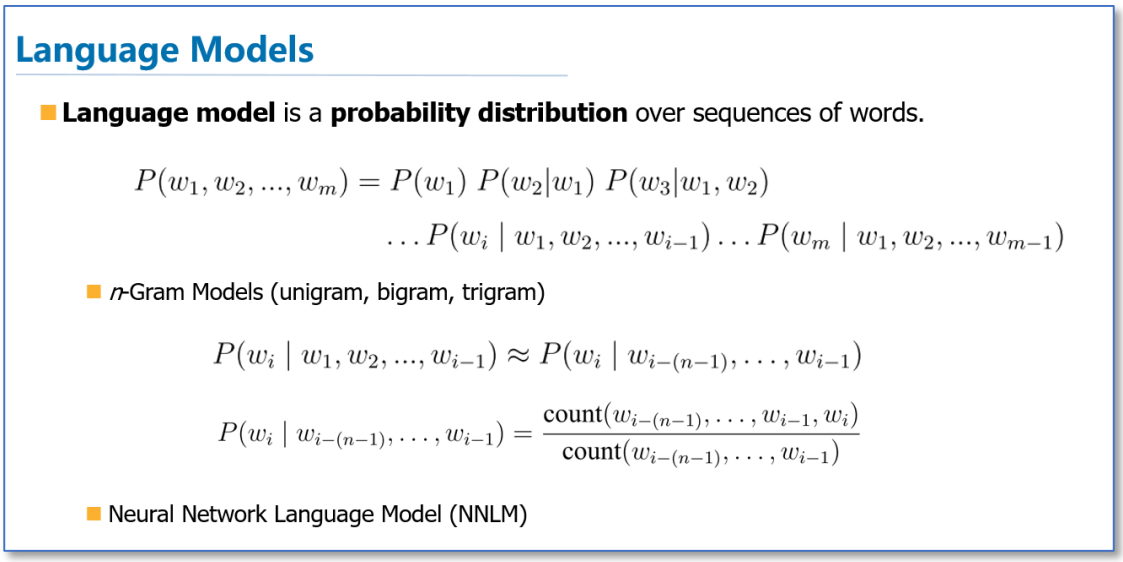
# 自然语言处理中的语言模型预训练方法

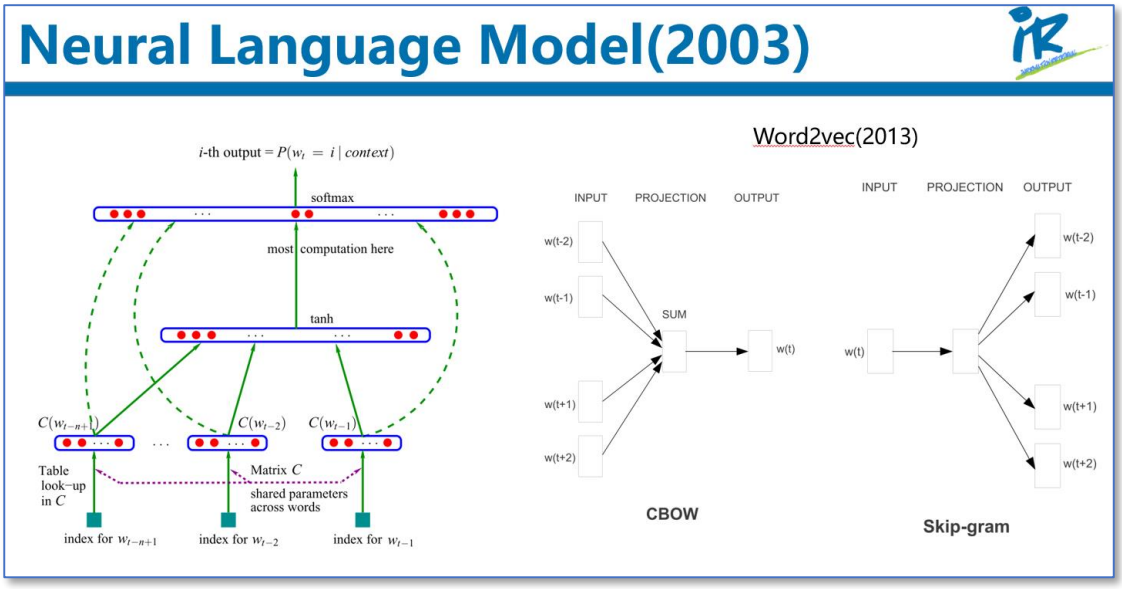
最近，在自然语言处理（NLP）领域中，使用语言模型预训练方法在多项 NLP 任务上都获得了不错的提升，广泛受到了各界的关注。就此，我将最近看的一些相关论文进行总结，选取了几个代表性模型（包括 ELMo [1]，OpenAI GPT [2]和 BERT [3]）和大家一起学习分享。

