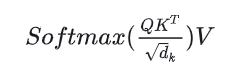
# **大模型面试八股**

一bert篇

**1.transformer 八股文**

这一部分包包给出了答案，在你十分清楚transformer结构后，可以加强这些题目来强化对于transformer的理解。当然如果你是一知半解，也可以读来做一个强化学习。

1. **Self-Attention的表达式**



**b.为什么上面那个公式要对QK进行scaling**

scaling后进行softmax操作可以使得输入的数据的分布变得更好，你可以想象下softmax的公式，数值会进入敏感区间，防止梯度消失，让模型能够更容易训练。

**c.self-attention一定要这样表达吗？**

不一定，只要可以建模相关性就可以。

当然，最好是能够高速计算（矩阵乘法），并且表达能力强（query可以主动去关注到其他的key并在value上进行强化，并且忽略不相关的其他部分），模型容量够（引入了project\_q/k/v，att\_out，多头）。

**d.有其他方法不用除根号**  **吗？**

有，只要能缓解梯度消失的问题就可以。详情可以了解Google T5的Xavier初始化。

**e.为什么transformer用Layer Norm？有什么用？**

任何norm的意义都是为了让使用norm的网络的输入的数据分布变得更好，也就是转换为标准正态分布，数值进入敏感度区间，以减缓梯度消失，从而更容易训练。当然，这也意味着舍弃了除此维度之外其他维度的其他信息。为什么能舍弃呢？请看下一题。

**f.为什么不用BN？**

首先要明确，如果在一个维度内进行normalization，那么在这个维度内，相对大小有意义的，是可以比较的；但是在normalization后的不同的维度之间，相对大小这是没有意义的

BN(batch normalization)广泛应用于CV，针对同一特征，以跨样本的方式开展归一化，也就是对不同样本的同一channel间的所有像素值进行归一化，因此不会破坏不同样本同一特征之间的关系，毕竟“减均值，除标准差”只是一个平移加缩放的线性操作。在“身高体重”的例子中，这就意味着“归一化前是高个儿的归一化后仍然是高个儿，归一化前胖的归一化后也不会变瘦”。这一性质进而决定了经过归一化操作后，样本之间仍然具有可比较性。但是，特征与特征之间的不再具有可比较性，也就是上一个问题中我所说的“舍弃了除此维度之外其他维度的其他信息”。

既然前面说了是CV中用BN，那为什么NLP中不用BN，而用LN呢？道理一样，因为NLP中：

* 对不同样本同一特征的信息进行归一化没有意义：
  + 三个样本（为中华之崛起而读书；我爱中国；母爱最伟大）中，“为”、“我”、“母”归一到同一分布没有意义。
* 舍弃不了BN中舍弃的其他维度的信息，也就是同一个样本的不同维度的信息：
  + “为”、“我”、“母”归一到同一分布后，第一句话中的“为”和“中”就没有可比性了，何谈同一句子之间的注意力机制？

加强一下，我们再回顾CV中：

* 对不同样本同一特征（channel）的信息进行归一化有意义：
  + 因为同一个channel下的所有信息都是遵循统一规则下的大小比较的，比如黑白图中越白越靠近255，反之越黑越靠近0
* 可以舍弃其他维度的信息，也就是同一个样本的不同维度间（channel）的信息：
  + 举例来说，RGB三个通道之间互相比较意义不大

**g.Bert为什么要搞一个position embedding？**

因为仅仅有之前提到的self-attention无法表达位置信息（对位置信息不敏感），比如说“1+1=2”和“1+2=1”是一样的，因此需要增强模型针对位置的表达能力。

**h.Bert为什么三个embedding可以相加？**

这里的三个embedding是指token embedding，segment embedding，position embedding。如果感兴趣，还是来看[Rethinking Positional Encoding in Language Pre-training](https://link.zhihu.com/?target=https://arxiv.org/abs/2006.15595" \t "https://zhuanlan.zhihu.com/p/_blank)原文，不过为了理解也可以看下邱老师的回答：

[为什么 Bert 的三个 Embedding 可以进行相加？](https://www.zhihu.com/question/374835153/answer/1080315948)

如果你是在质疑加法会导致”信息损失“，但是本质上神经网络中每个神经元收到的信号也是“权重”相加得来。

详细想想，在实际场景中，叠加是一个更为常态的操作。比如声音、图像等信号。一个时序的波可以用多个不同频率的正弦波叠加来表示。只要叠加的波的频率不同，我们就可以通过傅里叶变换进行逆向转换。

一串文本也可以看作是一些时序信号，也可以有很多信号进行叠加，只要频率不同，都可以在后面的复杂神经网络中得到解耦（但也不一定真的要得到解耦）。在BERT这个设定中，token，segment，position明显可以对应三种非常不同的频率。

