Week6

本文对 Bert 进行较为详细的阐述。

一、简述 ElMo, ULMFiT 和 OpenAI GPT

Word2Vec 问题：不能解决一词多义的问题，以及词的多层特性问题；

ELMo(2018): 通过无监督的方式对语言模型进行预训练来学习单词表示。它的思路是用深度的双向 LM((Language Model) 在大量未标注数据上训练语言模型，此解决了上下文问题，可根据语境，得到词向量；

ULMFiT(2018): 这是一种有效的迁移学习方法，可以适用于任何 NLP 任务。它提供了通用领域 LM Pre-training、目标任务 LM Fine-tuning、目标任务分类器 Fine-tuning 三部分框架；

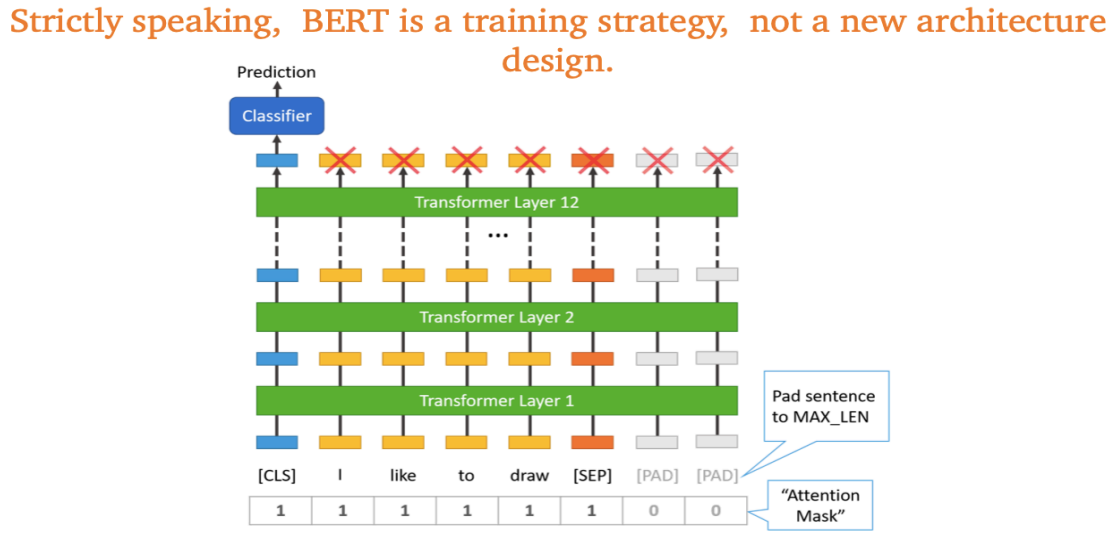
OpenAI GPT(2018): 使用单向 Transformer(多层的 transformer decoder) 学习一个语言模型，对句子进行无监督的 Embedding(Pre-training)， 然后根据具体任务对 Transformer 的参数进行微调(对特定任务进行 Fine-tuning)；

ELMo 和 GPT 最大的问题就是传统的语言模型是单向的 —— 我们根据之前的历史来预测当前词。但是我们不能利用后面的信息。Transformer 学习语言模型，必须用 Mask 来让它看不到未来的信息，所以不能注意到后面的信息。即使 ELMo 训练了双向的两个 LSTM，但是一个 LSTM 只能看一个方向，因此也是无法 "同时" 利用前后两个方向的信息的。ELMo 、 GPT 和 Bert 算法原理图见附件图一。

二、Bert 阐述

Bert 是 Google 以无监督的方式利用大量无标注文本「炼成」的语言模型，使用的是双向 transformer encoder 结构，通过 mask 方法应用，能够同时利用前后两个方向的信息，同时，通过 Next sentence prediction 预测，把上下文关系也学进去了。

Bert 原理图如下：



1. **词嵌入(Embedding)**

由 Token Embeddings、Segment Embedding 和 Position Embedding 求和而成，见附件图二。

Token Embedding 使用的是 Wordpieces tokenization 方法。此方法减少了总词表数量，有效的解决了 OOV 问题，且每个词出现的次数变多，增加了训练信息。但此方法在进行 mask 时，可能 mask 掉一个词的一半，如 pro+##bali+##lity(probability)，只 mask 掉 ##bali，导致训练变易，不利于预测效果，此时需进行人工调节，对整个单词进行 mask。在 NMT 任务中，Transformer 的 encoder 的核心任务是提取完整的句子语义信息，它其实并特别关注某个词的具体位置是什么，所以用 Position Encode 的功能将每个位置区分开很适合。但在 Bert 中，它的 encoder 需要建模完整的 word order。尤其是对于序列标注类的下游任务，模型需要给出每个位置的预测结果。这种时候，完全训练得来的 Position Embedding 就比公式赋值的 Position Encode 要好。

