答数据集（SQuAD Standford Question Answering Dataset）是一个由 100k 个众包的问题/答案对组成的集合（Rajpurkar et al., 2016）。给出一个问题和一段来自维基百科包含这个问题答案的段落，我们的任务是预测这段答案文字的区间。例如：

- 输入问题：
  Where do water droplets collide with ice crystals to form precipitation?
- 输入段落
  … Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. …
- 输出答案
  within a cloud

这种区间预测任务与 GLUE 的序列分类任务有很大的区别，但是我们能够让 BERT 以一种直接的方式在 SQuAD 上运行。就像在 GLUE 中，我们将输入问题和段落表示为一个单一打包序列（packed sequence），其中问题使用 A 嵌入，段落使用 B 嵌入。在微调模型期间唯一需要学习的新参数是区间开始向量 $S \in \mathbb{R}^H$ 和区间结束向量 $E \in \mathbb{R}^H$。让 BERT 模型最后一层的隐藏向量的第 i^{th} 输入标记被表示为 $T_i \in \mathbb{R}^H$。如图 3（c）可视化的表示。然后，计算单词 i 作为答案区间开始的概率，它是 T_i 和 S 之间的点积并除以该段落所有单词的结果之后再 softmax:

$$P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$$

同样的式子用来计算单词作为答案区间的结束的概率，并采用得分最高的区间作为预测结果。训练目标是正确的开始和结束位置的对数可能性。

我们使用 $5e-5$ 的学习率，32 的批次大小训练模型 3 个周期。在模型推断期间，因为结束位置与开始位置没有条件关系，我们增加了结束位置必须在开始位置之后的条件，但没有使用其他启发式。为了方便评估，把序列化后的标记区间对齐回原始未序列化的输入。

结果如表 2 中描述那样。SQuAD 使用一个高度严格的测试过程，其中提交者必须手动联系小组组织人员，然后在一个隐藏的测试集上运行他们的系统，所以我们只提交了最好的模型来测试。表中显示的结果是我们提交给小组的第一个也是唯一一个测试。我们注意到上面的结果在小组排行榜上没有最新的公共模型描述，并被允许在训练他们的模型时使用任何的公共数据。因此，我们在提交的模型中使用非常有限的数据增强，通过在 SQuAD 和 TriviaQA(Joshi et al., 2017) 联合训练。

我们表现最好的模型在集成模型排名中上比排名第一模型高出 1.5 个 F1 值，在一个单模型排行榜中比排名第一的模型高出 1.7（译者注：原文是 1.3） 个 F1 值。实际上，我们的单模型 BERT 就比最优的集成模型表现更优。即使只在 SQuAD 数据集上（不用 TriviaQA 数据集）我们只损失 0.1-0.4 个 F1 值，而且我们的模型输出结果仍然比现有模型的表现好很多。

System	Dev		Test	
	EM	F1	EM	F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
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: SQuAD results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

> *表 2：SQuAD 结果。Ensemble BERT 是使用不同的预训练模型检查点和微调种子的 7x 模型。*

4.3 命名实体识别

为了评估标记任务的性能，我们在 CoNLL 2003 命名实体识别数据集（NER Named Entity Recognition）上微调 BERT 模型。该数据集由 200k 个训练单词组成，这些训练词被标注为人员、组织、地点、杂项或其他（无命名实体）。

为了微调，我们将最后一层每个单词的隐藏表示 $T_i \in \mathbb{R}^H$ 送入一个在 NER 标签集合的分类层。每个单词的分类不以周围预测为条件（换句话说，没有自回归和没有 CRF）。为了与词块（WordPiece）序列化相适应，我们把 CoNLI-序列化的（CoNLL-tokenized）的输入词输入我们的 WordPiece 序列化器，然后使用这些隐藏状态相对应的第一个块而不用预测标记为 X的块。例如：

```
Jim   Hen   ##son was a puppet ##eer
I-PER I-PER X     O   O O      X
```

由于单词块序列化边界是输入中已知的一部分，因此对训练和测试都要这样做。
结果如表 3 所示。$BERT_{LARGE}$ 优于现存的最优模型，使用多任务学习的交叉视野训练 (Clark et al., 2018)，CoNLL-2003 命名实体识别测试集上高 0.2 F1 值。

System	Dev F1	Test F1
ELMo+BiLSTM+CRF	95.7	92.2
CVT+Multi (Clark et al., 2018)	-	92.6
$\text{BERT}_{\text{BASE}}$	96.4	92.4
$\text{BERT}_{\text{LARGE}}$	**96.6**	**92.8**

Table 3: CoNLL-2003 Named Entity Recognition results. The hyperparameters were selected using the Dev set, and the reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

> *表 3：CoNLL-2003 命名实体识别。模型超参数使用验证集进行选择，报告的验证集和测试分数使用这些超参数进行随机五次以上的实验然后取实验的平均结果。*

4.4 SWAG

Adversarial Generations（SWAG）数据集由 113k 个句子对组合而成，用于评估基于常识的推理 (Zellers et al., 2018)。

给出一个来自视频字幕数据集的句子，任务是在四个选项中选择最合理的延续。例如：

```
A girl is going across a set of monkey bars.  She
(i) jumps up across the monkey bars.
(ii) struggles onto the bars to grab her head.
(iii) gets to the end and stands on a wooden plank.
(iv) jumps up and does a back flip.
```

为 SWAG 数据集调整 BERT 模型的方式与为 GLUE 数据集调整的方式相似。对于每个例子，我们构造四个输入序列，每一个都连接给定的句子（句子A）和一个可能的延续（句子B）。唯一的特定于任务的参数是我们引入向量 $V \in \mathbb{R}^H$，然后它点乘最后层的句子总表示 $C_i \in \mathbb{R}^H$ 为每一个选择 i 产生一个分数。概率分布为 softmax 这四个选择:

$$P_i = \frac{e^{V \cdot C_i}}{\sum_j^4 e^{S \cdot C_j}}$$

我们使用 $2e-5$ 的学习率，16 的批次大小训练模型 3 个周期。结果如表 4 所示。$BERT_{LARGE}$ 优于作者的 ESIM+ELMo 的基线标准模型的 27.1% 。

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

Table 4: SWAG Dev and Test accuracies. Test results were scored against the hidden labels by the SWAG authors. †Human performance is measure with 100 samples, as reported in the SWAG paper.

