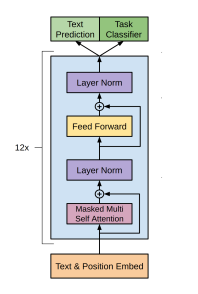
# GPT介绍

OpenAI公式在2018年提出了一种生成式预训练(Generative Pre-Trainging,GPT)模型用来提升自然语言理解任务的效果，正式将自然语言处理带入预训练时代，预训练时代意味着利用更大规模的文本数据一级更深层次的神经网络模型学习更丰富的文本语义表示。同时，GPT的出现提出了“”生成式预训练+判别式任务精调的自然语言处理新范式，使得自然语言处理模型的搭建变得不在复杂。

- 生成式预训练：在大规模文本数据上训练一个高容量的语言模型，从而学习更加丰富的上下文信息；

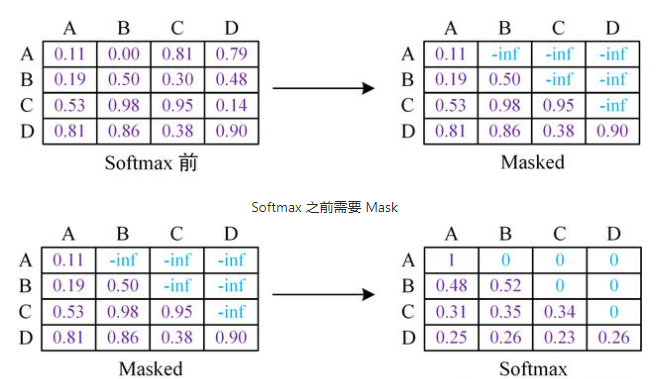
- 判别式任务精调：将预训练好的模型适配到下游任务中，并使用有标注的数据学习判别式任务。

GPT的整体结构是一个基于Transformer的单向语言模型，即从左到右对输入文本建模，模型结构如图所示。



GPT只使用了Transformer的Decoder结构，由于没有了Encoder，就没有Encoder的输出，所以去掉了原本的Encoder-Decoder Attention。

GPT模型使用Transformer的Decoder结构，在Decoder self-attention结构中使用了mask机制，将自注意力矩阵的上三角mask掉，每个单词只能获取它本身以及它之前词的注意力，防止模型看到未来时刻信息，因为语言模型就是要预测未来时刻单词的，让其看到未来时刻信息相当于作弊了，模型学不到任何东西。如图所示，将上三角的值设置为负无穷，经过softmax计算后就会变成0，每个词只能注意到当前词以及之前词。例如图中，A这个词这一行，只有第一列有值，它只能注意自身，B这个词这一行，它前两列有值，说明B只能注意到A以及B，注意不到C和D，因为模型需要根据AB去预测C，所以不让模型注意到C以及更靠后的信息。



## 无监督预训练

GPT利用常规语言建模的方法优化给定文本序列$x=x\_{1},x\_{2},...,x\_{n}$的最大似然估计。

式中k表示语言模型的窗口大小，即基于k个历史词预测当前时刻的词,表示神经网络的参数。

对于长度为k的窗口词序列通过以下方式计算建模概率P。

式中，表示 的独热向量表示；表示词向量矩阵，表示位置向量矩阵，L表示总层数。

现在模型一般不取k个历史词，而是取所有历史词。

## 有监督下游任务微调

在预训练阶段，GPT利用大规模数据训练出基于深层Transformer的语言模型，已经掌握了文本的通用语义表示。精调（fine-tuning）的目的在通用语义的表示基础上，根据下游任务（Downstream task）的特征进行领域适配，使之与下游任务的形式更加契合，以获得更好的下游任务应用效果。

下游任务精调通常是由有标签数据进行训练和优化的。假设下游任务的标注数据为C，其中每个样例的输入是构成的长度为n的文本序列，与之对应的标签为y。首先将文本序列输入预训练的GPT中，获得最后一层最后一个词对应的隐含层输出,紧接着，将该隐含层输出通过一层全连接层，预测最终标签y。

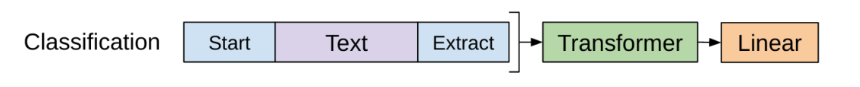
其中，表示全连接层权重，c为类别个数。

最终，通过优化以下损失函数精调下游任务。

另外，为了进一步提升精调后模型的通用性以及收敛速度，可以在下游任务精调时加入一定权重的预训练任务损失。这样做是为了缓解在下游任务精调时出现灾难性遗忘问题。因为在下游任务精调过程中，GPT的训目标是优化下游任务数据上的效果，更强调特殊性。因此势必会对预训练阶段学习的通用知识产生部分的覆盖或者擦除，丢失一定的通用性。通过结合下游任务精调损失和预训练任务损失，可以有效缓解灾难性遗忘问题，在优化下游任务效果的同时保留一定的通用性。在实际应用中，可通过下式精调下游任务。