## 1. 引言

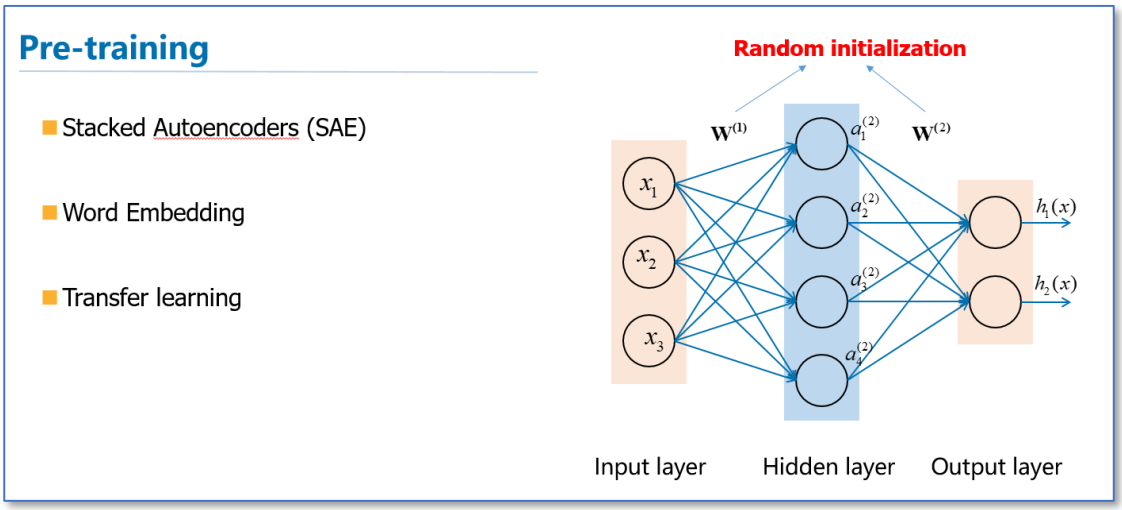
在介绍论文之前，我将先简单介绍一些相关背景知识。首先是语言模型（Language Model），语言模型简单来说就是一串词序列的概率分布。具体来说，语言模型的作用是为一个长度为 $m$ 的文本确定一个概率分布 P，表示这段文本存在的可能性。在实践中，如果文本的长度较长，$P(w_i \mid w_1, w_2, \ldots, w_{i-1})$的估算会非常困难。因此，研究者们提出使用一个简化模型：$n$ 元模型（n-gram model）。在 n 元模型中估算条件概率时，只需要对当前词的前 $n$ 个词进行计算。在 n 元模型中，传统的方法一般采用频率计数的比例来估算 $n$ 元条件概率。当 $n$ 较大时，机会存在数据稀疏问题，导致估算结果不准确。因此，一般在百万词级别的语料中，一般也就用到三元模型。




为了缓解 $n$ 元模型估算概率时遇到的数据稀疏问题，研究者们提出了神经网络语言模型。代表性工作是 Bengio 等人在 2003 年提出的神经网络语言模型，该语言模型使用了一个三层前馈神经网络来进行建模。其中有趣的发现了第一层参数，用做词表示不仅低维紧密，而且能够蕴涵语义，也就为现在大家都用的词向量（例如 word2vec）打下了基础。其实，语言模型就是根据上下文去预测下一个词是什么，这不需要人工标注语料，所以语言模型能够从无限制的大规模单语语料中，学习到丰富的语义知识。




接下来在简单介绍一下预训练的思想。我们知道目前神经网络在进行训练的时候基本都是基于后向传播（BP）算法，通过对网络模型参数进行随机初始化，然后通过 BP 算法利用例如 SGD 这样的优化算法去优化模型参数。那么预训练的思想就是，该模型的参数不再是随机初始化，而是先有一个任务进行训练得到一套模型参数，然后用这套参数对模型进行初始化，再进行训练。其实早期的使用自编码器栈式搭建深度神经网络就是这个思想。还有词向量也可以看成是第一层 word embedding 进行了预训练，此外在基于神经网络的迁移学习中也大量用到了这个思想。



