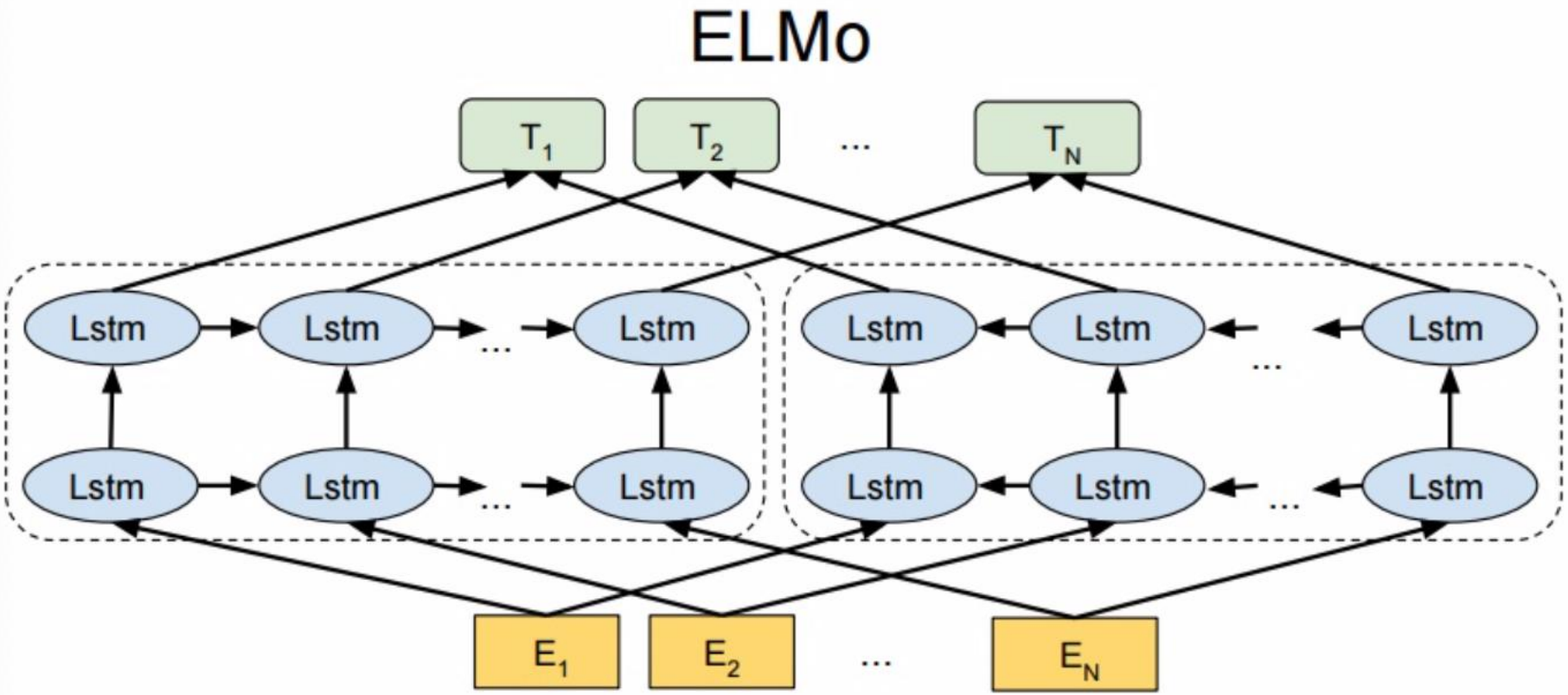
BERT论文精读

--Anruihe

Abstract

- 我们引入了一种新的语言表征模型BERT，即Bidirectional Encoder Representations from Transformers（来自Transformers的双向编码器表示）。与最近的语言表征模型不同（ELMo和GPT），**BERT旨在通过考虑未标记文本的左右（即双向）上下文（context）来预训练文本的深度双向表征**。因此，只需要一个额外的输出层，就可以对预训练的BERT模型进行微调，从而为各种任务(如问题回答和语言推断)创建最先进的模型，而无需对特定于任务的体系结构进行实质性修改。
- BERT在概念上很简单，在实验上很强大。它在11个自然语言处理任务上获得了最新的结果，包括将GLUE得分推至80.5%(绝对提高7.7%)，将多MultiNLI accuracy推至86.7%(绝对提高4.6%)，将SQuAD v1.1问答测试F1推至93.2(绝对提高1.5分)，将SQuAD v2.0测试F1推至83.1(绝对提高5.1分)。

ELMo



宏观上ELMo分三个主要模块.

最底层黄色标记的Embedding模块.