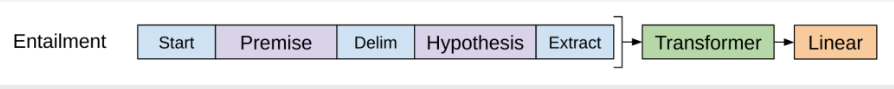
## 适配不同的下游任务

**分类任务**



对文本开头加一个开始标识符，结尾加上一个抽取标识符，将其输入Transformer解码器，最后的这个抽取标识符得到的结果在经过一个全连接层即可进行分类，使用最后一个标识符的原因是最后一个标识符可以抽取前面所有词的信息。

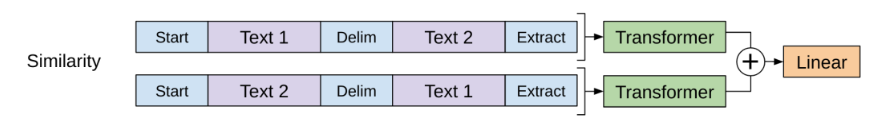
**蕴含任务**



给一段话，然后给一个假设，判断这段话有没有蕴含假设提出的内容。

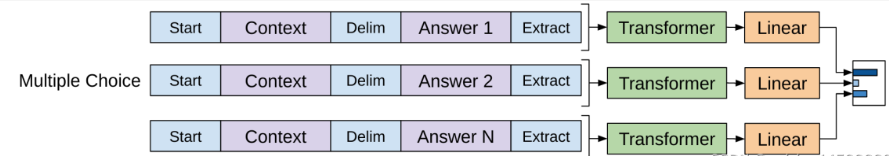
比如假设是a喜欢b，给出的这段话是a喜欢b，那么这段话是支持这个假设的，如果给出这段话是a讨厌b，那么这段话不支持这个假设，如果给出的这段话是a和b是邻居，那么既不支持也不反对，其实本质也是一个三分类任务。将两句话拼接起来，最开始加上开始符，中间加上分隔符，最后加上抽取符。

**相似任务**



判断两段话是否相似，也是需要将两段话拼接起来，前边加上开始符，中间分隔符，最后抽取符。相似是一个对称问题，a与b相似，那么b与a也是相似的，所以需要将两句话交换个位置，构造两个序列输入模型，将两个结果相加经过全连接层进行分类。

**选择题**

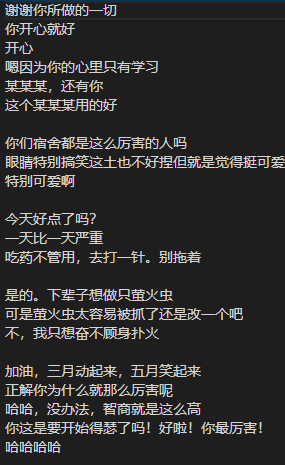


给定一个问题，以及对应的几个答案，让模型选出正确答案，有N个答案就构造如图所示的N个序列，分别输入模型，得到N个结果再分别输入N个全连接层，全连接层输出大小为1，这N个输出中，最大的我们认为是正确答案。

# 实验部分

## 2.1 数据集

使用了一个五十万条数据的数据集，数据集地址：<https://drive.google.com/file/d/1nEuew_KNpTMbyy7BO4c8bXMXN351RCPp/view?usp=sharing>



多轮对话数据集。五十万轮对话。

## 2.模型

使用huggingface上开源的中文GPT-2模型，使用100G纯文本预训练后的模型。模型地址<https://huggingface.co/uer/gpt2-chinese-cluecorpussmall>

## 2.3 方法介绍

GPT模型是一个语言模型，只能从前往后一个字符一个字符进行预测，怎么让它完成对话任务呢？

其中一个方法就是将每轮对话拼接起来，中间使用一个[SEP]符号，例如刚才图中的第一轮对话的例子，拼接起来后就变成这样，“谢谢你所做的一切[SEP]你开心就好[SEP]开心[SEP]嗯因为你的心里只有学习[SEP]某某某，还有你[SEP]这个某某某用的好”，将这一轮对话看作是一条样本，中间使用[SEP]拼接起来，以语言模型目标继续微调GPT模型，模型就可以基于前边已知的信息预测后边一个字符。

在推理阶段，输入一句话，假如是,"你在干嘛？"，然后在后面拼接上一个[SEP]输入到模型中，让模型预测下一个字符，比如模型预测出一个"我"，然后再将这个"我"字拼接到原来的基础上，得到“你在干嘛？[SEP]我”，然后让模型再预测下一个字符，依此类推，直到模型预测出[SEP]，预测出来的这部分作为回答，比如模型依次预测出了“我在玩呢”，将"我在玩呢"作为这句话的回答，如果再次提问,"你在玩什么呢？"，这时候需要将历史信息全部拼接起来，最后再加一个[SEP,]就像这样,"你在干嘛？[SEP]我在玩呢[SEP]你在玩什么呢？[SEP]"，然后再让模型依次预测下一个字符，直到预测出[SEP]，这样就可以实现多轮对话的功能，传统的基于seq2seq的对话模型，无法记住上下文信息，他的回答只能通过你当前的问题生成。刚才提到的这种方式，其可以记住上下文，基于上下文进行回答，chatgpt也是基于这种原理记住上下文的。