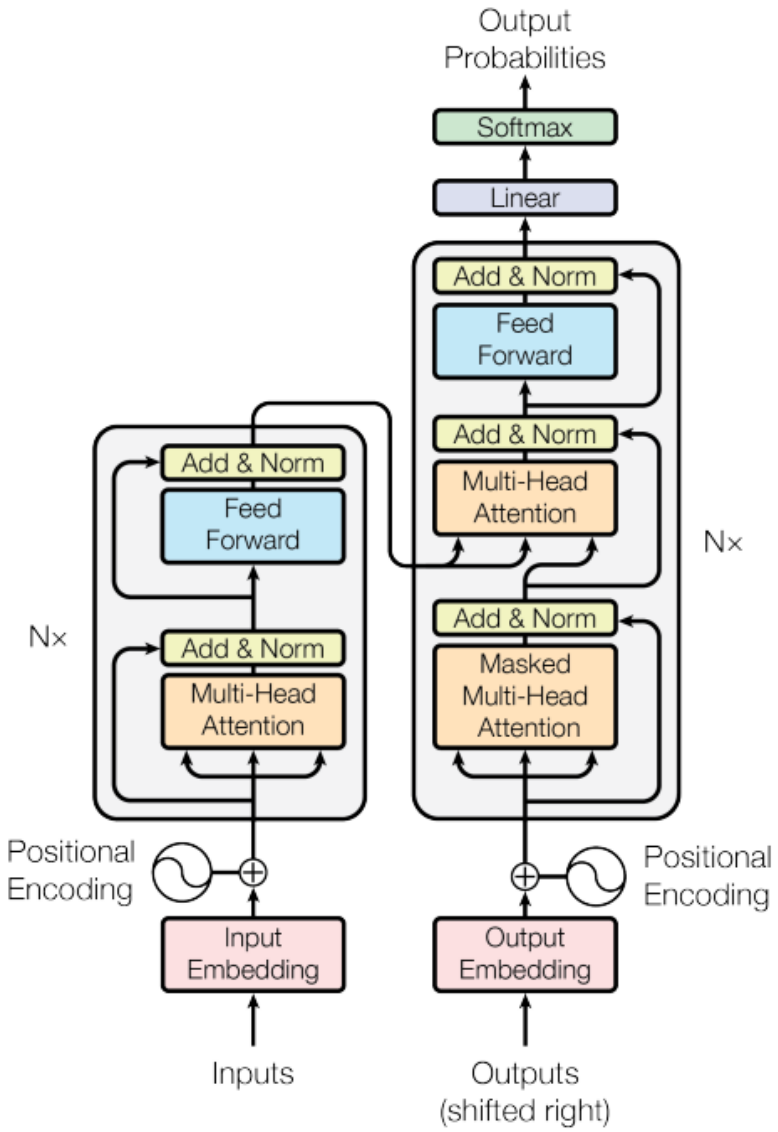
nlp论文2023 10月31-11月15阅读

Attention Is All You Need（transfromer）

1.提出来的目的

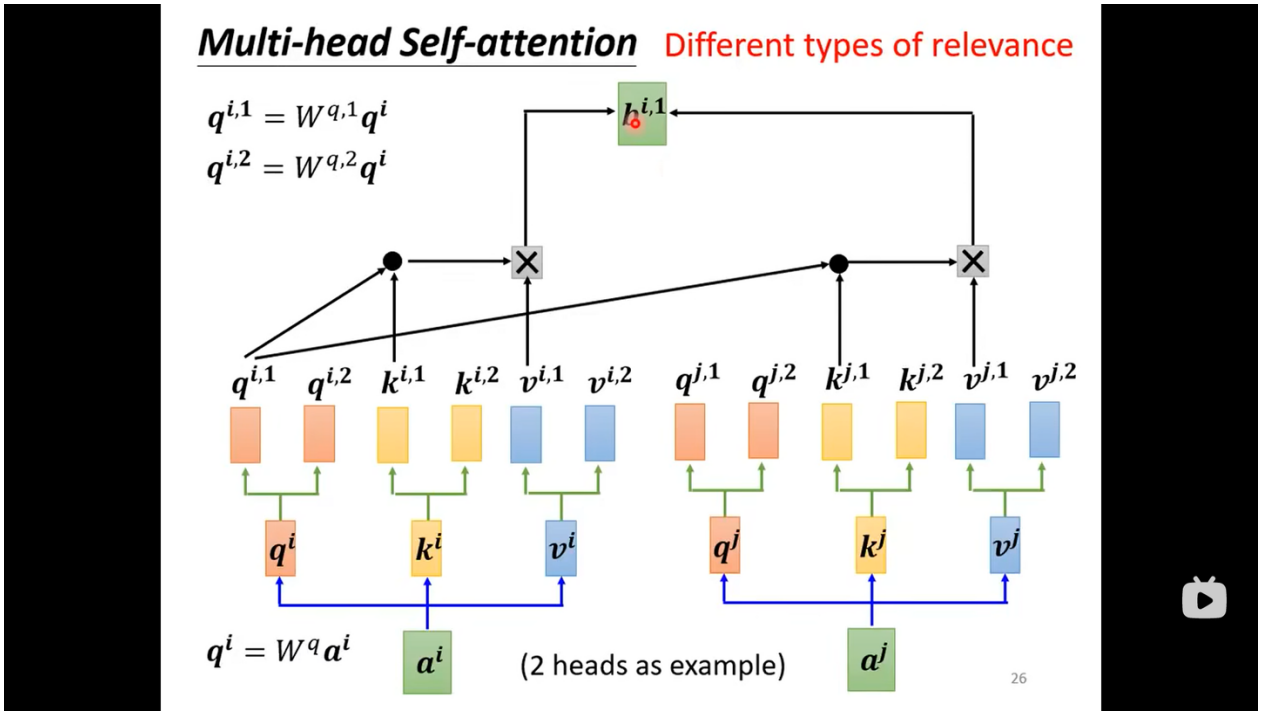
之前解决序列建模以及转导问题的时候都用的是rnn或者cnn，它们都有自己的缺陷，cnn不能够考虑到时序信息，rnn在长文本中的处理能力有欠缺，而transformer是一种只用到了注意力机制的模型，很好的解决了机器翻译的问题，现在也在对各方面有了推广。

2.模型的架构



Encoder

1.输入进来的文本通过词嵌入将词转化为相应的向量使其能够捕获单词的语义以及上下文的信息。
2.Positional Encoding:由于输入进去的句子模型不知道它们的位置信息，所以加入它们的位置编码。
3.Multi-Head Attention：其是由几个单头组合而成的



上图是二头注意力机制的图，几个头加一起的维度是512，刚好与一头注意力一致，但效果要比一头好。
4.Add&Norm：add是让输出的值跟输入的值相加（学习的是残差连接的方法），然后在对其进行归一化，这里的归一化用的是layernorm（cnn用的是batchnorm）
5.feed forward：前馈神经网络（MLP），就是两个线性层加在一起。第一层是y1 = W1 *x* + *b1，经过ReLU激活函数后，输出为：z = max(0, y1)，再到第二层y2 = W2* z + b2。