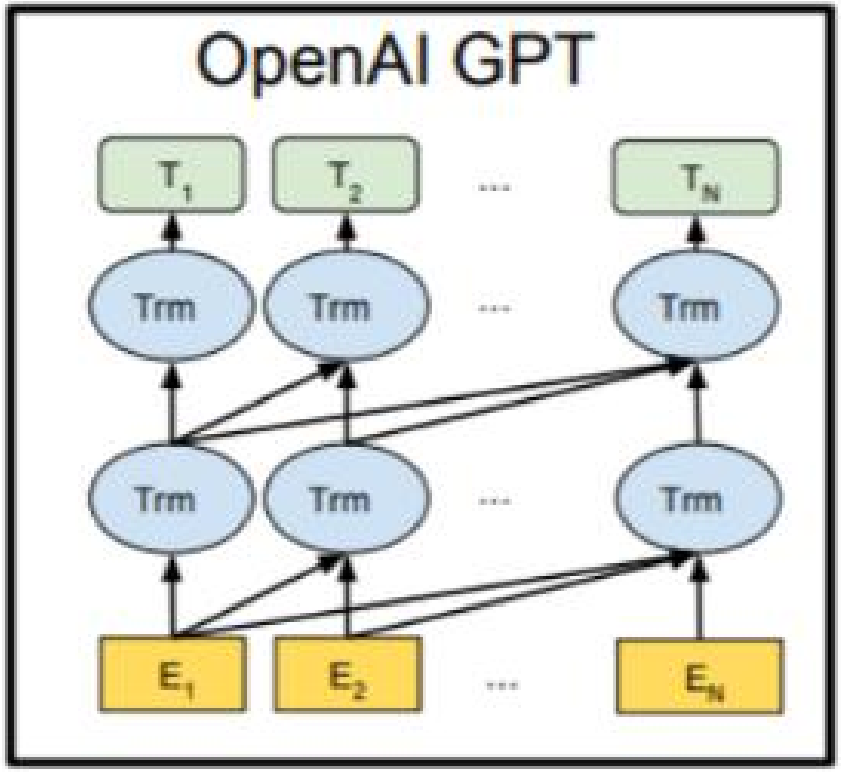
中间层蓝色标记的两部分双层LSTM模块.

最上层绿色标记的词向量表征模块.

- Embedding模块: ELMo最底层的词嵌入采用CNN对字符级进行编码, 本质就是获得一个静态的词嵌入向量作为网络的底层输入.
- 两部分的双层LSTM模块:
- 这是整个ELMo中最重要的部分, 架构中分成左侧的前向LSTM网络, 和右侧的反向LSTM网络.
- ELMo的做法是我们只预训练一个Language Model, 而word embedding是通过输入的句子实时给出的, 这样单词的嵌入向量就包含了上下文的信息.

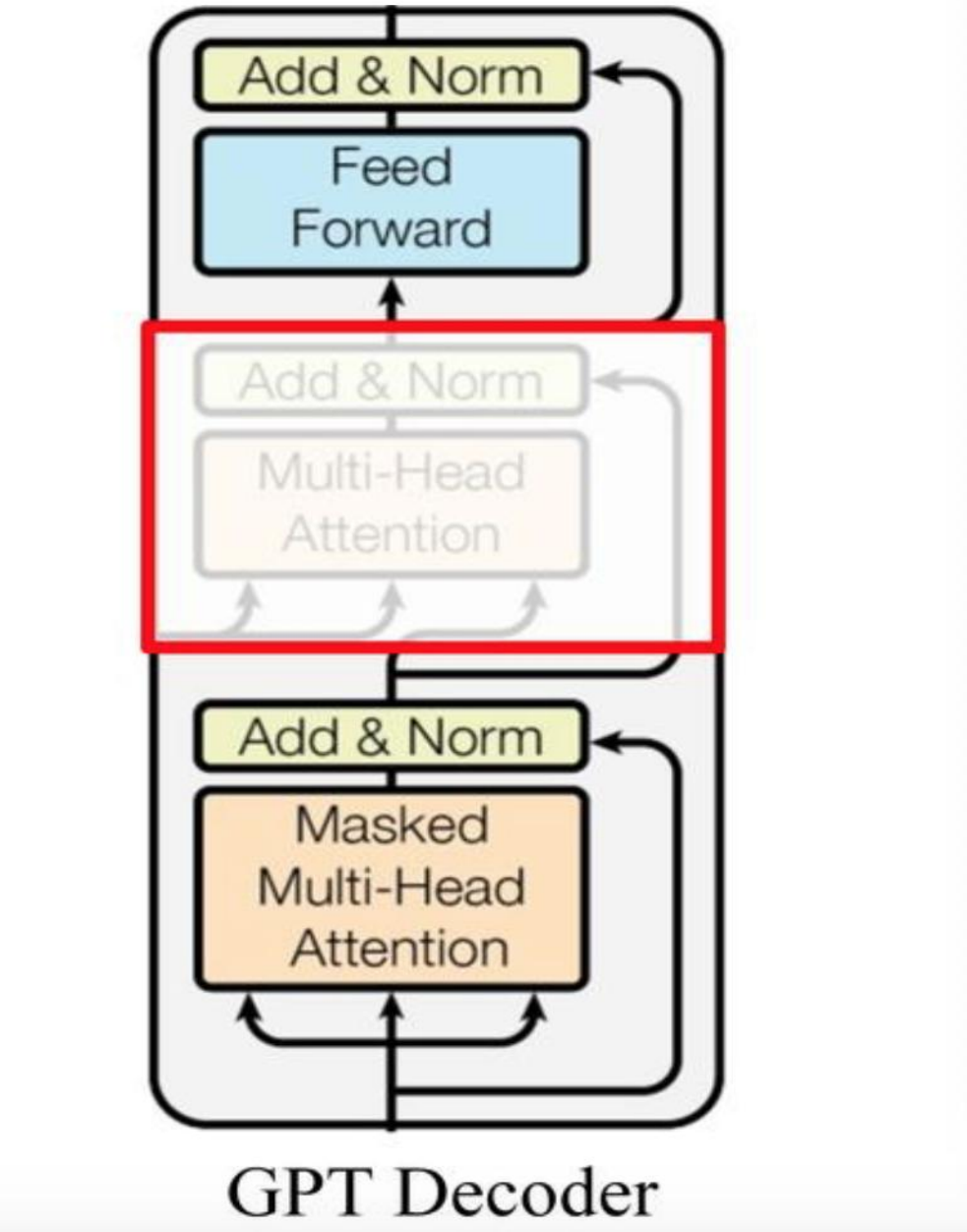
- ELMo的本质思想:
- 首先用一个语言模型学好一个单词的word embedding, 此时是无法区分多义词的, 但没关系. 当实际使用word embedding的时候, 该单词已经具备了特定的上下文信息, 这个时候可以根据上下文单词的语义去调整单词的word embedding表示, 这样经过调整后得到的word embedding向量就可以准确的表达单词在当前上下文中的真实含义了, 也就自然的解决了多义词问题.
- 结论就是ELMo模型是个根据当前上下文对word embedding动态调整的语言模型.