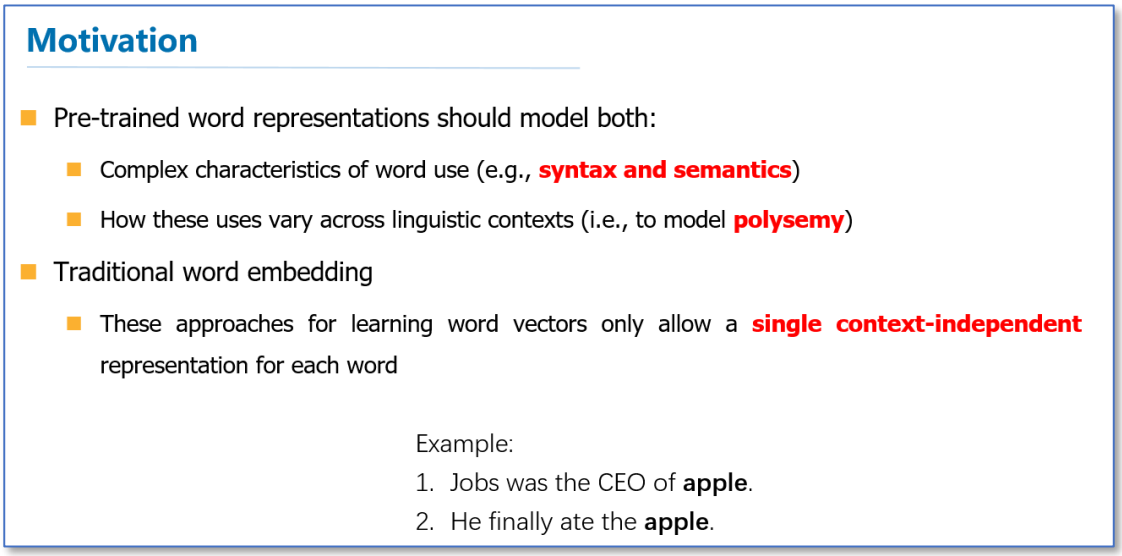

接下来，我们就具体看一下这几篇用语言模型进行预训练的工作。

## 2. ELMo

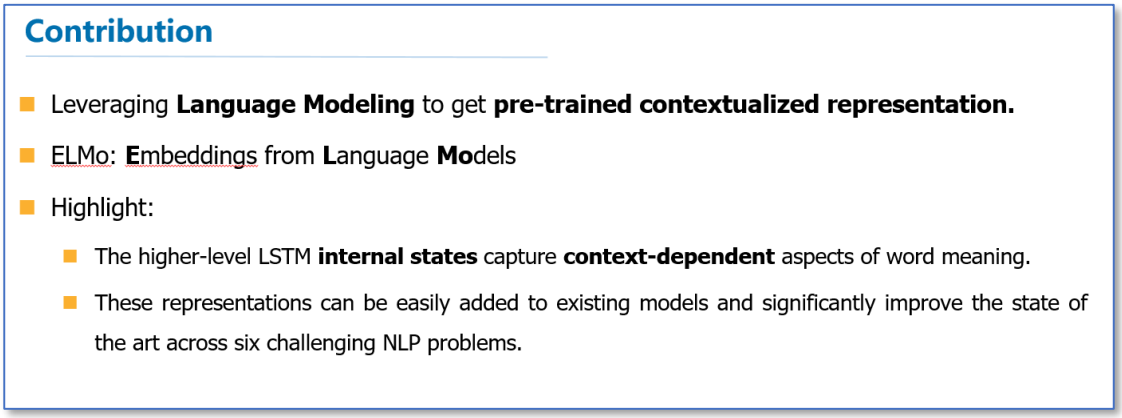
### 2.1 引言

《Deep Contextualized Word Representations》这篇论文来自华盛顿大学的工作，最后是

发表在今年的 NAACL 会议上，并获得了最佳论文。其实这个工作的前身来自同一团队在 ACL2017 发表的《Semi-supervised sequence tagging with bidirectional language models》 [4]，只是在这篇论文里，他们把模型更加通用化了。首先我们来看看他们工作的动机，他们认为一个预训练的词表示应该能够包含丰富的句法和语义信息，并且能够对多义词进行建模。而传统的词向量（例如 word2vec）是上下文无关的。例如下面“apple”的例子，这两个“apple”根据上下文意思是不同的，但是在 word2vec 中，只有 apple 一个词向量，无法对一词多义进行建模。




所以他们利用语言模型来获得一个上下文相关的预训练表示，称为 ELMo，并在 6 个 NLP 任务上获得了提升。




## 2.2 方法

在 EMLo 中，他们使用的是一个双向的 LSTM 语言模型，由一个前向和一个后向语言模型构成，目标函数就是取这两个方向语言模型的最大似然。

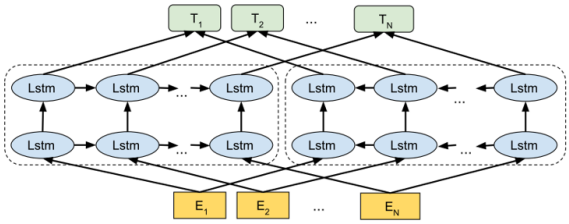
## Bidirectional language models

- A forward LM

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1})$$

- A backward LM

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N)$$




- Jointly maximize the log likelihood of the forward and backward directions

$$\sum_{k=1}^{N} ( \log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) )$$

在预训练好这个语言模型之后，ELMo 就是根据下面的公式来用作词表示，其实就是把这个双向语言模型的每一中间层进行一个求和。最简单的也可以使用最高层的表示来作为 ELMo。

## Embeddings from Language Models

- ELMo is a task specific combination of the **intermediate layer representations in the biLM**.
- For k-th token, L-layer bi-directional Language Models computes 2L+1 representations

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\}, \end{aligned}$$

- For a specific down-stream task, ELMo would learn a weight to combine these representations (In the simplest case, ELMo just selects the top layer $E(R_k) = \mathbf{h}_{k,L}^{LM}$ )

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

scalar parameter　softmax-normalized weights

14

然后在进行有监督的 NLP 任务时，可以将 ELMo 直接当做特征拼接到具体任务模型的词向量输入或者是模型的最高层表示上。总结一下，不像传统的词向量，每一个词只对应一个词向量，ELMo 利用预训练好的双向语言模型，然后根据具体输入从该语言模型中可以得到上下文依赖的当前词表示（对于不同上下文的同一个词的表示是不一样的），再当成特征加入到具体的 NLP 有监督模型里。

### 2.3 实验

这里我们简单看一下主要的实验，具体实验还需阅读论文。首先是整个模型效果的实验。他们在 6 个 NLP 任务上进行了实验，首先根据目前每个任务搭建了不同的模型作为 baseline，然后加入 ELMo，可以看到加入 ELMo 后 6 个任务都有所提升，平均大约能够提升 2 个多百分点，并且最后的结果都超过了之前的先进结果（SOTA）。

## SOTA

- Question answering (SQuAD), Textual entailment (SNLI), Semantic role labeling(SRL), Coreference resolution (Coref), Named entity extraction (NER), Sentiment analysis (SST-5)

| TASK | PREVIOUS SOTA | | OUR BASELINE | ELMO + BASELINE | INCREASE (ABSOLUTE/ RELATIVE) |
|---|---|---|---|---|---|
| SQuAD | Liu et al. (2017) | 84.4 | 81.1 | 85.8 | 4.7 / 24.9% |
| SNLI | Chen et al. (2017) | 88.6 | 88.0 | 88.7 ± 0.17 | 0.7 / 5.8% |
| SRL | He et al. (2017) | 81.7 | 81.4 | 84.6 | 3.2 / 17.2% |
| Coref | Lee et al. (2017) | 67.2 | 67.2 | 70.4 | 3.2 / 9.8% |
| NER | Peters et al. (2017) | 91.93 ± 0.19 | 90.15 | 92.22 ± 0.10 | 2.06 / 21% |
| SST-5 | McCann et al. (2017) | 53.7 | 51.4 | 54.7 ± 0.5 | 3.3 / 6.8% |