> *表 4：SWAG 验证集和测试集准确率。测试结果由 SWAG 作者对隐藏的标签进行评分。人类的表现是用 100 个样本来衡量的，正如 SWAG 论文中描述的那样。*

5. 消融研究（Ablation Studies）

虽然我们已经证明了非常强有力的实证结果，但到目前为止提出的结果并没有提现出 BERT 框架的每个部分具体的贡献。在本节中，我们对 BERT 的许多方面进行了消融实验，以便更好地理解每个部分的相对重要性。

5.1 预训练任务的影响

我们的核心观点之一是，与之前的工作相比，BERT 的深层双向性（通过遮蔽语言模型预训练）是最重要的改进。为了证明这一观点，我们评估了两个新模型，它们使用与 $BERT_{BASE}$ 完全相同的预训练数据、微调方案和 Transformer 超参数：

1. No NSP：模型使用“遮蔽语言模型”（MLM）但是没有“预测下一句任务”（NSP）。
2. LTR & No NSP：模型使用一个从左到右（LTR）的语言模型，而不是遮蔽语言模型。在这种情况下，我们预测每个输入词，不应用任何遮蔽。在微调中也应用了仅限左的约束，因为我们发现使用仅限左的上下文进行预训练和使用双向上下文进行微调总是比较糟糕。此外，该模型未经预测下一句任务的预训练。这与OpenAI GPT有直接的可比性，但是使用更大的训练数据集、输入表示和微调方案。

结果如表 5 所示。我们首先分析了 NSP 任务所带来的影响。我们可以看到去除 NSP 对 QNLI、MNLI 和 SQuAD 的表现造成了显著的伤害。这些结果表明，我们的预训练方法对于获得先前提出的强有力的实证结果是至关重要的。

接着我们通过对比 “No NSP” 与 “LTR & No NSP” 来评估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 上的下降特别大。对于SQuAD来说，很明显 LTR 模型在区间和标记预测方面表现很差，因为标记级别的隐藏状态没有右侧上下文。因为 MRPC 不清楚性能差是由于小的数据大小还是任务的性质，但是我们发现这种性能差是在一个完全超参数扫描和许多次随机重启之间保持一致的。

为了增强 LTR 系统，我们尝试在其上添加一个随机初始化的 BiLSTM 来进行微调。这确实大大提高了 SQuAD 的成绩，但是结果仍然比预训练的双向模型表现差得多。它还会损害所有四个 GLUE 任务的性能。

我们注意到，也可以培训单独的 LTR 和 RTL 模型，并将每个标记表示为两个模型表示的连接，就像 ELMo 所做的那样。但是：（a）这是单个双向模型参数的两倍大小；（b）这对于像 QA 这样的任务来说是不直观的，因为 RTL 模型无法以问题为条件确定答案；（c）这比深层双向模型的功能要弱得多，因为深层双向模型可以选择使用左上下文或右上下文。

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: Ablation over the pre-training tasks using the $\text{BERT}_{\text{BASE}}$ architecture. "No NSP" is trained without the next sentence prediction task. "LTR & No NSP" is trained as a left-to-right LM without the next sentence prediction, like OpenAI GPT. "+ BiLSTM" adds a randomly initialized BiLSTM on top of the "LTR + No NSP" model during fine-tuning.

> *表 5：在预训练任务中使用 $BERT_{BASE}$ 模型进行消融实验。"No NSP"表示不进行下一句预测任务来训练模型。"LTR & No NSP"表示就像 OpenAI GPT 一样，使用从左到右的语言模型不进行下一句预测任务来训练模型。"+ BiLSTM"表示在"LTR & No NSP"模型微调时添加一个随机初始化的 BiLSTM 层。*

5.2 模型大小的影响

在本节中，我们将探讨模型大小对微调任务准确度的影响。我们用不同的层数、隐藏单位和注意力头个数训练了许多 BERT 模型，同时使用了与前面描述的相同的超参数和训练过程。

选定 GLUE 任务的结果如表 6 所示。在这个表中，我们报告了 5 次在验证集上的微调的随机重启的平均模型准确度。我们可以看到，更大的模型在所选 4 个数据集上都带来了明显的准确率上升，甚至对于只有 3600 个训练数据的 MRPC 来说也是如此，并且与预训练任务有很大的不同。也许令人惊讶的是，相对于现有文献，我们能够在现有的模型基础上实现如此显著的改进。例如，Vaswani et al.(2017) 研究的最大 Transformer 为(L=6, H=1024, A=16)，编码器参数为 100M，我们所知的文献中的最大 Transformer 为(L=64, H=512, A=2)，参数为235M（Al-Rfou et al., 2018）。相比之下，$BERT_{BASE}$ 含有 110M 参数而

$BERT_{LARGE}$ 含有 340M 参数。

多年来人们都知道，增加模型的大小将持续提升在大型任务(如机器转换和语言建模)上的的表现，表 6 所示的由留存训练数据（held-out traing data）计算的语言模型的困惑度（perplexity）。然而，我们相信，这是第一次证明，如果模型得到了足够的预训练，那么将模型扩展到极端的规模也可以在非常小的任务中带来巨大的改进。

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

Table 6: Ablation over BERT model size. #L = the number of layers; #H = hidden size; #A = number of attention heads. "LM (ppl)" is the masked LM perplexity of held-out training data.