Decoder

1.output embedding：是目标序列（或解码序列）的词嵌入
2.Masked multi-Head Attention：与编码器唯一的区别是其要掩盖后面还未知晓的文字，此做法是让后面还没预测出的文字取负无穷，然后softmax就会取0，让后面的词失效。
3.Multi-Head Attention：这是编码器给其输入信息的关键，例如hello，world翻译成你好，世界，当解码器输出‘好’的时候，编码器就会对hello后面的world更加的关注，从而达到很好的翻译效果
其他的跟解码器差不多

3.为什么提出注意力机制

1.处理长序列问题：RNN在处理长序列时，容易出现梯度消失或梯度爆炸的问题，这使得它难以捕获长距离依赖关系。而注意力机制可以允许模型直接访问输入序列中的任意位置，从而更好地处理长序列。

2.并行计算能力：CNN和RNN都是顺序计算的，即计算下一个时刻的输出需要等待上一个时刻的计算结果。而注意力机制可以并行计算，因为它是一种基于权重的计算方式，这有助于提高模型的计算效率。
3.解释性：注意力机制可以为模型提供一定的解释性，因为它可以展示出模型在处理输入序列时，哪些部分受到了更多的关注。这种解释性在很多应用中都是很重要的。
4.增强模型的表达能力：注意力机制可以允许模型对输入序列进行有选择性的关注，这有助于模型捕捉到更多的上下文信息，从而增强模型的表达能力。