GPT



OpenAI GPT模型是在Google BERT模型之前提出的, 与BERT最大的区别在于GPT采用了传统的语言模型方法进行预训练, 即使用单词的上文来预测单词, 而BERT是采用了双向上下文的信息共同来预测单词.

- 作为两大模型的直接对比, BERT采用了Transformer的Encoder模块, 而GPT采用了Transformer的Decoder模块. 并且GPT的Decoder Block和经典Transformer Decoder Block还有所不同, 如右图所示:
- 经典的Transformer Decoder Block包含3个子层, 分别是Masked Multi-Head Attention层, encoder-decoder attention层, 以及Feed Forward层. 但是在GPT中取消了第二个encoder-decoder attention子层, 只保留Masked Multi-Head Attention层, 和Feed Forward层.



Introduction

- **Language model pre-training**
- 预训练语言模型表现出来能够提升很多自然语言处理任务的效果. 这些包含句子界别的任务, 如自然语言推断和paraphrasing, 目的在于通过分析全局预测句子间的关系, 也有token级别的任务如命名实体识别和问答, 这类任务中模型需要产生token级别精确的输出. ("token" 通常指的是被切分出来的文本单元，可以是单词、子词或者字符，具体取决于正在处理的任务和需求)

- 现有两种将预训练语言表示应用在下游任务的方法: feature-based和fine-tuning.
- feature-based: 用图像理解, 就是一个网络结构初始学到的是图像的纹理色彩角点等特征, 这里就是将一个网络的一层或者多层拿走, 在另一个任务上不对这些层进行调整, 而是仅仅进行特征提取, 然后再输入进任务特定构造的网络结构中.(task-specified)

- fine-tuning based:bert使用的方式, 到下游任务不对bert模型做大调整, 只在最后加一些简单的层, 然后微调时整个模型包括bert都学习调整.

- 作者认为，目前的方法限制了预训练表示的能力，特别是对微调方法. 主要的限制在于标准的语言模型是**单向**的，限制了预训练时的架构选择. 例如，OpenAI GPT的作者使用从左到右的架构，每一个token只能关注自注意力层前面的token. 这种限制是通过句子级别任务中的可选子模块来实现的，但对于应用微调到token级别的任务比如问答是有害的，**因为这类任务中联系从两个方向上下文是非常重要的.**
- 在这篇论文中，作者提出BERT来提升微调. BERT通过masked language model 预训练目标减轻了上面提到的双向限制. The masked language model 随机mask掉输入的部分token，目标是通过上下文信息预测出mask的单词. 和从左到右的语言模型与训练不同，MLM的目标是找到一种融合上下文的表示，这使得能训练一个深度双向transformer.