在下面的分析实验中，我们可以看到使用所有层的效果要比只使用最后一层作为 ELMo 的效果要好。在输入还是输出上面加 EMLo 效果好的问题上，并没有定论，不同的任务可能效果不一样。

## Analysis

**Alternate layer weighting schemes**

| Task | Baseline | Last Only | All layers λ=1 | All layers λ=0.001 |
|---|---|---|---|---|
| SQuAD | 80.8 | 84.7 | 85.0 | **85.2** |
| SNLI | 88.1 | 89.1 | 89.3 | **89.5** |
| SRL | 81.6 | 84.1 | 84.6 | **84.8** |

λ: regularize the ELMo weights to the loss. Large values such as λ = 1 effectively reduce the weighting function to a simple average over the layers, while smaller values (e.g., λ = 0.001) allow the layer weights to vary.

**Where to include ELMo?**

| Task | Input Only | Input & Output | Output Only |
|---|---|---|---|
| SQuAD | 85.1 | **85.6** | 84.8 |
| SNLI | 88.9 | **89.5** | 88.7 |
| SRL | **84.7** | 84.3 | 80.9 |

Including ELMo at both the input and output layers for SNLI and SQuAD improves over just the input layer, but for SRL performance is highest when it is included at just the input layer.

# 3. Open AI GPT

## 3.1 引言

我们来看看第二篇论文《Improving Language Understanding by Generative Pre-Training》，这是 OpenAI 团队前一段时间放出来的预印版论文。他们的目标是学习一个通用的表示，能够在大量任务上进行应用。这篇论文的亮点主要在于，他们利用了 Transformer 网络代替了 LSTM 作为语言模型来更好的捕获长距离语言结构。然后在进行具体任务有监督微调时使用了语言模型作为附属任务训练目标。最后再 12 个 NLP 任务上进行了实验，9 个任务获得了 SOTA。

## Contribution

- Their goal is to learn **a universal representation** that transfers with little adaptation to a wide range of tasks.
- Highlight:
  - Use **transformer networks** instead of LSTM to achieve better capture long-term linguistic structure.
  - Include **auxiliary training objectives** (e.g. language modeling) in addition to the task objective when fine-tuning.
  - Demonstrate the effectiveness of the approach on a wider range of tasks. (significantly improving upon the state of the art in **9 out of the 12 tasks** studied)

### 3.2 方法

首先我们来看一下他们无监督预训练时的语言模型。他们仍然使用的是标准的语言模型目标函数，即通过前 $k$ 个词预测当前词，但是在语言模型网络上他们使用了 google 团队在《Attention is all your need》论文中提出的 transformer 解码器作为语言模型。Transformer 模型主要是利用自注意力（self-attention）机制的模型，这里我就不多进行介绍，大家可以看论文或者参考我之前的博客（https://www.cnblogs.com/robert-dlut/p/8638283.html）。

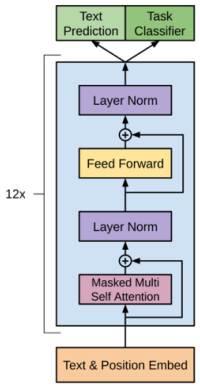
## Unsupervised pre-training

- Use a **standard language modeling objective** to maximize the following likelihood

$$L_1(\mathcal{U}) = \sum_i \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta)$$

- **A multi-layer Transformer decoder** for the language model

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$




然后再具体 NLP 任务有监督微调时，与 ELMo 当成特征的做法不同，OpenAI GPT 不需要再重新对任务构建新的模型结构，而是直接在 transformer 这个语言模型上的最后一层接上 softmax 作为任务输出层，然后再对这整个模型进行微调。他们额外发现，如果使用语言模型作为辅助任务，能够提升有监督模型的泛化能力，并且能够加速收敛。

## Supervised fine-tuning

- The final transformer block's activation is fed into an added linear output layer

$$P(y|x^1,\ldots,x^m) = \texttt{softmax}(h_l^m W_y)$$

- objective function

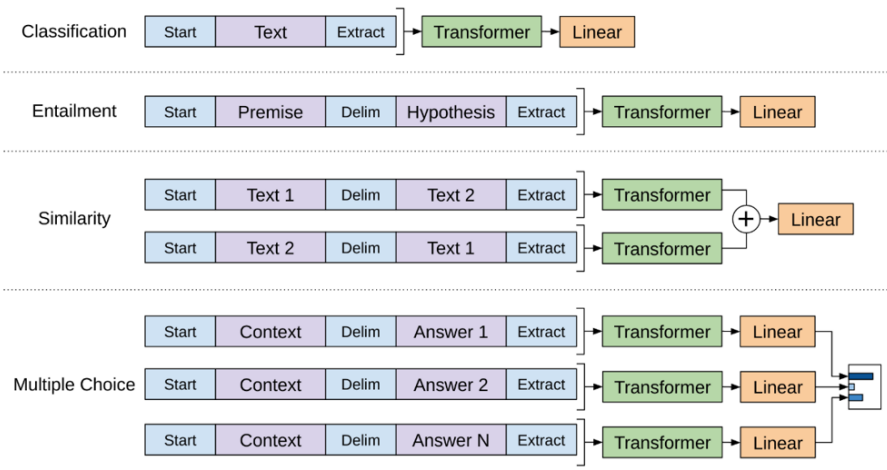
$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1,\ldots,x^m)$$

- We additionally found that including **language modeling as an auxiliary objective** to the fine-tuning helped learning by (a) improving generalization of the supervised model, and (b) accelerating convergence.