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

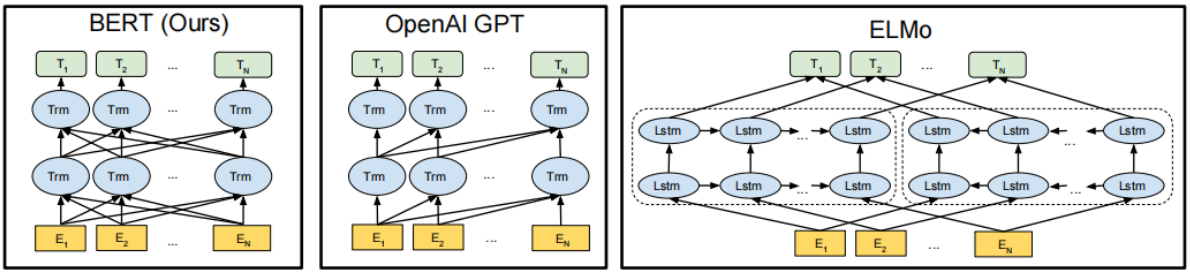
1.提出的原因，受到的启发以及bert特点

（1）原因：之前没有一个大型的自然语言处理框架可以处理大部分的nlp
（2）受到了elmo以及transformer的启发.
elmo是一种基于特征的方法(feature-base)，通过与文本本身的向量特征相容，得到更好的词嵌入向量特征作为输入，以便更好的进行特定的模型得到更好的输出，ELMo能够为不同的下游任务产生更丰富的输入表示，因为它可以捕捉不同任务所需的上下文信息。通过将ELMo生成的词嵌入与下游任务的模型结合使用，可以改进不同任务的性能，例如情感分析、命名实体识别、句法分析等。elmo方法的双向性给了bert启发，但elmo的基础架构是rnn。
transformer中的多头注意力机制对bert产生了启发，与双向性结合得到了bert模型
（3）基于微调的方法设计的，可以用在大多的下游任务中并且效果比之前的好一些。

2.bert预训练模型的架构。

（1）输入：其接受的输入是一整段文本，文本可以是一句话或者是几句话，为了让bert能够识别是第几句话，在进行编码时会加入【CLS】【SEP】,以及加入每段句子的位置信息，以及每个单词它们的位置信息，目的是让其更好的让之后的模型理解上下文信息.
（2）Pre-training BERT：Masked LM其是选择了15%的词进行了mask，然后让其预测，这样的目的是让模型能够理解单个句子之间的前后关系。其中15%的mask中又有一部分不进行掩码，一部分换了一个词，其目的是让模型有一定的泛化能力，微调的时候没有mask，所以你要是预训练都用mask，会让微调的效果得到提升。Next Sentence Prediction：随机抽两个句子，50%的概率是连续的，50%概率不是连续的，这会让bert了解不同句子之间的关系。
（3）Fine-tuning BERT：用自己的数据输入进入bert模型，以此来调整模型参数，这种方式被称为微调。

3.bert elmo gpt之间的比较



bert是双向的tranformer，可以了解文本前后的关系，GPT是单向的transformer，是从左向右进行，但其有解码器，所以可以用于交互式对话，elmo是双向lstm。

RoBERTa_ A Robustly Optimized BERT Pretraining Approach

自回归（autoregressive）：自回归模型基本思想是使用先前的数据点来预测序列中的下一个数据点。这种方法通常用于生成文本、预测时间序列数据、语音合成等任务，比如rnn，transformer就是使用了自回归，自回归在 NLP 中是一种序列建模方法。

1.与bert之间的区别

（1）它与bert的框架基本一样，在训练数据的大小，训练方法以及模型的参数和训练时间上有所差异。
（2）使用了动态掩码的操作，其目的是为了防止静态掩码时每次都选择相同的mask。这个改动在大的数据集上显得非常有用。（静态掩码在整个训练过程中保持不变，因此模型对于掩码的位置和方式是固定的；动态掩码是根据输入示例的特点而变化的，因此可以更灵活地适应不同的数据分布和任务）
（3）模型的输入形式以及是否去掉下一句预测（NSP）操作，文中使用单个句子，一段文本，是否使用NSP做了拼接，得到使用单个句子使下游任务效果下降，去掉NSP优于不去。
（4）大批量训练。大批量训练提高了掩模语言建模目标的困惑度，并提高了任务结束精度
（5）使用了字节对编码(BPE)，使得其减小了词汇的大小，还能够更好的处理多语言以及处理稀有词语。

