一、简介

Bert（Bidirectional Encoder Representations from Transformers）是一个双向的语言模型，基本神经网络单元是 transformer。

预训练语言模型存在两种应用策略：1、基于特征表示（*feature-based*），给下游任务提供每个 token 的向量表示；2、微调（*fine-tuning*），在已经训练好的语言模型上，加入少量的 task-specific-parameters（任务指向参数，即在最后输出层加入额外一层），然后用新的语料进行微调。

本篇论文认为单向的预训练模型限制了其自身的能力, 故本文训练了一种双向的语言模型，并提出了一种新的预训练目标任务：masked language model(MLM)。除了这个掩盖语言模型（MLM），本文介绍了一个"next sentence prediction"任务。

三种预训练模型（Bert、GPT、ELMo）的架构对比：

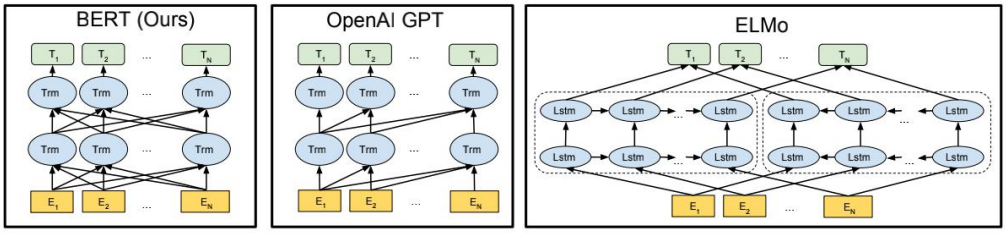

Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations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

二、Bert 模型架构

1) BERT 模型的架构是一个多层的双向的 Transformer 编码。本文中使用 L 表示 Transformer 的层数，H 表示隐藏层的大小，A 表示了自注意力头的数目。

本文报告了两种参数量的模型：

BERT_base:L=12,H=768,A=12,Total Parameters=110M

BERT_large:L=24,H=1024,A=16,Total Parameters=340M

2) 本文输入表示是一个单一的文本句子或一对句子。对于个给定的 token，它的向量表示是由相应的 WordPiece 嵌入（词嵌入）、segment 嵌入（分割嵌入，用于在句子对中区分句子）、position 嵌入（位置嵌入）进行相加得到；下图可视化的展示了输入向量化的一个详细表示方法。

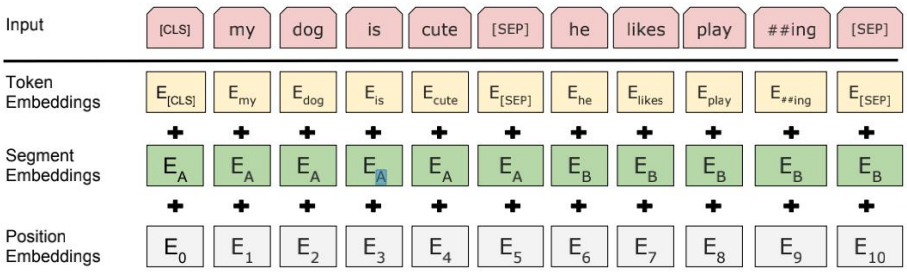

Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

具体细节：

- 使用了 30000 个 token 的 Embeddings，使用##指示切分词；
- 使用了一个可被学习的位置 Embeddings, 可支持 512 个 token 的序列长度；
- 每个序列的第一个 token 总是这个特殊的分类 embedding[CLS]，它的最后隐藏状态的 embedding，在句子分类任务中，常被用来当作聚合的序列向量，对于非分类任务，这个向量常被忽略；
- 句子对被打包为一个句子序列时，为了区分句子，采用了两种区分方式：使用特殊的 token 符号[SEP]、添加了一个可被学习的 Sentence embedding，第一个句子中，每个 token 使用 Sentence A 的 embedding 表示，第二个句子中，每个 token 使用 Sentence B 的 embedding 表示；
- 对于单个句子输入，每个 token 仅使用 Sentence A 的 embedding 表示；

3) 预训练任务

Task #1:Masked LM

为了训练一个深度双向模型，本文采用了一种直观的方法，随机掩饰掉一定比例的输入 tokens, 然后预测这些掩饰掉的 tokens。本文称这种方法为“masked LM” (MLM)。实验中掩饰掉了 15%的输入 token，而不是对每个输入都做重构。尽管这样的 mask 方法能够获得一个预训练模型，但是这在后期的 fine-tune 时，预训练和 fine 的 token 是不一致的。为了减缓这个问题，本文不总是使用[MASK]去替换”masked”的 token，而是采用数据生成器去执行如下的替换方式：

- 80%的时候，采用[MASK]替换调用要 masked 的 token，eg. My dog is hairy --> My dog is [MASK].
- 10%的时候，使用随机的词替换调用要 masked 的 token，eg. My dog is hairy --> My dog is apple.
- 10%的时候，保持原来的 token 不变， eg. My dog is hairy --> My dog is hairy.

Task #2:Next Sentence Prediction

本文使用 50%的语料中，sentence B 是 sentence A 的 next 句子，50%的语料中 sentence A 后面是随机跟随的句子。

Eg. 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

预训练过程：

数据：BooksCopus(800M)、English Wikipedia(2500M)

samples 生成：50%的情况下，样本生成时抽取连续的两个句子 Sentence A 和 Sentence B 作为一个 sample,Sentence B 是 Sentence A 的 next 句子；50%的情况下，样本生成时，随机抽取两个句子，Setentence B 并不真是 Sentence A 的 next 句子。

超参数设置： batch size: 256sequences (256sequences*512 tokens =

128000 tokens/batch)；执行了 1000000steps,大约有 40 个 epoch。使用了 Adam 优化器，学习率 1e-4，β1 = 0.9, β2 = 0.999, L2 权重衰减系数为 0.01,学习率每 10000 个 steps 进行一次 warmup。所有的层采用了 0.1 的 dropout。使用了 gelu 激活函数替代标准的 OpenAI GPT 中的 relu 激活函数。损失函数是 masked 的语言模型的平均似然值和 next sentence 预测的平均似然值的加和。

微调过程：

对于序列级的分类任务，Bert 的微调是直观的。为了得到一个输入序列的固定纬度的聚合向量，使用每个输入序列的第一个位置[CLS]的最后隐藏状态作为句向量的表示。

为了完成分类任务，在 bert 后面增添一个标准的 softmax 分类器，这个唯一添加的参数 W 可以通过该分类任务学习到。Bert 的所有的超参数设置必须保持不变，与预训练的时候完全一致。

超参数：

Batch size：16，32

Learning rate(Adam):5e-5,3e-5,2e-5

Number of epochs:3,4

核心介绍记录到此，其他后续实验详情，请看原始论文。

作者：闫海磊
2019 年 10 月
hailei2014@163.com