> *表 6：调整 BERT 的模型大小。#L = 层数；#H = 隐藏维度大小；#A = 注意力头的个数。"LM (ppl)"表示遮蔽语言模型在预留训练数据上的困惑度。*

5.3 训练步数的影响

图 4 显示了经过 K 步预训练模型的检查点再模型微调之后在 MNLI 验证集上的准确率。这让我们能够回答下列问题:

1. 问:BERT真的需要这么多的预训练 (128,000 words/batch * 1,000,000 steps) 来实现高的微调精度吗?
   答：是的，$BERT_{BASE}$ 在 MNLI 上进行 1M 步预训练时的准确率比 500k 步提高了近 1.0%。
2. 问:遮蔽语言模型的预训练是否比 LTR 模型预训练训收敛得慢，因为每批只预测 15% 的单词，而不是每个单词?
   答：遮蔽语言模型的收敛速度确实比 LTR 模型稍慢。然而，在绝对准确性方面，遮蔽语言模型几乎在训练一开始就超越 LTR 模型。

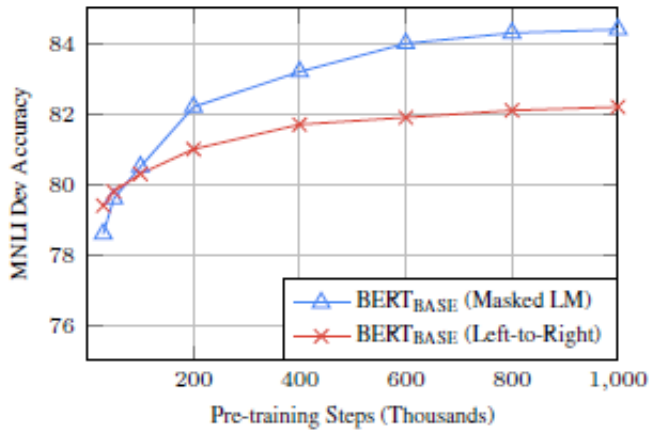


Figure 4: Ablation over number of training steps. This shows the MNLI accuracy after fine-tuning, starting from model parameters that have been pre-trained for k steps. The x-axis is the value of k.

> *图 4：调整模型的训练步数。图中展示了已经预训练了 k 步后的模型参数在 MNLI 数据集上的再经过微调后的准确率。x 轴的值就是 k。*

5.4 使用 BERT 基于特征的方法

到目前为止，所有的 BERT 结果都使用了微调方法，将一个简单的分类层添加到预训练的模型中，并在一个下行任务中对所有参数进行联合微调。然而，基于特征的方法，即从预训练模型中提取固定的特征，具有一定的优势。首先，并不是所有 NLP 任务都可以通过 Transformer 编码器体系结构轻松地表示，因此需要添加特定于任务的模型体系结构。其次，能够一次性耗费大量计算预先计算训练数据的表示，然后在这种表示的基础上使用更节省计算的模型进行许多实验，这有很大的计算优势。

在本节中，我们通过在 CoNLL-2003 命名实体识别任务上生成类似于 elmo 的预训练的上下文表示来评估基于特征的方法中的 BERT 表现有多好。为此，我们使用与第 4.3 节相同的输入表示，但是使用来自一个或多个层的激活输出，而不需要对BERT的任何参数进行微调。在分类层之前，这些上下文嵌入被用作对一个初始化的两层 768 维 Bi-LSTM 的输入。

结果如表 7 所示。最佳的执行方法是从预训练的转换器的前 4 个隐藏层串联符号表示，这只比整个模型的微调落后 0.3 F1 值。这说明 BERT 对于微调和基于特征的方法都是有效的。

Layers	Dev F1
Finetune All	96.4
First Layer (Embeddings)	91.0
Second-to-Last Hidden	95.6
Last Hidden	94.9
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1
Sum All 12 Layers	95.5

Table 7: Ablation using BERT with a feature-based approach on CoNLL-2003 NER. The activations from the specified layers are combined and fed into a two-layer BiLSTM, without backpropagation to BERT.

> *表 7：在 CoNLL-2003 命名实体识别上使用基于特征的方法，并调整 BERT 层数。来自指定层的激活输出被组合起来，并被送到一个两层的 BiLSTM 中，而不需要反向传播到 BERT。*

6. 结论

最近，由于使用语言模型进行迁移学习而取得的实验提升表明，丰富的、无监督的预训练是许多语言理解系统不可或缺的组成部分。特别是，这些结果使得即使是低资源（少量标签的数据集）的任务也能从非常深的单向结构模型中受益。我们的主要贡献是将这些发现进一步推广到深层的双向结构，使同样的预训练模型能够成功地广泛地处理 NLP 任务。

虽然这些实证结果很有说服力，在某些情况下甚至超过了人类的表现，但未来重要的工作是研究 BERT 可能捕捉到的或不捕捉到的语言现象。

二、BERT实践：SMP2020-EWECT微博情绪分类

在SMP2020的微博情绪分类任务上，微调在中文预料上预训练的BERT模型，进行文本分类。

1、数据集简介

来源于微博情绪分析评测(SMP2020-EWECT)，本届微博情绪分类评测任务一共包含两个数据集：

- 通用微博数据集，其中的微博是随机收集的包含各种话题的数据
- 疫情微博数据集，其中的微博数据均与本次疫情相关。

微博情绪分类任务旨在识别微博中蕴含的情绪，输入是一条微博，输出是该微博所蕴含的情绪类别。在本次评测中，我们将微博按照其蕴含的情绪分为以下六个类别之一：积极、愤怒、悲伤、恐惧、惊奇和无情绪。

情绪	通用微博数据集	疫情微博数据集
积极（happy）	哥，你猜猜看和喜欢的人一起做公益是什么感觉呢。我们的项目已经进入一个新阶段了，现在特别有成就感。加油加油。	愿大家平安、健康[心]#致敬疫情前线医护人员# 愿大家都健康平安
愤怒(angry)	每个月都有特别气愤的时候。，多少个瞬间想甩手不干了，杂七杂八，当我是什么。	整天歌颂医护人员伟大的自我牺牲精神，人家原本不用牺牲好吧！吃野味和隐瞒疫情的估计是同一波人，真的要死自己去死，别拉上无辜的人。
悲伤(sad)	回忆起老爸的点点滴滴，心痛...为什么.接受不了	救救武汉吧，受不了了泪奔，一群孩子穿上大人衣服学着救人 请官方不要瞒报谎报耽误病情，求求武汉zf了[泪][泪][泪][泪]
恐惧(fear)	明明是一篇言情小说，看完之后为什么会恐怖的睡不着呢，越想越害怕[吃驚]	对着这个症状，没病的都害怕[允悲][允悲]
惊奇(surprise)	我竟然不知道kkw是丑女无敌里的那个	我特别震惊就是真的很多人上了厕所是不会洗手的。。。。
无情绪(neutral)	我们做不到选择缘分，却可以珍惜缘分。	辟谣，盐水漱口没用。