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

由于不同 NLP 任务的输入有所不同，在 transformer 模型的输入上针对不同 NLP 任务也有所不同。具体如下图，对于分类任务直接讲文本输入即可；对于文本蕴涵任务，需要将前提和假设用一个 Delim 分割向量拼接后进行输入；对于文本相似度任务，在两个方向上都使用 Delim 拼接后，进行输入；对于像问答多选择的任务，就是将每个答案和上下文进行拼接进行输入。

## Task-specific input transformations




### 3.3 实验

下面我简单的列举了一下不同 NLP 任务上的实验结果。

语言推理任务：

## Natural Language Inference

Table 2: Experimental results on natural language inference tasks, comparing our model with current state-of-the-art methods. 5x indicates an ensemble of 5 models. All datasets use accuracy as the evaluation metric.

| Method | MNLI-m | MNLI-mm | SNLI | SciTail | QNLI | RTE |
|---|---|---|---|---|---|---|
| ESIM + ELMo [44] (5x) | - | - | 89.3 | - | - | - |
| CAFE [58] (5x) | 80.2 | 79.0 | 89.3 | - | - | - |
| Stochastic Answer Network [35] (3x) | 80.6 | 80.1 | - | - | - | - |
| CAFE [58] | 78.7 | 77.9 | 88.5 | 83.3 | | |
| GenSen [64] | 71.4 | 71.3 | - | - | 82.3 | 59.2 |
| Multi-task BiLSTM + Attn [64] | 72.2 | 72.1 | - | - | 82.1 | **61.7** |
| Finetuned Transformer LM (ours) | **82.1** | **81.4** | **89.9** | **88.3** | **88.1** | 56.0 |

- RTE, one of the smaller datasets
- Given the strong performance of our approach on larger NLI datasets.

问答和常识推理任务：

## Question answering and commonsense reasoning

Table 3: Results on question answering and commonsense reasoning, comparing our model with current state-of-the-art methods.. 9x means an ensemble of 9 models.

| Method | Story Cloze | RACE-m | RACE-h | RACE |
|---|---|---|---|---|
| val-LS-skip [55] | 76.5 | - | - | - |
| Hidden Coherence Model [7] | 77.6 | - | - | - |
| Dynamic Fusion Net [67] (9x) | - | 55.6 | 49.4 | 51.2 |
| BiAttention MRU [59] (9x) | - | 60.2 | 50.3 | 53.3 |
| Finetuned Transformer LM (ours) | **86.5** | **62.9** | **57.4** | **59.0** |

This demonstrates the ability of our model to handle long-range contexts effectively.

语义相似度和分类任务：

## Semantic Similarity and Classification

Table 4: Semantic similarity and classification results, comparing our model with current state-of-the-art methods. All task evaluations in this table were done using the GLUE benchmark. (*mc*= Mathews correlation, *acc*=Accuracy, *pc*=Pearson correlation)

| Method | Classification | | Semantic Similarity | | | GLUE |
|---|---|---|---|---|---|---|
| | CoLA (mc) | SST2 (acc) | MRPC (F1) | STSB (pc) | QQP (F1) | |
| Sparse byte mLSTM [16] | - | **93.2** | - | - | - | - |
| TF-KLD [23] | - | - | **86.0** | - | - | - |
| ECNU (mixed ensemble) [60] | - | - | - | 81.0 | - | - |
| Single-task BiLSTM + ELMo + Attn [64] | 35.0 | 90.2 | 80.2 | 55.5 | 66.1 | 64.8 |
| Multi-task BiLSTM + ELMo + Attn [64] | 18.9 | 91.6 | 83.5 | 72.8 | 63.3 | 68.9 |
| Finetuned Transformer LM (ours) | **45.4** | 91.3 | 82.3 | **82.0** | **70.3** | **72.8** |

可以看到在多项任务上，OpenAI GPT 的效果要比 ELMo 的效果更好。从下面的消除实验来看，在去掉预训练部分后，所有任务都大幅下降，平均下降了 14.8%，说明预训练很有效；在大数据集上使用语言模型作为附加任务的效果更好，小数据集不然；利用 LSTM 代替 Transformer 后，结果平均下降了 5.6%，也体现了 Transformer 的性能。

## Analysis

Table 5: Analysis of various model ablations on different tasks. Avg. score is a unweighted average of all the results. (*mc*= Mathews correlation, *acc*=Accuracy, *pc*=Pearson correlation)

| Method | Avg. Score | CoLA (mc) | SST2 (acc) | MRPC (F1) | STSB (pc) | QQP (F1) | MNLI (acc) | QNLI (acc) | RTE (acc) |
|---|---|---|---|---|---|---|---|---|---|
| Transformer w/ aux LM (full) | 74.7 | 45.4 | 91.3 | 82.3 | 82.0 | **70.3** | **81.8** | **88.1** | **56.0** |
| Transformer w/o pre-training | 59.9 | 18.9 | 84.0 | 79.4 | 30.9 | 65.5 | 75.7 | 71.2 | 53.8 |
| Transformer w/o aux LM | **75.0** | **47.9** | **92.0** | **84.9** | **83.2** | 69.8 | 81.1 | 86.9 | 54.4 |
| LSTM w/ aux LM | 69.1 | 30.3 | 90.5 | 83.2 | 71.8 | 68.1 | 73.7 | 81.1 | 54.6 |

- Overall, the trend suggests that larger datasets benefit from the auxiliary objective but smaller datasets do not.
- A 5.6 average score drop when using the LSTM instead of the Transformer.
- The lack of pre-training hurts performance across all the tasks, resulting in a 14.8% decrease compared to our full model.