Related Work
-2.1 Unsupervised Feature-based Approaches

- 图像领域预训练的成功也启发了NLP领域研究，深度学习时代广泛使用的词向量（即词嵌入，Word Embedding）即属于NLP预训练工作。使用深度神经网络进行NLP模型训练时，首先需要将待处理文本转为词向量作为神经网络输入，词向量的效果会影响到最后模型效果。
- 词向量的效果主要取决于训练语料的大小，很多NLP任务中有限的标注语料不足以训练出足够好的词向量，通常使用跟当前任务无关的大规模未标注语料进行词向量预训练，因此预训练的另一个好处是能增强模型的泛化能力。
- 目前，大部分NLP深度学习任务中都会使用预训练好的词向量（如Word2Vec和GloVe等）进行网络初始化（而非随机初始化），从而加快网络的收敛速度。

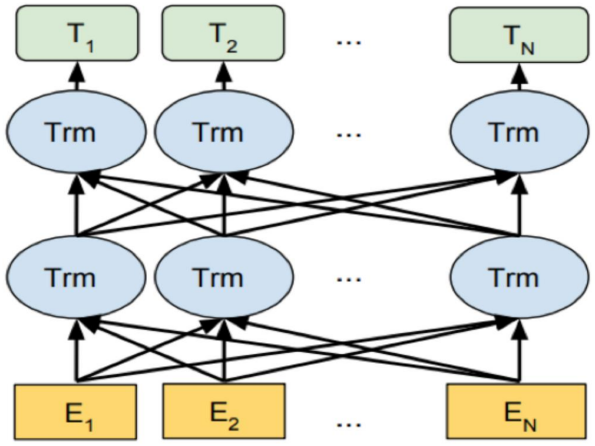
- 预训练词向量通常只考虑编码词汇间的关系，**对上下文信息考虑不足**，**且无法处理一词多义问题**。如“bank”一词，根据上下文语境不同，可能表示“银行”，也可能表示“岸边”，却对应相同的词向量，这样显然是不合理的。
- 为了更好的考虑单词的上下文信息，Context2Vec使用两个双向长短时记忆网络（Long Short Term Memory，LSTM）来分别编码每个单词左到右（Left-to-Right）和右到左（Right-to-Left）的上下文信息。
- 类似地，ELMo也是基于大量文本训练深层双向LSTM网络结构的语言模型。ELMo在词向量的学习中考虑深层网络不同层的信息，并加入到单词的最终Embedding表示中，在多个NLP任务中取得了提升。
- ELMo这种使用预训练语言模型的词向量作为特征输入到下游目标任务中，被称为Feature-based方法。

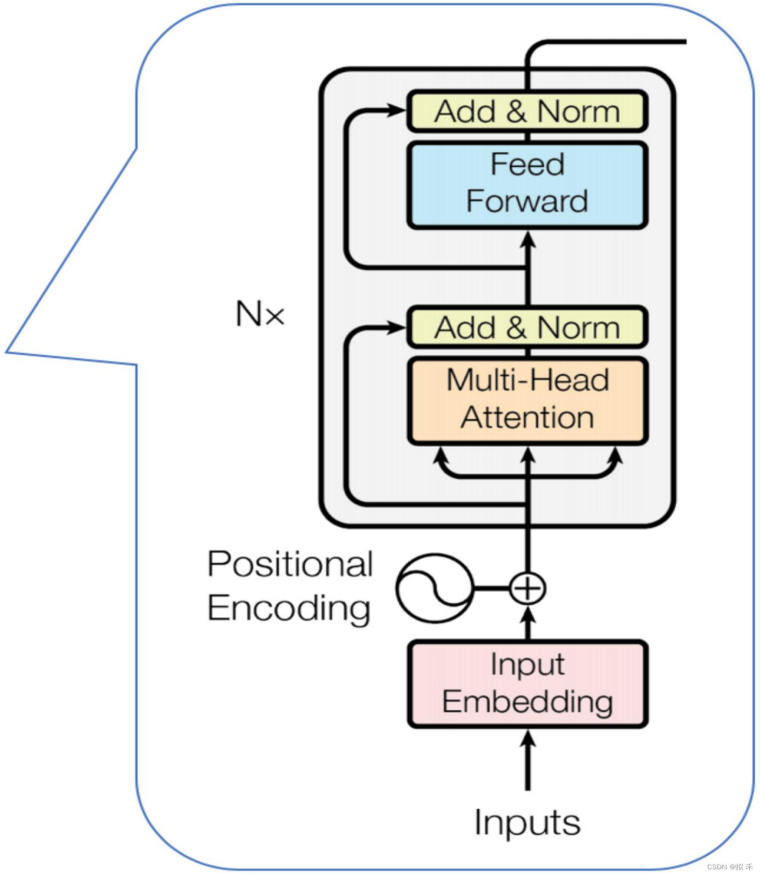
2.2 Unsupervised Fine-tuning Approaches

- 另一种方法是微调（Fine-tuning）。
- GPT（只考虑单向的语义信息）、BERT和后续的预训练工作都属于这一范畴，直接在深层Transformer网络上进行语言模型训练，收敛后针对下游目标任务进行微调，不需要再为目标任务设计Task-specific网络从头训练。