2、数据预处理

本次训练中只使用了**通用微博数据集**，针对数据集的规模、类别分布、数据质量进行分析、处理。

2.1、通用数据集大小

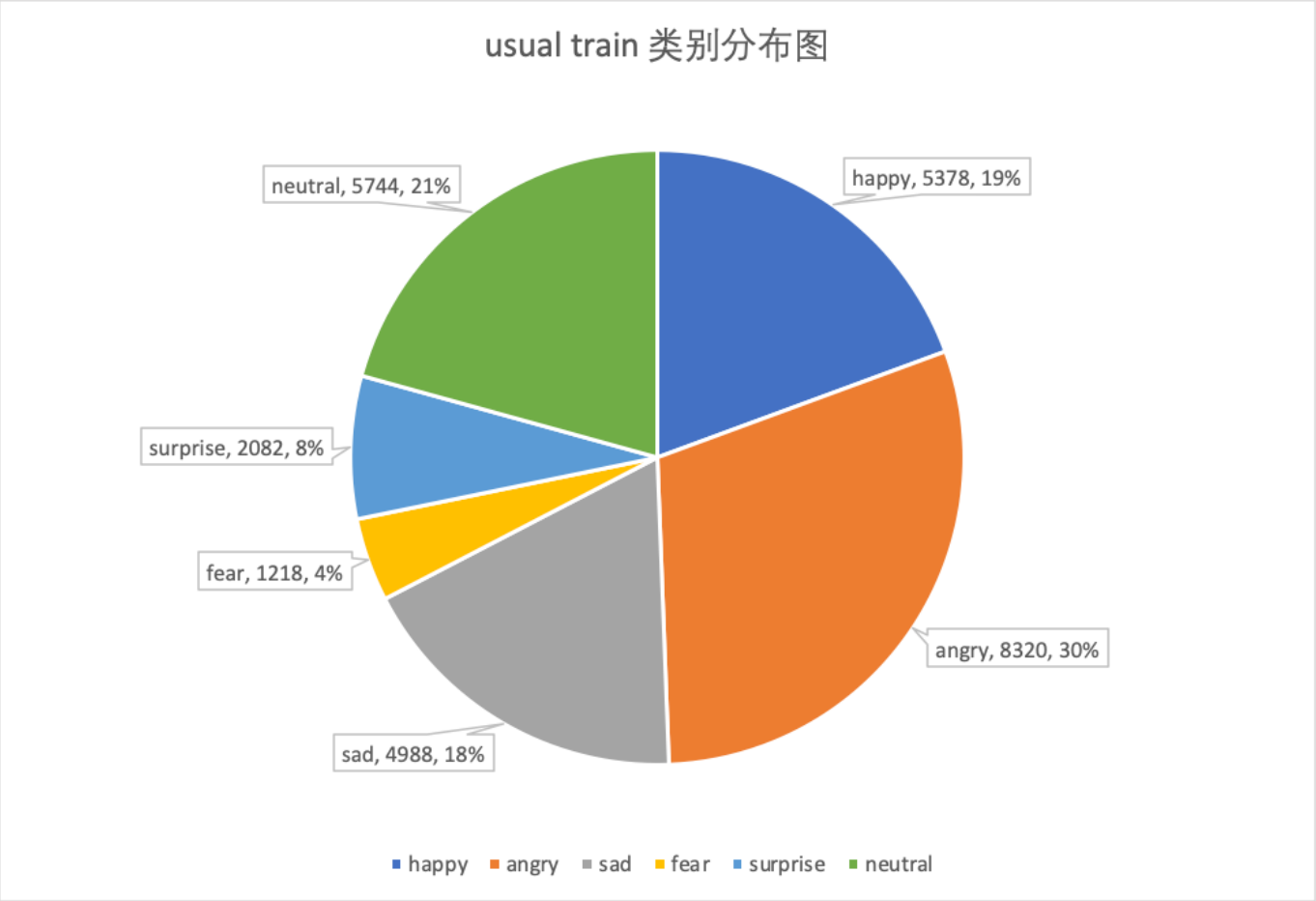
通用数据集训练集、验证集、测试集数据集大小:

	TRAIN	VAL	TEST
通用数据集	27768	2000	5000

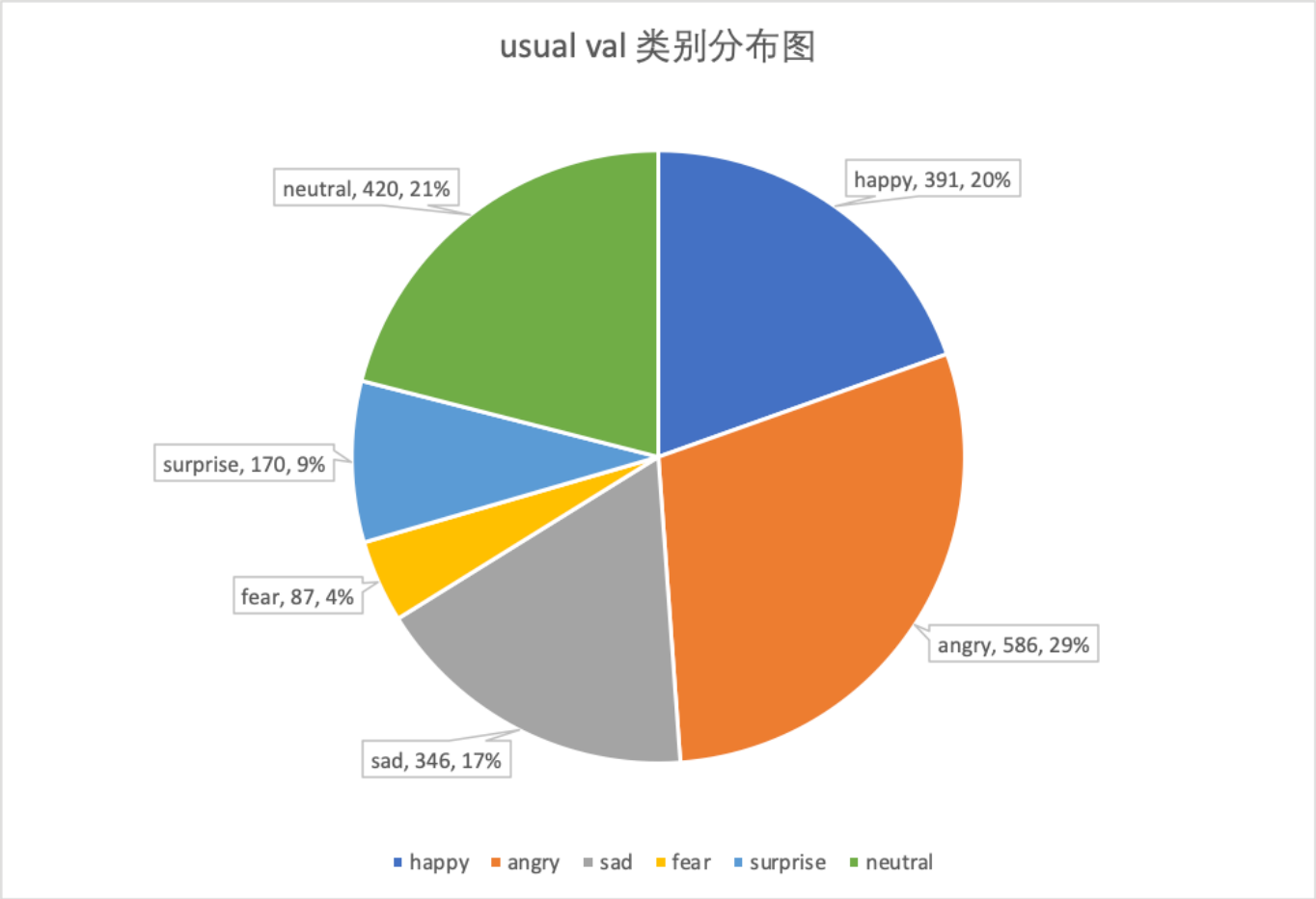
> *VAL 只有 Train的7%，在训练时应该提高val的大小，防止验证结果有偏。*