2. **Masked LM(MLM)**

随机遮盖或替换一句话里面的任意字或词，然后让模型通过上下文预测那一个被遮盖或替换的部分，之后做 Loss 的时候也只计算被遮盖部分的 Loss。

随机把一句话中 15% 的 token（字或词）替换成以下内容：

a. 这些 token 有 80% 的几率被替换成 [MASK]，例如 my dog is hairy→my dog is [MASK];

b. 有 10% 的几率被替换成任意一个其它的 token，例如 my dog is hairy→my dog is apple;

c. 有 10% 的几率原封不动，例如 my dog is hairy→my dog is hairy;

这样做的好处是，BERT 并不知道 [MASK] 替换的是哪一个词，而且任何一个词都有可能是被替换掉的，比如它看到的 apple 可能是被替换的词。这样强迫模型在编码当前时刻词的时候不能太依赖当前的词，而要考虑它的上下文，甚至根据上下文进行 "纠错"。比如上面的例子中，模型在编码 apple 时，根据上下文 my dog is，应该把 apple 编码成 hairy 的语义而不是 apple 的语义。

3. Next Sentence Prediction（NSP）

LM 存在的问题是，缺少句子之间的关系，这对许多 NLP 任务很重要。为预训练句子关系模型，bert 使用一个非常简单的二分类任务：将两个句子 A 和 B 链接起来，预测原始文本中句子 B 是否排在句子 A 之后。

Bert 预训练阶段实际上是将上述 2 和 3 两个任务结合起来，同时进行，然后将所有的 Loss 相加，进行训练。

Bert 可通过不同层，提取嵌入向量信息，也可直接接几层，通过 Fine-tuning 做下游任务。

4. **Fine-tuning**

Bert 的 Fine-Tuning 共分为 4 种类型，详见附件图 3。

a. 对于普通的分类任务，输入是一个序列，如图中右上所示，所有的 Token 都是属于同一个 Segment(Id=0)，我们用第一个特殊 Token [CLS]的最后一层输出接上 softmax 进

行分类，用分类的数据来进行 Fine-Tuning。注意在这里，CLS 代表整句话完整信息。

b. 对于相似度计算等输入为两个序列的任务，过程如图左上所示。两个序列的 Token 对应不同的 Segment(Id=0/1)。我们也是用第一个特殊 Token [CLS]的最后一层输出接上 softmax 进行分类，然后用分类数据进行 Fine-Tuning。

c. 第三类任务是序列标注， 比如命名实体识别，输入是一个句子(Token 序列)，除了[CLS]和[SEP]的每个时刻都会有输出的标签，比如 B-PER 表示人名的开始，将句子中各个 token 对应位置的 output 分别送入不同的 Linear，预测出该 token 的标签，然后用输出的标签进行 Fine-Tuning。其实这本质上还是个分类问题，只不过是对每个 token 都要预测一个类别。

d. 第四类是问答类问题，比如 SQuAD v1.1 数据集，输入是一个问题和一段很长的包含答案的文字(Paragraph)，输出在这段文字里找到问题的答案。首先把问题和 Paragraph 表示成一个长的序列， 中间用[SEP]分开，问题对应一个 Segment(id=0)，包含答案的文字对于另一个 Segment(id=1)。这里有一个假设，那就是答案是 Paragraph 里的一段连续的文字(Span)。BERT 把寻找答案的问题转化成寻找这个 Span 的开始下标和结束下标的问题。

如上图的左下所示。对于 Paragraph 的第 i 个 Token，Bert 的最后一层把它编码成Ti，然后我们用一个向量 S(这是模型的参数，需要根据训练数据调整)和它相乘(内积)，计算它是开始位置的得分，因为 Paragraph 的每一个 Token(当然 WordPiece 的中间，比如 ##ing 是不可能是开始的)都有可能是开始可能，我们用 softmax 把它变成概率，然后选择概率最大的作为答案的开始：

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

类似的有一个向量 T，用于计算答案结束的位置。

参考文献：

1. Bert 模型详解，李理，李理的博客，

   http://fancyerii.github.io/2019/03/09/bert-theory/

2. BERT 详解（附带 ELMo、GPT 介绍），mathor，2020，

   https://wmathor.com/index.php/archives/1456/

3. 其它资料参考老师上课 PPT

附件：

图 1

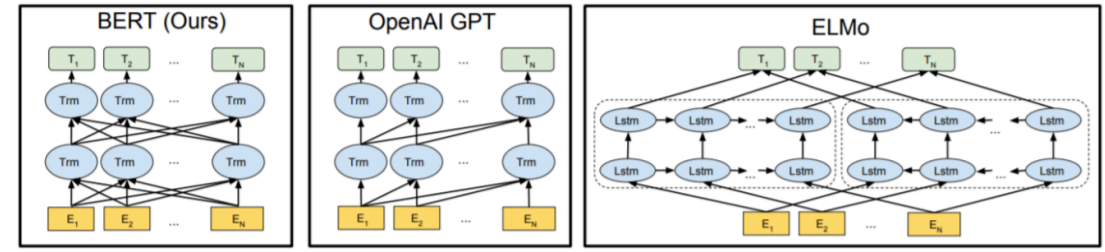


Figure 3: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right context in all layers. In addition to the architecture differences, BERT and OpenAI GPT are fine-tuning approaches, while ELMo is a feature-based approach.