2.训练结果特点以及比较

（1）更大的数据以及更大的预训练步数使得模型的效果越来越好。（(16GB→160GB文本)，(100K→300K→500K步)）
（2）在 GLUE, SQuaD and RACE基准上都有不错的表现。（GLUE 是一个集合了多个 NLP 任务的评估基准，包括文本分类、自然语言推理、情感分析等。它的目标是衡量模型的通用语言理解能力，以便进行任务无关的性能比较，GLUE里面有包括了MNLI QNLI QQP RTE SST MRPC CoLA STS WNLI等，对每个任务实验都有不一样的微调方法来与之前的实验分数进行比较；SQuAD 是一个问答任务的数据集，其中包含了一系列文章和与这些文章相关的问题。

模型的任务是从文章中提取出正确的答案。SQuAD 评估了模型在文本理解和问题回答方面的能力；RACE 是一个阅读理解任务的数据集，特别针对英语学习者。学生需要阅读一篇文章，然后回答一系列与文章相关的问题。这有助于评估模型在阅读理解和问题回答方面的性能）在GLUE中无需进行多任务微调以及SQuAD无需提供更多的数据也能得到好的效果。

DeBERTa Decoding-enhanced BERT with Disentangled

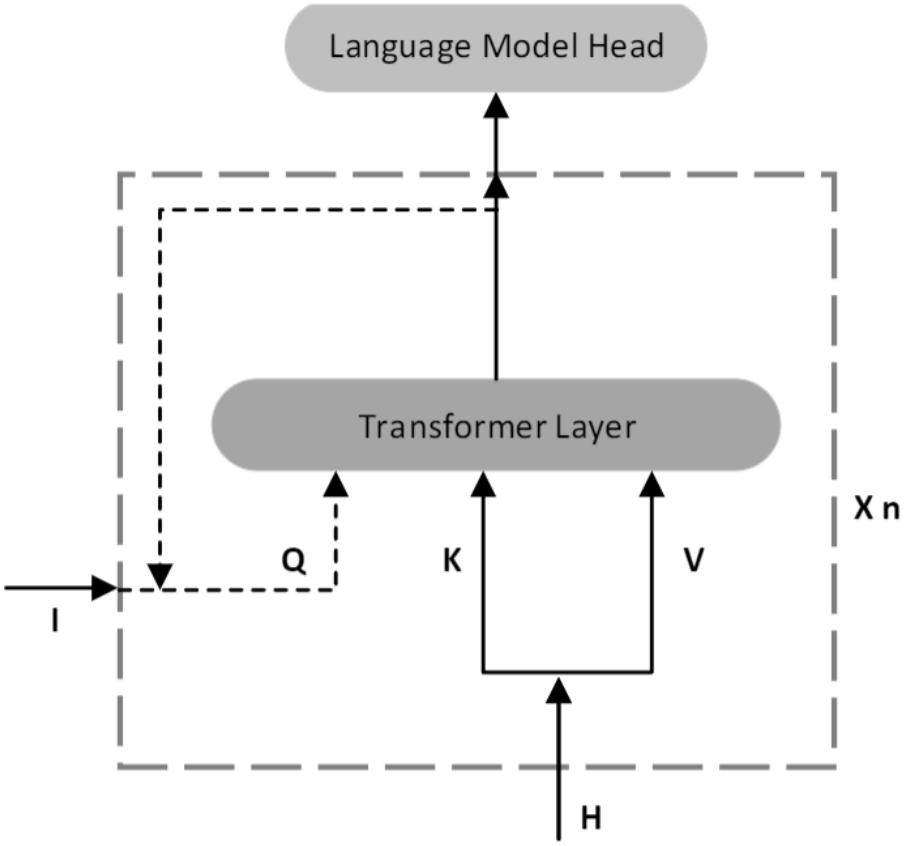
1.与bert不同之处

（1）引入分离的注意力机制，每个单词使用两个向量来表示其内容和位置，并使用分离的矩阵计算单词之间的注意力权重（BERT模型确实引入了位置信息，使用了位置嵌入（Positional Embeddings）来处理词语的位置信息。然而，BERT的位置嵌入并没有显式地建模单词之间的相对位置关系，而是简单地将位置信息加到词向量中，DeBERTa之所以创新引入相对位置信息，是因为相对位置编码可以更精细地捕捉单词之间的距离关系。相对位置编码允许模型更好地理解不同单词之间的相对位置，有助于处理长距离依赖关系）