2.2、通用数据集类别分布

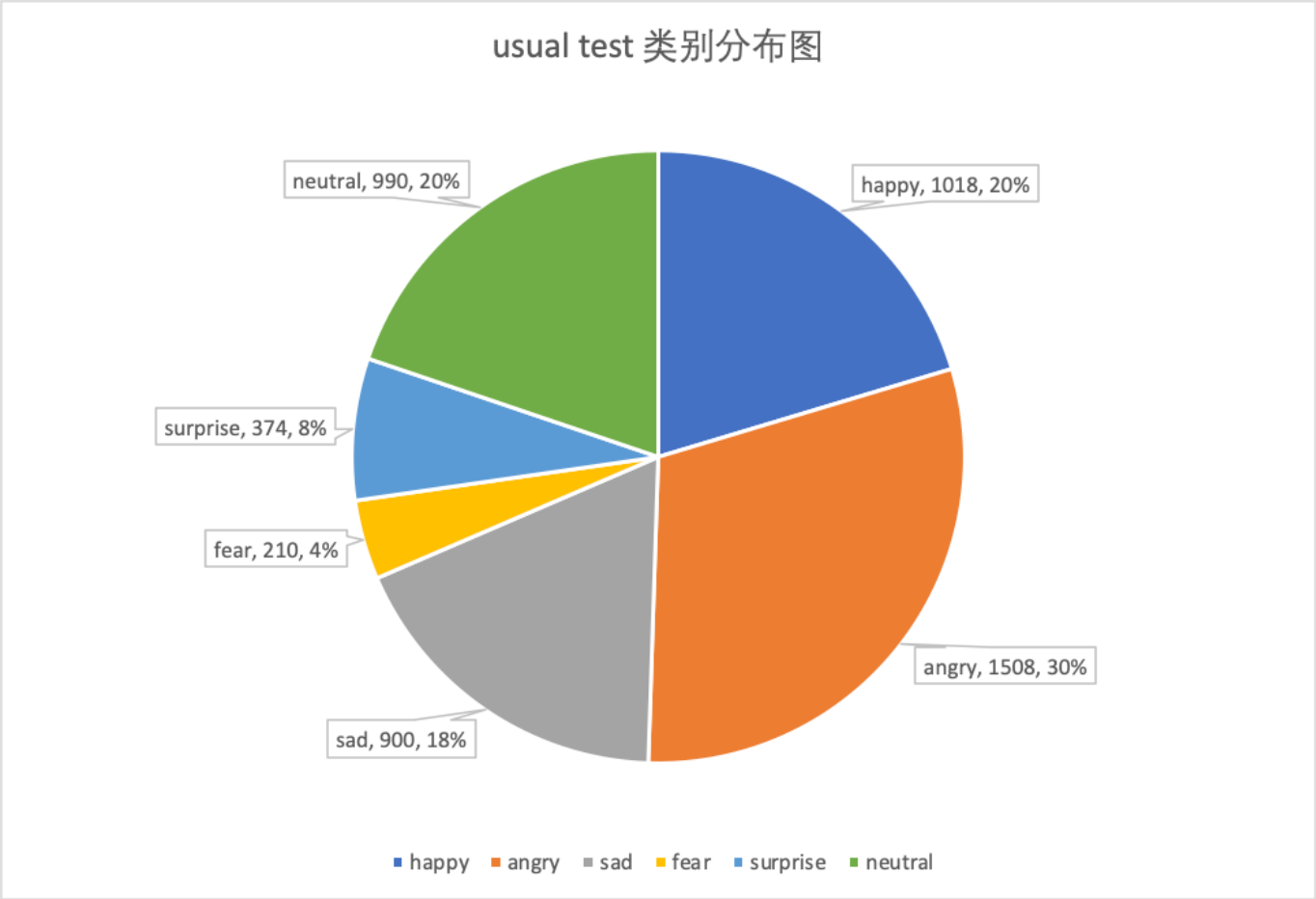
> *usual train*



> *usual val*


usual val 类别分布图
happy, 391, 20%
angry, 586, 29%
sad, 346, 17%
fear, 87, 4%
surprise, 170, 9%
neutral, 420, 21%
happy
angry
sad
fear
surprise
neutral

> *usual test*



训练集、验证集、测试集的类别比例基本一致，但是存在明显的样本不均衡，fear类仅有4%。

2.3、通用数据集清洗

进行了全角转半角、繁转简、英文大写转小写、去除url、去除email、去除@以及保留emoji等操作，下表展示了部分清洗数据，在模型处理中，我们限制数据的最大长度为140。

> *参考代码：https://github.com/thinkingmanyangyang/smp-ewect-code/blob/master/clean_data.py*

清洗策略	清洗前	清洗后
繁简体转换	昨個回髮廊洗頭結果設計師竟然想吹直我的頭髮！嘿小姐妳幫我用卷的竟敢吹直它妳可以對我沒印象但妳總該看一下吧！嗯嗯我不會再去了！	昨个回发廊洗头结果设计师竟然想吹直我的头发!嘿小姐你帮我用卷的竟敢吹直它你可以对我没印象但你总该看一下吧!嗯嗯我不会再去了!
微博@标签	每次遇见这样，我都想下去手撕这样违法吗@陈震同学	每次遇见这样,我都想下去手撕这样违法吗
URL	对于结婚我可能真低调到了，生完娃仍然有许多人发来☞你什么时候结婚的？疑问。对于辣妈，我真得战战兢兢忐忐忑忑，辣容易，当妈好难。极强的动手能力与时间分配能力，杂物整理能力，好耐力等等的考验。说实话我前所未有的觉得自己需要莫大的支持和身边的帮助。可。哎。。。http://t.cn/R2Wx9I3	对于结婚我可能真低调到了,生完娃仍然有许多人发来☞你什么时候结婚的?疑问。对于辣妈,我真得战战兢兢忐忐忑忑,辣容易,当妈好难。极强的动手能力与时间分配能力,杂物整理能力,好耐力等等的考验。说实话我前所未有的觉得自己需要莫大的支持和身边的帮助。可。哎。。。

3、模型训练

使用Bert在中文数据集上的预训练后的模型，进行fine-tune。

> *参考仓库：https://github.com/huggingface/transformers*
>
> *Transformers 提供了数以千计的预训练模型，支持 100 多种语言的文本分类、信息抽取、问答、摘要、翻译、文本生成。它的宗旨让最先进的 NLP 技术人人易用。*

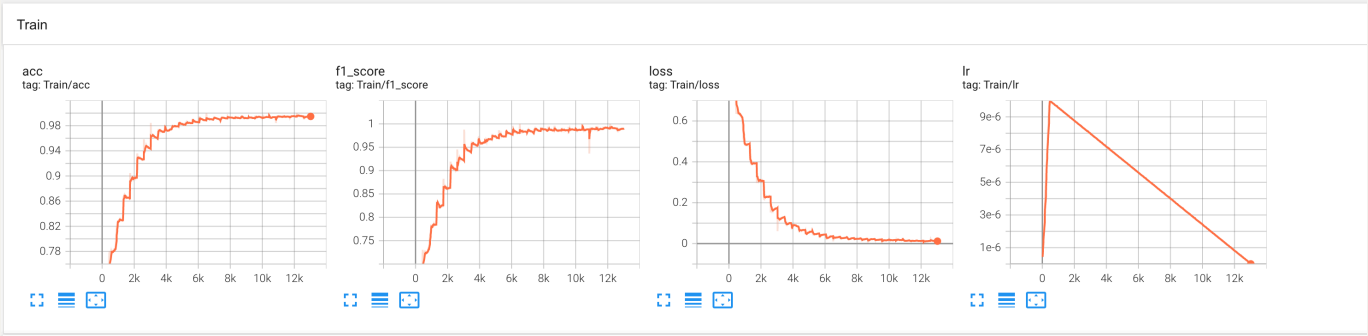
训练参数如下：

参数	设置
显卡	GTX1080Ti with 12G
模型名称	bert-base-chinese
batch_size	64
混合精度训练	amp True
学习率	1e-5
Warm up	1 epoch
训练轮数	30
优化器	AdamW
序列的最大长度	140