# 4. BERT

## 4.1 引言

上周 Google 放出了他们的语言模型预训练方法，瞬时受到了各界广泛关注，不少媒体公众号也进行了相应报道，那我们来看看这篇论文《BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding》。这篇论文把预训练语言表示方法分为了基于特征的方法（代表 ELMo）和基于微调的方法（代表 OpenAI GPT）。而目前这两种方法在预训练时都是使用单向的语言模型来学习语言表示。

## Motivation

- Pre-trained language representations
  - **Feature-based**: ELMo
  - **Fine-tuning**: OpenAI GPT
- Both approaches share the same objective function during pre-training, where they use **unidirectional language models** to learn general language representations

这篇论文中，作者们证明了使用双向的预训练效果更好。其实这篇论文方法的整体框架和 GPT 类似，是进一步的发展。具体的，他们 BERT 是使用 Transformer 的编码器来作为语言模型，在语言模型预训练的时候，提出了两个新的目标任务（即遮挡语言模型 MLM 和预测下一个句子的任务），最后在 11 个 NLP 任务上取得了 SOTA。

## Contributions

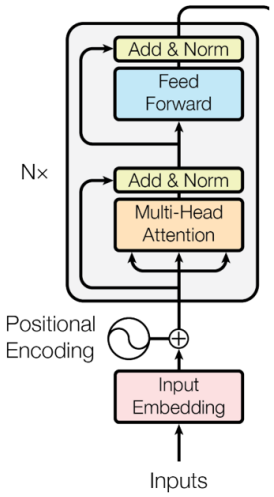
- BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers.
  - We demonstrate the importance of bidirectional pre-training for language representations.
  - Use Transformer encoder as the LM and a new pre-training objective: the "masked language model" (MLM).
  - Introduce a "next sentence prediction" task that jointly pre-trains text-pair representations.
  - They show that pre-trained representations eliminate the needs of many heavily-engineered task-specific architectures. BERT advances the state-of-the-art for eleven NLP tasks.

### 4.2 方法

在语言模型上，BERT 使用的是 Transformer 编码器，并且设计了一个小一点 Base 结构和一个更大的 Large 网络结构。

## Model Architecture

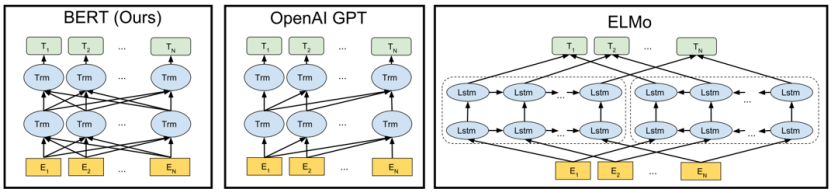
- BERT's model architecture is a **multi-layer bidirectional Transformer encoder**.
  - BERT$_{BASE}$: L=12, H=768, A=12, Total Parameters=110M. (It was chosen to have an identical model size as OpenAI GPT for comparison purposes. )
  - BERT$_{LARGE}$: L=24, H=1024, A=16, Total Parameters=340M.




对比一下三种语言模型结构，BERT 使用的是 Transformer 编码器，由于 self-attention 机制，所以模型上下层直接全部互相连接的。而 OpenAI GPT 使用的是 Transformer 编码器，它是一个需要从左到右的受限制的 Transformer，而 ELMo 使用的是双向 LSTM，虽然是双向的，但是也只是在两个单向的 LSTM 的最高层进行简单的拼接。所以作者们任务只有 BERT 是真正在模型所有层中是双向的。

## Model Architecture

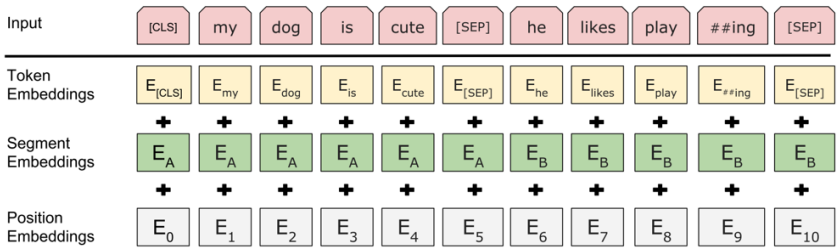
- Among three, only BERT representations are jointly conditioned on both left and right context in all layers.




- **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.

而在模型的输入方面，BERT 做了更多的细节，如下图。他们使用了 WordPiece embedding 作为词向量，并加入了位置向量和句子切分向量。并在每一个文本输入前加入了一个 CLS 向量，后面会有这个向量作为具体的分类向量。

## Input Representation




- WordPiece embeddings with a 30,000 token vocabulary.
- Learned positional embeddings with supported sequence lengths up to 512 tokens.
- The first token of every sequence is always the special classification embedding ([CLS]).
- Sentence pairs are packed together into a single sequence.

在语言模型预训练上，他们不在使用标准的从左到右预测下一个词作为目标任务，而是提出了两个新的任务。第一个任务他们称为 MLM，即在输入的词序列中，随机的挡上 15%的词，然后任务就是去预测挡上的这些词，可以看到相比传统的语言模型预测目标函数，MLM 可以从任何方向去预测这些挡上的词，而不仅仅是单向的。但是这样做会带来两个缺点：1）预训练用[MASK]提出挡住的词后，在微调阶段是没有[MASK]这个词的，所以会出现不匹配；2）预测 15%的词而不是预测整个句子，使得预训练的收敛更慢。但是对于第二点，作者们觉得虽然是慢了，但是效果提升比较明显可以弥补。

## Pre-training Task1: Masked LM (MLM)

- Masking some percentage of the input tokens at random, and then predicting only those masked tokens.
- The MLM objective allows the representation to fuse the left and the right context, which allows us to pre-train a deep bidirectional Transformer.
- Downsides:
  - Create a mismatch between pre-training and fine-tuning, since the [MASK] token is never seen during fine-tuning.
  - Only 15% of tokens are predicted in each batch, which suggests that more pre-training steps may be required for the model to converge.