$$
\begin{gathered}
\boldsymbol{Q_c} = \boldsymbol{H}\boldsymbol{W_{q,c}}, \boldsymbol{K_c} = \boldsymbol{H}\boldsymbol{W_{k,c}}, \boldsymbol{V_c} = \boldsymbol{H}\boldsymbol{W_{v,c}}, \boldsymbol{Q_r} = \boldsymbol{P}\boldsymbol{W_{q,r}}, \boldsymbol{K_r} = \boldsymbol{P}\boldsymbol{W_{k,r}} \\
\tilde{A}_{i,j} = \underbrace{\boldsymbol{Q_i^c}\boldsymbol{K_j^{c\intercal}}}_{\text{(a) content-to-content}} + \underbrace{\boldsymbol{Q_i^c}\boldsymbol{K_{\delta(i,j)}^{r}}^{\intercal}}_{\text{(b) content-to-position}} + \underbrace{\boldsymbol{K_j^c}\boldsymbol{Q_{\delta(j,i)}^{r}}^{\intercal}}_{\text{(c) position-to-content}} \\
\boldsymbol{H_o} = \operatorname{softmax}(\frac{\tilde{\boldsymbol{A}}}{\sqrt{3d}})\boldsymbol{V_c}
\end{gathered}
$$

如图公式，有了位置，内容向量之后，其注意力机制的计算就成了内容到内容，位置到内容，内容到位置，其中位置到位置因为没有用就删掉了。

（2）在掩码处加入绝对位置信息。因为之前的bert只是考虑了相对位置，有可能导致预测的效果不够好，比如‘a new store opened beside the new mall’，store跟mall不一致但是前面都是new，机器很难知道学习到，所以要加入绝对位置。绝对位置的加入是在transformer解码完之后（指的是将文本编码为向量之后），进行softmax之前（预测掩码之前），及transformer解码完再加入绝对位置然后进行掩码的预测，这样可以比bert有更好的结果。其掩码的解码器用的也是transformer的结构，如下图所示：



(b) Enhanced Mask Decoder

如图，绝对位置是解码第一次I口出的，之后的I口出的是前一次的输出，最后就是softmax层进行输出。（虽然“增强型掩码解码器”（Enhanced Masked Language Model，EMLM）的引入涉及到在掩码语言模型（MLM）任务中的某种解码机制，但这并不等同于传统意义上的解码器。在传统的序列到序列任务中，解码器通常用于生成目标序列，而编码器用于处理输入序列。而在 DeBERTa 中，这种增强型机制主要涉及到对编码器的改进，以更好地处理掩码标记的上下文信息。模型的整体结构仍然以编码器为主，并没有一个独立的解码器组件，这里的softmax可以理解为解码，但不属于解码器）

（3）使用了一个名为Scale-invariant Fine-Tuning (SiFT)的虚拟对抗训练算法来实现的。SiFT算法通过将扰动应用于归一化的词嵌入向量来提高训练的稳定性，首先将微调数据的词嵌入向量进行归一化然后再加入扰动因素来提高模型的泛化能力。

2.结果

（1）扩展到15亿参数之后，其在SuperGLUE的性能领先于其他的模型。
（2）在GLUE, SQuaD and RACE领域比起之前的roberta也有了提升。