图 2

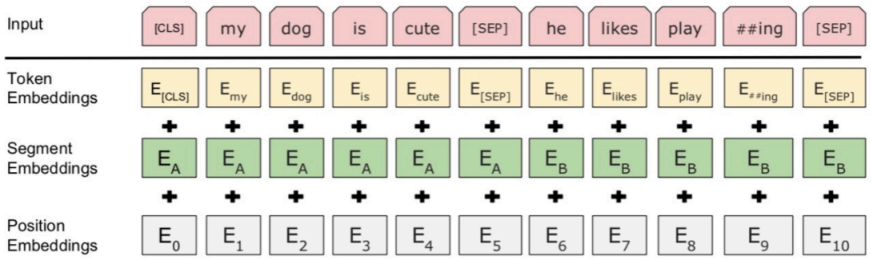
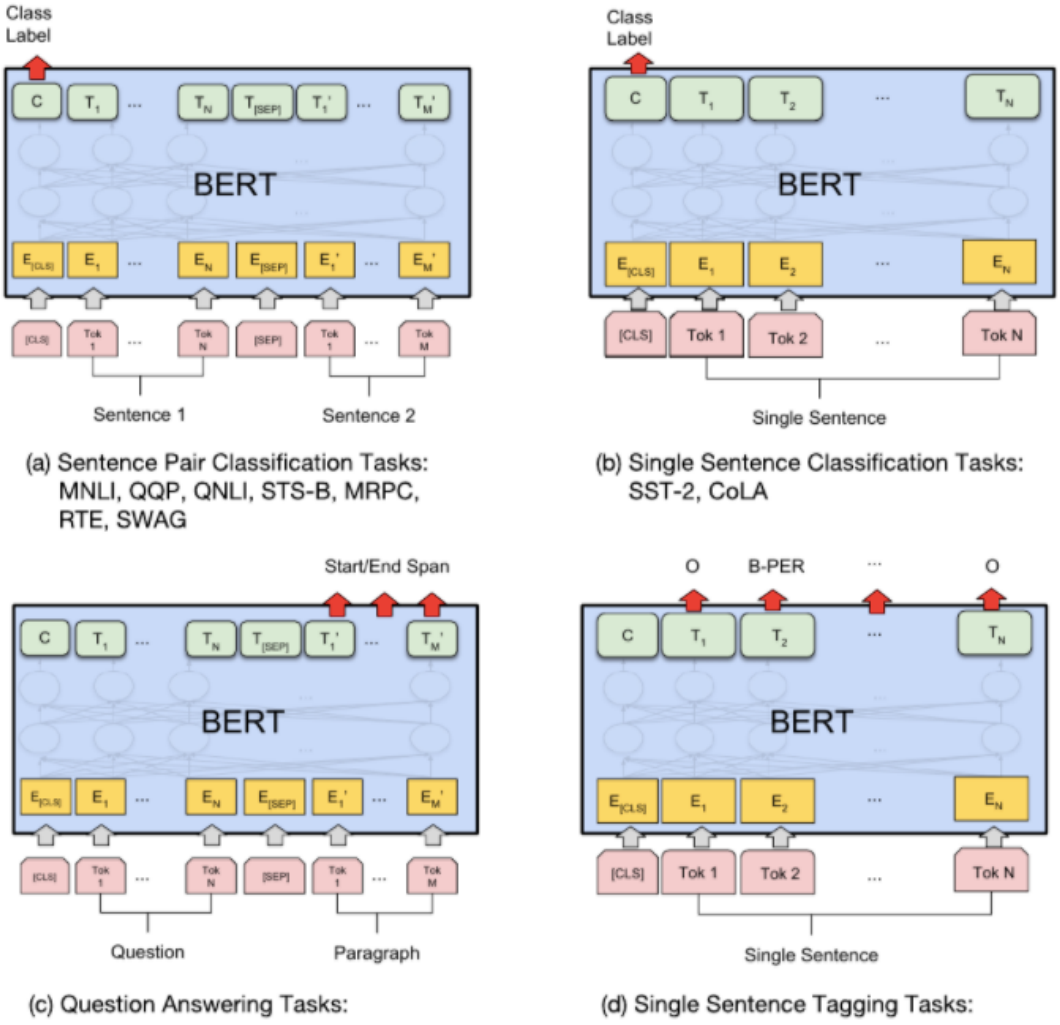


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

图 3



图：BERT的Fine-Tuning