3 BERT

- BERT是基于Transformer的深度双向语言表征模型，基本结构下图所示，本质上是利用Transformer结构构造了一个多层双向的Encoder网络。Transformer是Google在2017年提出的基于自注意力机制（Self-attention）的深层模型，在包括机器翻译在内的多项NLP任务上效果显著，超过RNN且训练速度更快。





- 在预训练阶段，作者的模型是在无标签数据上做不同的预训练任务来训练．在微调阶段，BERT模型首先用预训练参数初始化，然后所有的参数用有标签数据通过下游任务来微调．虽然下游任务被初始化为相同的预训练参数，但每个下游任务有不同的微调模型．

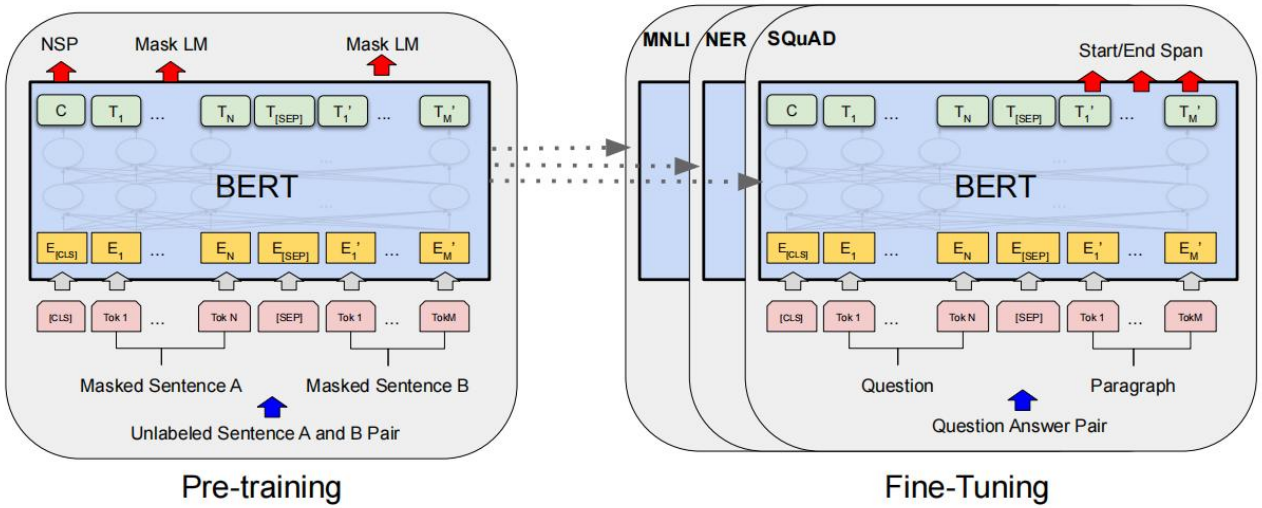
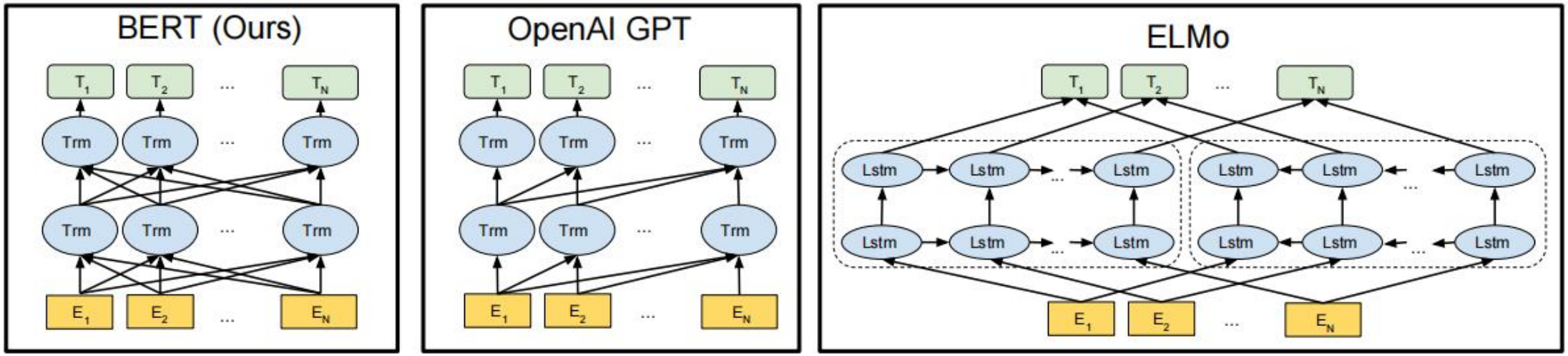