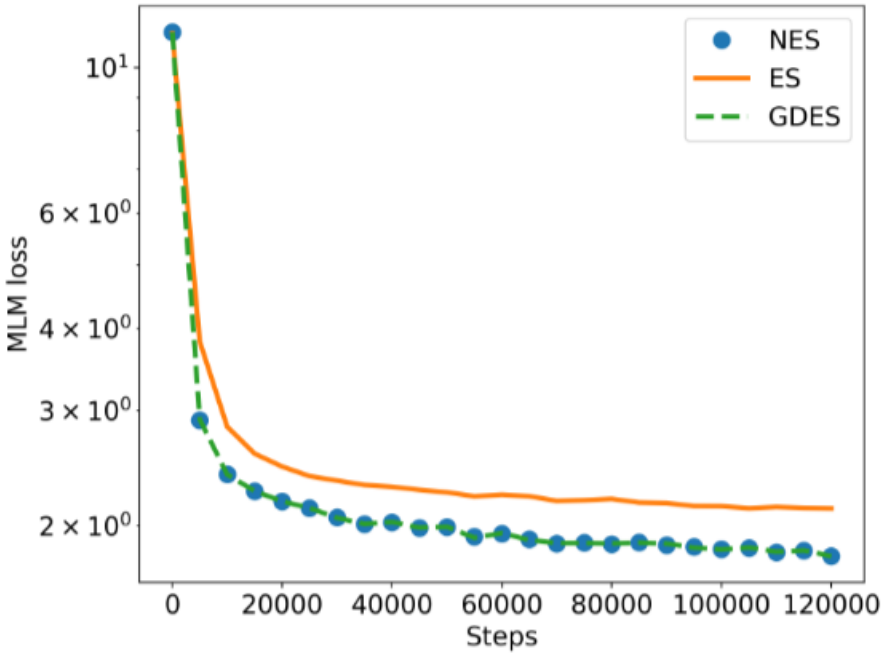
DeBERTaV3 Improving DeBERTa using ELECTRA-Style P

1.与deberta的区别以及deberta3的细节

（1）预训练的时候把MLM换成了RTD（replaced token detection），保留了deberta时相位位置编码的优点。（RTD是跟对抗神经网络类似，有一个生成器，一个鉴别器，其中生成器是通过MLM训练-->用于生成替代输入序列中被掩盖的标记;鉴别器是通过二元分类目标来训练的-->是判断相应的标记是原始标记还是由生成器替换的标记）。RTD是由ELECTRA提出了一种新的预训练范式，其是同时训练两个编码器。

（2）RTD中涉及到是否共享嵌入向量，共享嵌入的话会导致鉴别器和生成器的训练损失将令牌嵌入拉向相反的方向，会导致收敛比较慢，但是不共享嵌入会导致RTD的性能会比共享差一些但收敛快，为了利用两者的优势，文中设计了gradient-disentangled embedding sharing (GDES)，其设计的用处就是生成器和鉴别器共享嵌入向量，GDES不允许RTD损失影响生成器的梯度，**从而避免了由于目标冲突而导致的干扰和低效问题，停止梯度传播来防止鉴别器更新生成器的嵌入**,GDES只使用MLM损失来更新生成器的嵌入。

（3）文章通过比较NES和ES的性能以及收敛速度得到了字符嵌入共享有优势以及两个编码器的训练损失是相反的会降低收敛速度。如图：



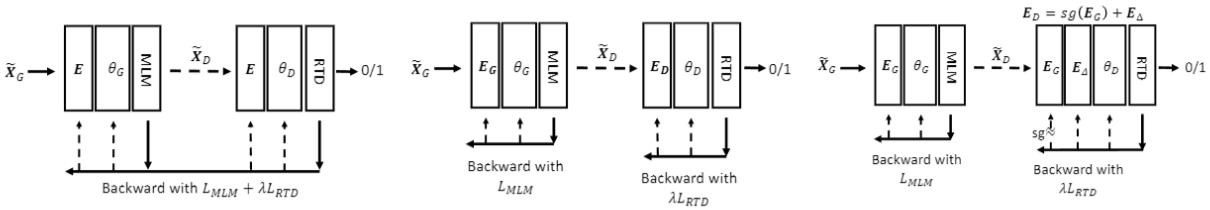


Figure 1: Illustration of different embedding sharing methods. (a) ES: $\boldsymbol{E}$, θ_G and θ_D will be jointly updated in a single backward pass with regards to $L_{MLM} + \lambda L_{RTD}$. (b) NES: $\boldsymbol{E}_G$ and θ_G will first be updated via the backward pass with regards to L_{MLM}, then $\boldsymbol{E}_D$ and θ_D will be updated via the backward pass with regards to λL_{RTD} . (c) GDES: $\boldsymbol{E}_G$ and θ_G will first be updated in the backward pass with regards to L_{MLM}, then $\boldsymbol{E}_\Delta$ and θ_D will be updated via the backward pass with regards to λL_{RTD} and $\boldsymbol{E}_G$. sg is the stop gradient operator that prevents the discriminator from updating $\boldsymbol{E}_G$