对于第一点他们采用了下面的技巧来缓解，即不是总是用[MASK]去替换挡住的词，在 10%的时间用一个随机词取替换，10%的时间就用这个词本身。

## Pre-training Task1: Masked LM

- The [MASK] token is never seen during fine-tuning.
- To mitigate this, Rather than always replacing the chosen words with [MASK], the data generator will do the following:
  - 80% of the time: Replace the word with the [MASK] token, e.g., my dog is hairy → my dog is [MASK]
  - 10% of the time: Replace the word with a random word, e.g., my dog is hairy → my dog is apple
  - 10% of the time: Keep the word unchanged, e.g., my dog is hairy → my dog is hairy.

而对于传统语言模型，并没有对句子之间的关系进行考虑。为了让模型能够学习到句子之间的关系，作者们提出了第二个目标任务就是预测下一个句子。其实就是一个二元分类问题，50%的时间，输入一个句子和下一个句子的拼接，分类标签是正例，而另 50%是输入一个句子和非下一个随机句子的拼接，标签为负例。最后整个预训练的目标函数就是这两个任务的取和求似然。

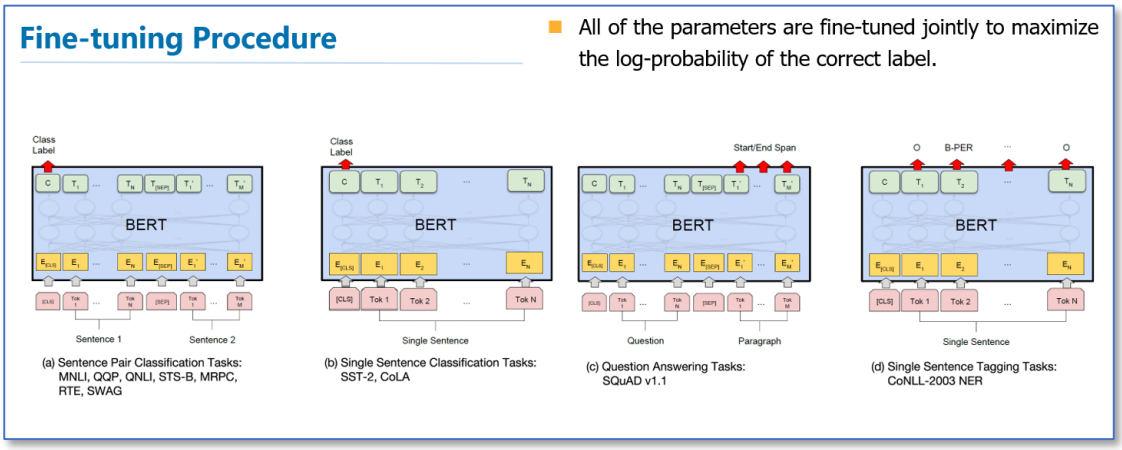
## Pre-training Task2: Next Sentence Prediction

- In order to train a model that understands sentence relationships, they pre-train a binarized next sentence prediction task.
- When choosing the sentences A and B for each pre-training example, 50% of the time B is the actual next sentence that follows A, and 50% of the time it is a random sentence from the corpus.

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

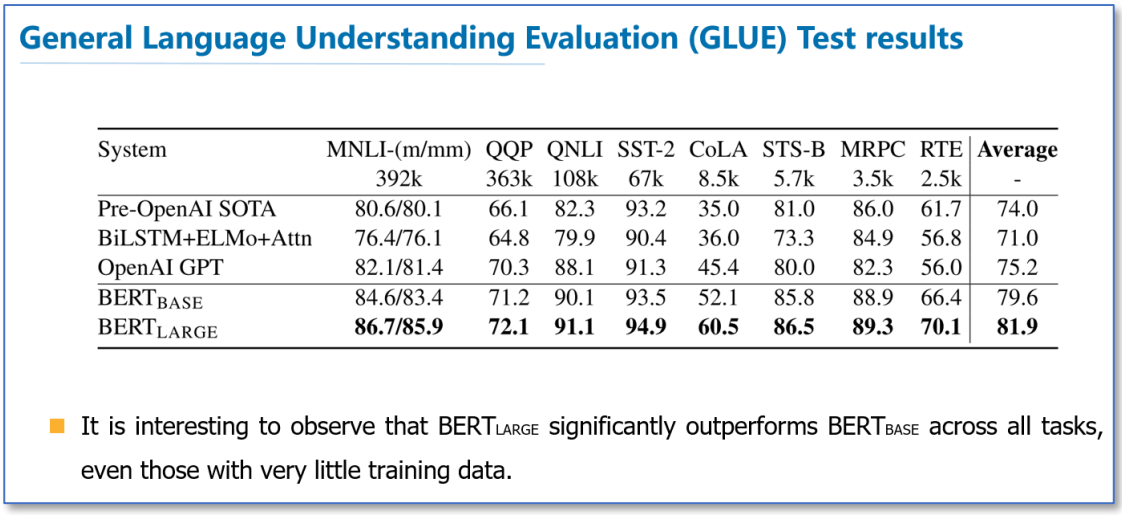
在微调阶段，不同任务的模型如下图，只是在输入层和输出层有所区别，然后整个模型所有参数进行微调。




## 4.3 实验

下面我们列出一下不同 NLP 上 BERT 的效果。

GLUE 结果：




| 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** |

QA 结果：

## SQuAD results

Data augmentation in our submitted system by jointly training on SQuAD and TriviaQA

| System | Dev EM | Dev F1 | Test EM | Test 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** |

实体识别结果：

## NER results

| 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.