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

- 预训练模型架构的差异如下图所示。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lstm的连接来为下游任务生成特征。
- 在这三种表示中，只有BERT表示同时以所有层中的左右上下文为条件。除了架构差异之外，BERT和OpenAI GPT是一种Fine-tuning方法，而ELMo是一种Feature-based的方法。



Input/Output Representations

- 为了让BERT能应对广泛的下游任务, 输入表示要能在一个token 序列中没有歧义的表示单个句子或者句子对. 在这项工作中, 一个句子可以是任意跨度的句子, 而不一定是实际的语言句子. 一个序列指的是BERT的输入token序列, 可能是一个句子或者两个句子打包.
- 作者使用包含30,000个token的WordPiece embedding. 每个句子的第一个token总是一个特殊符号([CLS]). CLS对应的隐藏层的最后一层的输出通常被用作分类任务. 句子对被打包成一个序列. 作者用两种方式来区分两个句子. 首先, 用特殊符号[SEP]分割句子. 第二点, 给每个token添加一个embedding来指示它是句子A或者句子B.

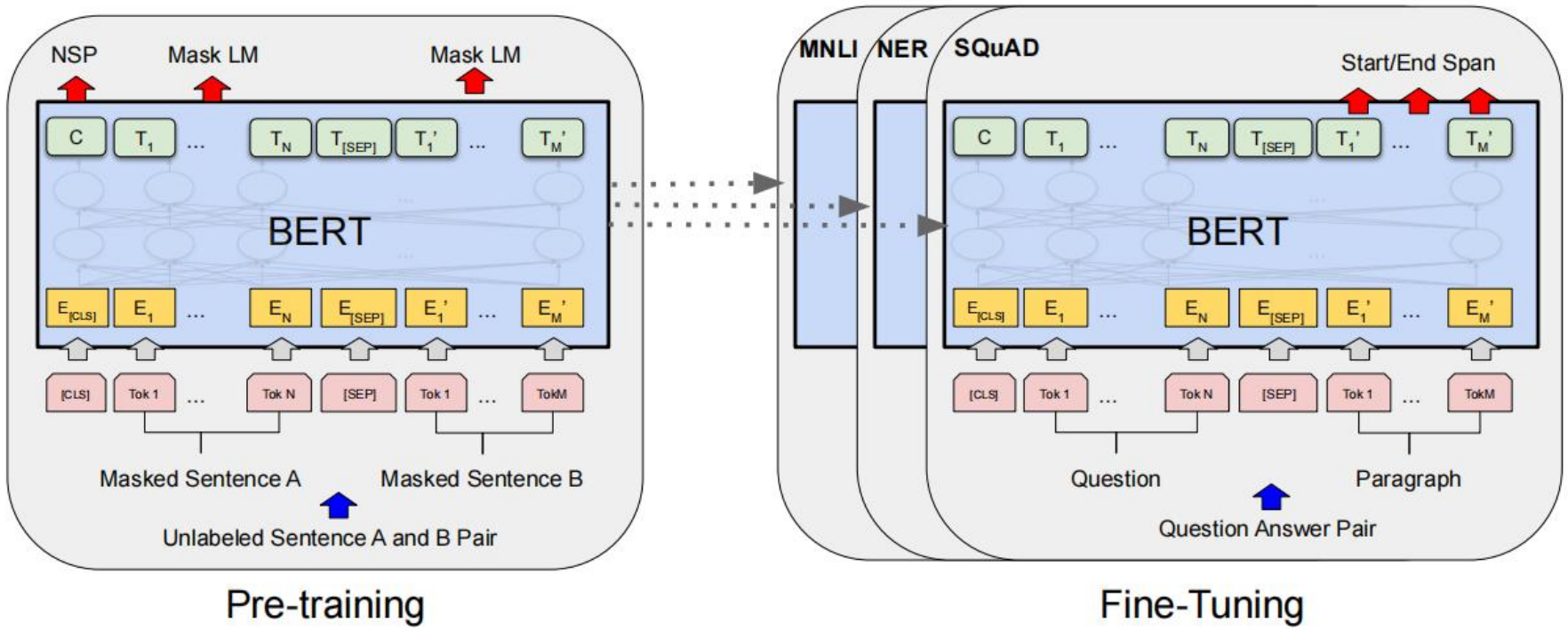


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

- 对于给定的token，输入表示是由token对应的embedding，分块部分和位置embedding三部分加和构成的.