sg是阻止鉴别器更新EG的停止梯度操作符.EG是生成器的标记嵌入，ED是鉴别器的标记嵌

入。

2.结论

在小参数以及大参数的情况下都有很好的效果。

LoRA Low-Rank Adaptation of Large Language Models

1.引入的原因

（1）模型越来越大，其预训练的参数也越来越大，微调的参数越来越多，导致成本费用变得贵，所以我们引入一个微调模型来解决这一个问题。
（2）之前也有人在这一方面展开过研究但是他们有推理延迟（扩展模型深度或减少模型的可用序列长度），这些模型最关键的还是无法匹配基准基线，本文提出的LoRA很好的解决了这一点。（在之前的工作中，引入了适配器模型。适配器模型是一种参数和计算效率更高的方法，用于模型的自适应。它通过在Transformer块中添加适配器层或优化输入层激活的某些形式来实现。然而，适配器模型也存在一些限制，特别是在大规模和延迟敏感的生产场景中。适配器层会引入推理延迟，并且在在线推理设置中可能会导致显著的延迟增加。另一种方法是直接优化提示，但这种方法很难进行优化，并且在可训练参数的数量上表现不稳定）

2.LoRA的构造

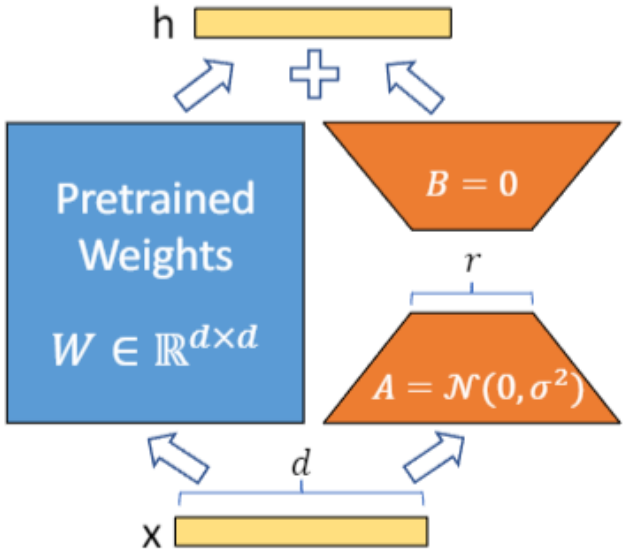


Figure 1: Our reparametrization. We only train A and B.

如图，其是通过把预训练模型的参数给冻结，然后使用几个小LoRA模型来对模型进行微调，小模型矩阵使用的秩r比原始的要小。

其模型运行过程的细节是预训练模型的权重参数维度用全连接层映射到训练层，训练层是在每个transformer块上注入进去的，然后让训练层进行训练，等所有的transformer块都训练好之后再用全连接层进行映射让其维度与预训练模型的权重参数维度一样，最后两者结合起来预测下游任务的输出。（公式是W = W0 + BA W0是预训练模型的权重，B和A都是可以训练的权重，训练过程中A和B发生变化，W0不变）

3.LoRA优势

（1）预训练的模型可以共享，并用于为不同的任务构建许多小型LoRA模块
（2）当使用自适应优化器时，LoRA使训练更有效，并将硬件进入障碍降低3倍，因为不需要计算梯度或维护大多数参数的优化器状态。相反，只优化注入的小得多的低秩矩阵
（3）简单的线性设计允许在部署时将可训练矩阵与冻结权重合并，与完全微调的模型相比，通过构造，不会引入推理延迟

4.实验部分

（1）模型讨论了选择transformer里面的哪些权重矩阵进行修改效果会比较好。（文章主要关注了自注意模块中的权重矩阵）
（2）讨论了选了低秩更新权重矩阵中秩的选择。