SWAG 结果：

## Situations With Adversarial Generations (SWAG) results

Given a sentence from a video captioning dataset, the task is to decide among four choices the most plausible continuation. For example:

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.

| 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 |

可以看到在这些所有 NLP 任务上，BERT 都取得了 SOTA，而且相比 EMLo 和 GPT 的效果提升还是比较大的。

在预训练实验分析上，可以看到本文提出的两个目标任务的作用还是很有效的，特别是在 MLM 这个目标任务上。

## Effect of Pre-training Tasks

| 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 |

- **No NSP**: A model which is trained using the "masked LM" (MLM) but without the "next sentence prediction" (NSP) task.
- **LTR**: A model which is trained using a Left-to-Right (LTR) LM
- **+BiLSTM**: adding a randomly initialized BiLSTM on top of it for fine-tuning.

作者也做了模型规模的实验，大规模的模型效果更好，即使在小数据集上。

## Effect of Model Size

| 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 |

Larger models lead to a strict accuracy improvement across all four datasets, even for MRPC which only has 3,600 labeled training examples.

此外，作者也做了像 ELMo 当成特征加入的实验，从下图可以看到，当成特征加入最好效果能达到 96.1%和微调的 96.4%差不多，说明 BERT 对于基于特征和基于微调这两种方法都是有效的。

## Feature-based Approach with BERT

| 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 |

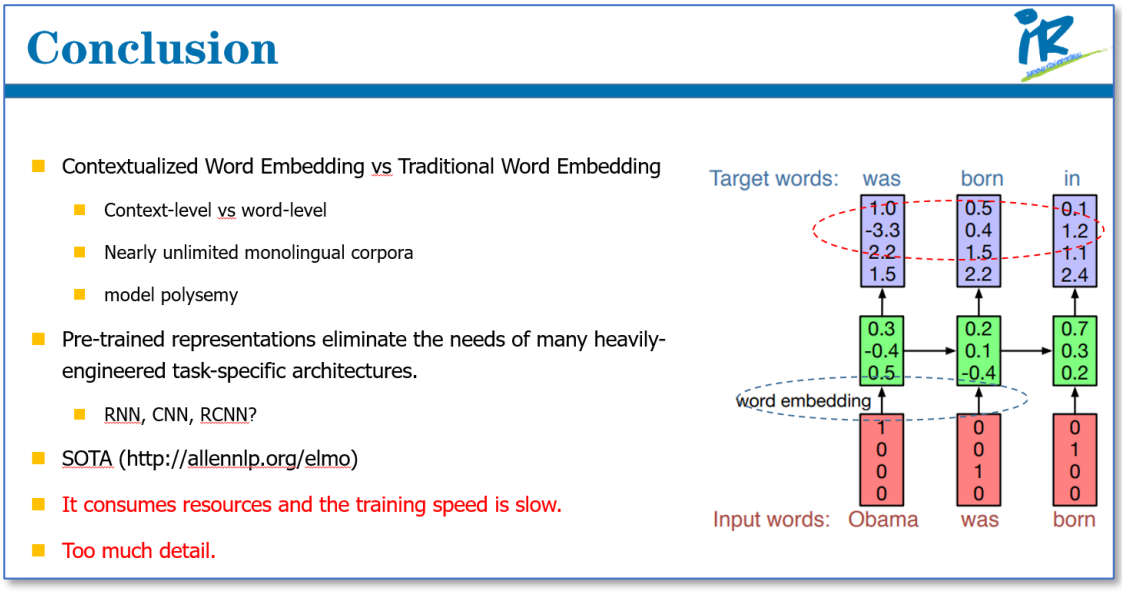
Table 7: Ablation using BERT with a feature-based approach on CoNLL-2003 NER. The activations from the specified layers are combined and fed into a two-layer BiLSTM, without backpropagation to BERT.

- This demonstrates that BERT is effective for both the fine-tuning and feature-based approaches

## 5. 总结

最后进行简单的总结，和传统的词向量相比，使用语言模型预训练其实可以看成是一个句子级别的上下文的词表示，它可以充分利用大规模的单语语料，并且可以对一词多义进行建模。而且从后面两篇论文可以看到，通过大规模语料预训练后，使用统一的模型或者是当成特征直接加到一些简单模型上，对各种 NLP 任务都能取得不错的效果，说明很大程度上

缓解了具体任务对模型结构的依赖。在目前很多评测上也都取得了 SOTA。ELMo 也提供了官网供大家使用。但是这些方法在空间和时间复杂度上都比较高，特别是 BERT，在论文中他们训练 base 版本需要在 16 个 TGPU 上，large 版本需要在 64 个 TPU 上训练 4 天，对于一般条件，一个 GPU 训练的话，得用上 1 年。还有就是可以看出这些方法里面都存在很多工程细节，一些细节做得不好的话，效果也会大大折扣。




## 参考文献


[1] Peters, M. E. et al. Deep contextualized word representations. naacl (2018).

[2] Radford, A. & Salimans, T. Improving Language Understanding by Generative Pre-Training. (2018).

[3] Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2018).

[4] Peters, M. E., Ammar, W., Bhagavatula, C. & Power, R. Semi-supervised sequence tagging with bidirectional language models. Acl (2017).