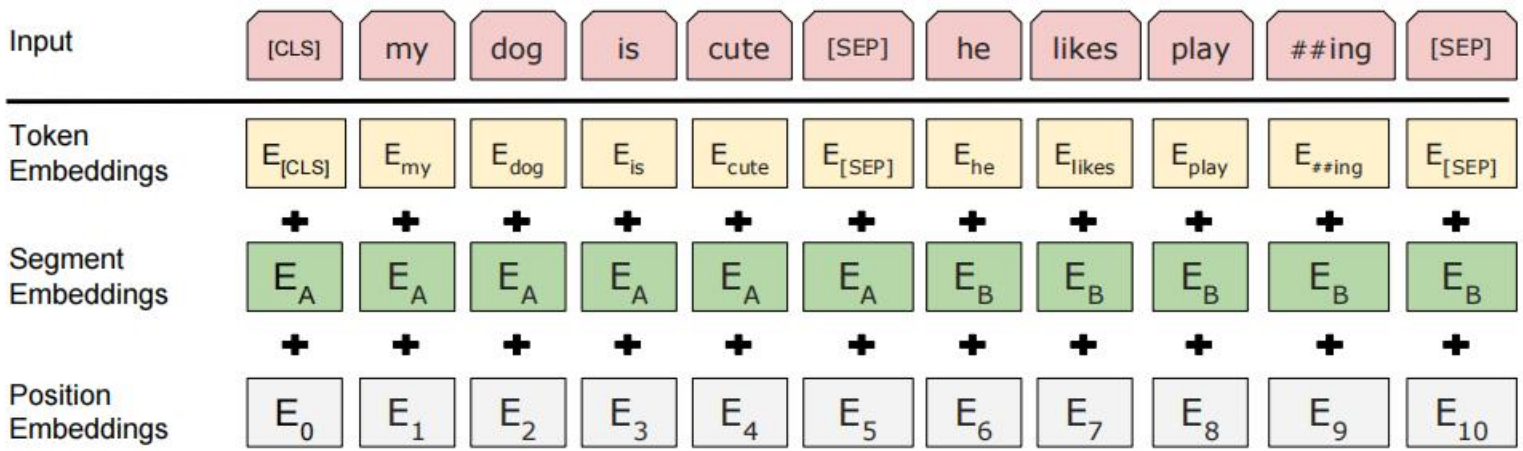


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

3.1 Pre-training BERT

- 不同于以往的方法, 作者不使用传统的从左到右或者从右到左语言模型预训练BERT. 而是使用两个无监督任务预训练BERT.

Task #1: Masked LM

- 为了训练深度双向表示，我们简单地随机屏蔽一定百分比的输入标记，然后预测那些被屏蔽的标记。我们将此过程称为“Masked LM”（MLM），它在文献中通常被称为完形填空任务（Taylor，1953）。所有的实验中，我们随机屏蔽了每个序列中所有 WordPiece 标记的 15%。

- 尽管这使我们能够获得双向预训练模型，但缺点是我们在预训练和微调之间造成了不匹配，因为在微调期间不会出现 [MASK] 标记。为了缓解这种情况，我们并不总是用实际的 [MASK] 标记替换“masked”单词。训练数据生成器随机选择 15% 的标记位置进行预测。如果选择第 i 个标记，我们通过三种方式将第 i 个标记进行替换替换： 1) 80% 的时间用 [MASK] 标记 ； 2) 10% 的时间用一个随机标记 ； 3) 10% 的时间标记不变。然后，最终的隐藏向量 Ti ，通过softmax 激活函数，预测原始标记。具体地：

- 80% 的时间将单词替换为 [MASK] 标记，例如，my dog is hairy → my dog is [MASK]
- 10% 的时间将单词替换为随机单词，例如，my dog is hairy → my dog is apple
- 10% 的时间保持单词不变，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单词。

- 通过这样的策略就可以使得 BERT 不再只对 mask token 敏感，而对所有的 token 都敏感，于是就能抽取出任何 token 的表征信息
- 论文提供了 ablation for different masking procedures，其中基于特征方法的 NER 并没有添加任何 adjust representation 的操作，因此它的 mismatch 会更大一些。作者的实验结果如下表格所示，可以发现，完全使用 [MASK] 替换的 feature-based NER 确实存在严重的 mismatch 的问题。同样的，完全使用随机替换的方式也并不是很好

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI	NER	
			Fine-tune	Fine-tune	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: Ablation over different masking strategies.