3.1、train

Tensorboard的监控，包括：

- accuracy
- f1 score(macro)
- train loss
- lr：在第一个epoch使用warm up

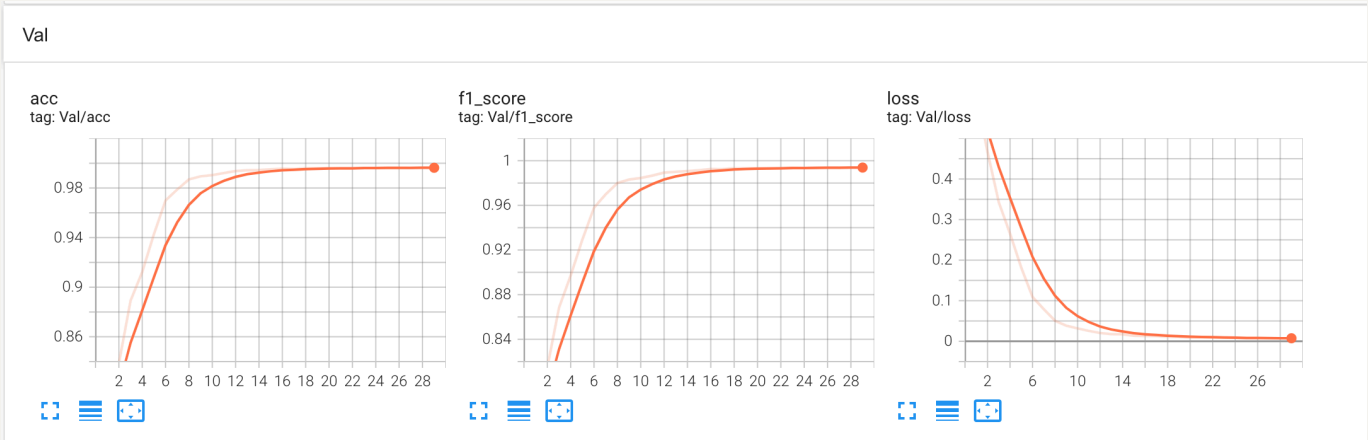


> *随着训练的进行，train的acc、f1_score都是稳步提升，loss稳步下降，收敛。（横坐标为迭代步数）*

3.2、validation

Tensorboard的监控，包括：

- accuracy
- f1 score(macro)
- val loss



> *每个epoch训练完成都在验证集上进行验证，acc、f1_score、loss和训练集上的趋势基本一致。（横坐标为训练轮数）*

4、模型测试

在测试集上评测模型：

- accuracy：0.772
- f1 score(macro)：0.739

最终排名	队伍名称	机构名称	最终指标	各指标得分及排名				提交日期
				通用微博Macro_F	通用微博Accuracy	疫情微博Macro_F	疫情微博Accuracy	
1	Tencent	Tencent Oteam	0.7467	0.7856 / 1	0.8076 / 2	0.7078 / 1	0.8247 / 1	2020-08-10 23:49
2	清博大数据	北京清博大数据科技有限公司	0.7393	0.7823 / 3	0.8076 / 1	0.6964 / 2	0.8100 / 4	2020-08-09 13:43
3	拿第一导师请吃肯德基	东南大学	0.7360	0.7788 / 7	0.8014 / 8	0.6932 / 4	0.8120 / 3	2020-08-10 20:15
4	BERT 4EVER	广东外语外贸大学	0.7346	0.7756 / 8	0.8018 / 7	0.6936 / 3	0.8097 / 6	2020-08-09 19:08
5	sys1874	大连理工大学	0.7337	0.7815 / 4	0.8054 / 4	0.6859 / 5	0.8140 / 2	2020-08-10 17:11
6	炬火	山西大学	0.7314	0.7832 / 2	0.8056 / 3	0.6796 / 10	0.8080 / 8	2020-08-10 23:03
7	马家沟园林中心	东北林业大学	0.7298	0.7795 / 6	0.8028 / 6	0.6800 / 9	0.8090 / 7	2020-08-10 10:25
8	scau_EWECT	scau_SIGSDS	0.7295	0.7805 / 5	0.8044 / 5	0.6785 / 12	0.8050 / 11	2020-08-09 23:43
9	thuir	清华大学	0.7272	0.7732 / 10	0.7934 / 13	0.6813 / 6	0.8097 / 5	2020-08-09 09:53
10	我就看看，不拿奖	苏州大学&黑龙江大学	0.7263	0.7734 / 9	0.7970 / 9	0.6792 / 11	0.8063 / 10	2020-08-10 23:06
11	好未来	好未来	0.7235	0.7660 / 13	0.7944 / 12	0.6810 / 7	0.8077 / 9	2020-08-10 18:01
12	DUT93814	大连理工大学	0.7225	0.7699 / 11	0.7962 / 10	0.6750 / 14	0.8000 / 16	2020-08-10 10:08
13	名字太难听了	BLCU	0.7209	0.7617 / 16	0.7890 / 14	0.6802 / 8	0.8047 / 12	2020-08-10 01:44
14	NGU	天津科技大学	0.7206	0.7686 / 12	0.7948 / 11	0.6727 / 15	0.8013 / 13	2020-08-10 21:33
15	加油2020	南京理工大学NUSTM	0.7167	0.7648 / 14	0.7888 / 15	0.6685 / 18	0.8007 / 14	2020-08-10 23:08
16	hawkeye	上海阅文信息技术有限公司	0.7161	0.7632 / 15	0.7858 / 16	0.6690 / 17	0.7993 / 19	2020-08-10 22:36
17	THU_NGN	清华大学下一代网络及应用实验室	0.7139	0.7566 / 19	0.7846 / 18	0.6712 / 16	0.7997 / 17	2020-08-09 21:46
18	bb_dog	DLUT	0.7084	0.7587 / 17	0.7852 / 17	0.6580 / 20	0.7993 / 18	2020-08-10 21:55
19	没有情感的debug	东南大学	0.7045	0.7529 / 20	0.7738 / 23	0.6562 / 21	0.7940 / 21	2020-08-10 22:31
20	小熊维尼很可爱	小米	0.7020	0.7360 / 26	0.7684 / 25	0.6679 / 19	0.7960 / 20	2020-08-10 11:11
21	TF@1409	复旦大学大数据学院	0.7011	0.7480 / 22	0.7764 / 20	0.6543 / 22	0.7820 / 22	2020-08-10 23:12
22	upsidedown	国网信通产业集团福建亿榕信息技术有限公司	0.6972	0.7568 / 18	0.7842 / 19	0.6377 / 25	0.7773 / 24	2020-08-10 03:55

> *以上是本次比赛最后的排名。在仅仅使用单个BERT模型，不使用：*
>
> - *疫情数据集*
> - *RoBerta等改进的中文预训练模型*
> - *多模型融合*
>
> *也取得了有一定竞争力的效果。*

三、总结

BERT采用了Transformer Encoder的模型来作为语言模型， 完全抛弃了RNN/CNN等结构，而完全采用Attention机制来进行input-output之间关系的计算，在无监督预料上进行超大规模的预训练，能够有效提升下游语言任务的准确性。本文在BERT的文本分类任务上进行实践，数据集来源于**SMP2020-EWECT**微博情绪分类的通用数据集，在没有使用任何trick的情况下，在中文预训练的BERT上进行微调，单模型取得了77.2%的准确率。

四、相关代码

```
git clone https://github.com/BrownSweater/BERT_SMP2020-EWECT.git
```

1、附件

> ├── *clean_data.py*　　*## 数据预处理代码*
> ├── ***data/***　　*## 数据集*
> │　├── ***clean/***　　*## 原始数据集*
> │　└── ***raw/***　　*## 清洗后数据集*
> ├── *events.out.tfevents.1640466123.wjj.9864.0　## tensorboard结果文件*
> ├── *finetune.py*　　*## 训练代码*
> ├── *inference.py*　*## 推理代码*
> ├── *requirements.txt*　*## 依赖包*
> ├── *test.py*　　*## 测试代码*
> ├── *test.sh*　　*## 测试脚本*
> └── *train.sh*　　*## 训练脚本*

> *模型文件：https://pan.baidu.com/s/1e6Cs9STYmcdCWv6j3bjKQg　2ddr*

2、训练

- 安装依赖包：`pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple some-package`
- 开始训练：`sh train.sh`，在workspace/wb/目录下会保存相关训练的文件

3、测试

- 下载提供的模型文件 or 自己训练得到的模型，放在workspace/wb/目录下
- 开始测试：`sh test.sh`

最终排名	队伍名称	机构名称	最终指标	各指标得分及排名				提交日期
				通用微博Macro_F	通用微博Accuracy	疫情微博Macro_F	疫情微博Accuracy	
1	Tencent	Tencent Oteam	0.7467	0.7856 / 1	0.8076 / 2	0.7078 / 1	0.8247 / 1	2020-08-10 23:49
2	清博大数据	北京清博大数据科技有限公司	0.7393	0.7823 / 3	0.8076 / 1	0.6964 / 2	0.8100 / 4	2020-08-09 13:43
3	拿第一导师请吃肯德基	东南大学	0.7360	0.7788 / 7	0.8014 / 8	0.6932 / 4	0.8120 / 3	2020-08-10 20:15
4	BERT 4EVER	广东外语外贸大学	0.7346	0.7756 / 8	0.8018 / 7	0.6936 / 3	0.8097 / 6	2020-08-09 19:08
5	sys1874	大连理工大学	0.7337	0.7815 / 4	0.8054 / 4	0.6859 / 5	0.8140 / 2	2020-08-10 17:11
6	炬火	山西大学	0.7314	0.7832 / 2	0.8056 / 3	0.6796 / 10	0.8080 / 8	2020-08-10 23:03
7	马家沟园林中心	东北林业大学	0.7298	0.7795 / 6	0.8028 / 6	0.6800 / 9	0.8090 / 7	2020-08-10 10:25
8	scau_EWECT	scau_SIGSDS	0.7295	0.7805 / 5	0.8044 / 5	0.6785 / 12	0.8050 / 11	2020-08-09 23:43
9	thuir	清华大学	0.7272	0.7732 / 10	0.7934 / 13	0.6813 / 6	0.8097 / 5	2020-08-09 09:53
10	我就看看，不拿奖	苏州大学&黑龙江大学	0.7263	0.7734 / 9	0.7970 / 9	0.6792 / 11	0.8063 / 10	2020-08-10 23:06
11	好未来	好未来	0.7235	0.7660 / 13	0.7944 / 12	0.6810 / 7	0.8077 / 9	2020-08-10 18:01
12	DUT93814	大连理工大学	0.7225	0.7699 / 11	0.7962 / 10	0.6750 / 14	0.8000 / 16	2020-08-10 10:08
13	名字太难听了	BLCU	0.7209	0.7617 / 16	0.7890 / 14	0.6802 / 8	0.8047 / 12	2020-08-10 01:44
14	NGU	天津科技大学	0.7206	0.7686 / 12	0.7948 / 11	0.6727 / 15	0.8013 / 13	2020-08-10 21:33
15	加油2020	南京理工大学NUSTM	0.7167	0.7648 / 14	0.7888 / 15	0.6685 / 18	0.8007 / 14	2020-08-10 23:08
16	hawkeye	上海阅文信息技术有限公司	0.7161	0.7632 / 15	0.7858 / 16	0.6690 / 17	0.7993 / 19	2020-08-10 22:36
17	THU_NGN	清华大学下一代网络及应用实验室	0.7139	0.7566 / 19	0.7846 / 18	0.6712 / 16	0.7997 / 17	2020-08-09 21:46
18	bb_dog	DLUT	0.7084	0.7587 / 17	0.7852 / 17	0.6580 / 20	0.7993 / 18	2020-08-10 21:55
19	没有情感的debug	东南大学	0.7045	0.7529 / 20	0.7738 / 23	0.6562 / 21	0.7940 / 21	2020-08-10 22:31
20	小熊维尼很可爱	小米	0.7020	0.7360 / 26	0.7684 / 25	0.6679 / 19	0.7960 / 20	2020-08-10 11:11
21	TF@1409	复旦大学大数据学院	0.7011	0.7480 / 22	0.7764 / 20	0.6543 / 22	0.7820 / 22	2020-08-10 23:12
22	upsidedown	国网信通产业集团福建亿榕信息技术有限公司	0.6972	0.7568 / 18	0.7842 / 19	0.6377 / 25	0.7773 / 24	2020-08-10 03:55

> *以上是本次比赛最后的排名。在仅仅使用BERT模型，而不是使用*

4、推理

- 下载提供的模型文件 or 自己训练得到的模型，放在workspace/wb/目录下
- 执行代码：`python inference.py --input '你是个什么东西，垃圾' --device cpu`

> *执行结果：##################### result: angry #####################*

5、Web部署

- 下载提供的模型文件 or 自己训练得到的模型，放在workspace/wb/目录下
- 执行代码：`python3 web_demo.py --device cpu`