Task #2: Next Sentence Prediction (NSP)

- 许多重要的下游任务，例如问答 (Question Answering，QA) 和自然语言推理 (Natural Language Inference，NLI)，都是基于对两个句子之间关系的理解，而语言建模无法直接捕捉到这一点。为了训练一个理解句子关系的模型，我们预先训练一个二值化的下一个句子预测任务（binarized next sentence prediction task），该任务可以从任何单语语料库（monolingual corpus）中轻松生成。
- 为了训练一个理解句子间关系的模型，引入一个下一句预测任务。这一任务的训练语料可以从语料库中抽取句子对包括两个句子A和B来进行生成，其中50%的概率B是A的下一个句子，50%的概率B是语料中的一个随机句子。NSP任务预测B是否是A的下一句。NSP的目的是获取句子间的信息，这点是语言模型无法直接捕捉的。
- Google的论文结果表明，这个简单的任务对问答和自然语言推理任务十分有益，但是后续一些新的研究发现，去掉NSP任务之后模型效果没有下降甚至还有提升。我们在预训练过程中也发现NSP任务的准确率经过1-2个Epoch训练后就能达到98%-99%，去掉NSP任务之后对模型效果并不会有太大的影响。

3.2 Fine-tuning BERT

- 除了输出层之外，在预训练和微调中也使用了相同的架构。相同的预训练模型参数用于初始化不同下游任务（downstream tasks）的模型。在微调过程中，对所有参数进行微调，BERT对每一个词元（token）返回抽取了上下文信息的特征向量。
- 即使下游任务各有不同，使用BERT微调时均只需要增加输出层，但根据任务的不同，输入的表示，和使用的BERT特征也会不一样。

4 Experiments

4.1 GLUE

- The General Language Understanding Evaluation (GLUE) 是多个自然语言理解任务的集合.
- 对于 BERTLARGE，作者发现微调有时在小型数据集上不稳定，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打乱（shuffle）和分类器层初始化。

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: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). 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.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

4.2 SQuAD v1.1

- 斯坦福问答数据集 (Stanford Question Answering Dataset，SQuAD v1.1) 是 10 万个众包问答对的集合。给出一个问题和一段来自维基百科的回答，任务是预测文章中的答案文本跨度。

System	Dev		Test	
	EM	F1	EM	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: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

4.3 SQuAD v2.0

- SQuAD 2.0 任务扩展了 SQuAD 1.1 的问题定义，允许在提供的段落中不存在简短答案的可能性，使问题更加现实。

F1分数是一种衡量分类模型性能的指标，通常用于衡量模型的准确率和召回率

EM分数是用于评估问答系统、对话系统或类似任务的性能指标之一，它代表精确匹配（Exact Match）的比例。在这些任务中，系统需要给出一个答案或者回复，而EM分数是衡量系统输出是否完全与预期答案匹配的指标。

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: SQuAD 2.0 results. We exclude entries that use BERT as one of their components.

4.4 SWAG

- Situations With Adversarial Generations (SWAG) 数据集包含 113k 句对完成示例，用于评估基于常识的推理。给定一个句子，任务是在四个选项中选择最合理的延续。

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 Dev and Test accuracies. †Human performance is measured with 100 samples, as reported in the SWAG paper.

5.1 Effect of Pre-training Tasks

- 作者展示了BERT的深度双向(注意力)的重要性, 通过评价两个预训练目标在相同的预训练数据, 微调方法和超参,方法如下:
- No NSP: 双向模型在masked LM上训练, 但没有next sentence prediction(NSP)任务.
- LTR&No NSP: 一个 left-context-only 模型使用标准Left-to-Right(LTR)语言模型, 而不是MLM. 只有上文限制也被应用在微调上, 因为移除了这个限制将导致预训练/微调的不匹配, 导致下游性能下降. 此外, 也被在没有NSP任务下预训练. 除了使用更大的训练数据, 输入表示和微调模式, 其他的和GPT相同.

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.

5.2 Effect of Model Size

- 在这一部分, 作者探索模型大小对微调任务准确率的影响. 作者训练了一些BERT模型用不同的层数, 隐藏单元数, 和attention 头数目, 保证其他参数和原本相同.
- 在选定的GLUE任务上的结果展示在表中. 使用5个随机重启的fine-tuning结果得到准确率. 可以发现, 更大的模型在四个模型上都带来严格的准确率提升, 甚至对于只有3600标记的MRPC数据集, 它的本质和预训练任务完全不同. 现有最优模型已经很大了, 作者的模型能够再次基础上做出显著的提升是令人惊讶的. 例如, 最大的transformer有100M编码器参数, 整个模型235M参数. 相对的, BERT base包含110M参数, Large包含340M参数.

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

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.

5.3 Feature-based Approach with BERT

- 在这一部分中，作者通过将BERT应用到CoNLL-2003NER任务中比较了两种方式. 在BERT输入中，作者使用了case-preserving WordPiece模型，并最大程度保留数据提供的上下文. 和标准做法相一致的，作者将这个作为标注任务，但不使用CRF层. 使用第一个sub-token表示作为token级别分类器的输入来打标签.

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 Named Entity Recognition results. Hyperparameters were selected using the Dev set. The reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

Thank you for watching!