由此可以再深入想一想，在一串文本中，如果每个词的特征都可以用叠加波来表示，整个序列又可以进一步叠加。哪些是低频信号（比如词性？），哪些是高频信号（比如语义？），这些都隐藏在embedding中，也可能已经解耦在不同维度中了。说不定可以是一种新的表示理论

**i.transformer为什么要用三个不一样的QKV？**

前面提到过，是为了增强网络的容量和表达能力。更极端点，如果完全不要project\_q/k/v，就是输入x本身来做，当然可以，但是表征能力太弱了（x的参数更新得至少会很拧巴）

**j.为什么要多头？举例说明多头相比单头注意力的优势**

和上一问一样，进一步增强网络的容量和表达能力。你可以类比CV中的不同的channel（不同卷积核）会关注不同的信息，事实上不同的头也会关注不同的信息。

假设我们有一个句子"the cat, which is black, sat on the mat"。在处理"sat"这个词时，一个头（主语头）可能会更关注"cat"，因为"cat"是"sat"的主语；另一个头（宾语头）可能会更关注"on the mat"，因为这是"sat"的宾语；还有一个头（修饰头）可能会关注"which is black"，因为这是对"cat"的修饰。

当然，这只是为了方便你理解，事实上就和卷积核一样，不同头关注的内容是很抽象的。

你当然可以就用一个头同时做这个事，但是还是这个道理，我们的目的就是通过增加参数量来增强网络的容量从而提升网络表达能力。

经过多头之后，我们还需要att\_out线性层来做线性变换，以自动决定（通过训练）对每个头的输出赋予多大的权重，从而在最终的输出中强调一些头的信息，而忽视其他头的信息。这是一种自适应的、数据驱动的方式来组合不同头的信息。

**k.为什么Bert中要用WordPiece/BPE这样的subword Token？**

避免OOV（Out Of Vocabulary），也就是词汇表外的词。在NLP中，通常会预先构建一个词汇表，包含所有模型能够识别的词。然而，总会有一些词没有出现在预先构建的词汇表中，这些词就是 OOV。

传统的处理方式往往是将这些 OOV 映射到一个特殊的符号，如 <UNK>，但这种方式无法充分利用 OOV 中的信息。例如，对于词汇表中没有的词 "unhappiness"，如果直接映射为 <UNK>，则模型就无法理解它的含义。

WordPiece/Byte Pair Encoding (BPE) 等基于子词的分词方法提供了一种解决 OOV 问题的方式。现在更多的语言大模型选择基于BPE的方式，只不过BERT时代更多还是WordPiece。BPE 通过将词分解为更小的单元（子词或字符），可以有效地处理词汇表外的词。对于上面的 "unhappiness" 例子，即使 "unhappiness" 本身不在词汇表中，但是它可以被分解为 "un"、"happiness" 等子词，而这些子词可能在词汇表中。这样，模型就可以通过这些子词来理解 "unhappiness" 的含义。

另一方面就是，BPE本身的语义粒度也很合适，一个token不会太大，也不会小到损失连接信息（如一个字母）。

**l.Bert中为什么要在开头加个[CLS]?**

[sliderSun：关于BERT中的那些为什么](https://zhuanlan.zhihu.com/p/360343071)

其实这个回答也写了一些为什么，其中就包含这个题目。为了文章的完整性我再输出一点自己的观点。

具体来说，我们想让[CLS]做的事情就是利用好BERT强大的表示能力，这个表示能力不仅限于token层面，而且我们尝试要得到整个seqence的表示。因此，[CLS]就是做这个事情的。具体来说，整个encoder的最后一层的[CLS]学到的向量可以很好地作为整句话的语义表示，从而适配一些setence层面的任务，如整句话的情感分类。

那关键点就在于，为什么[CLS]可以建模整句话的语义表征呢？简单来说也很好理解，因为“这个无明显语义信息的符号会更“公平”地融合文本中各个词的语义信息，从而更好的表示整句话的语义。”

——为什么无明显语义？因为训练的时候BERT发现每个句子头都有，这样他能学到什么语义呢？

——为什么要公平？因为控制变量，我们不希望做其他下游任务的时候用于区分不同样本间特征的信息是有偏的。

当然，不放在句子开头的其他位置是否可行？一个未经考证的臆想是，任何其他位置的position embedding都无法满足放在开头的一致性。所以不同句子间可能会有一定的不同，这并不是我们做一些句间的分类问题想要的。

**m.不用[CLS]的语义输出，有其他方式可以代替吗？**

这个问题还是考察到了[CLS]的核心内涵，也就是如何获得整个sentence的语义表示。既然不让使用特意训好的[CLS]，那我们就从每个token得到的embedding入手，把所有的token弄到一起。

很直观的思路，就是对BERT的所有输出词向量（忽略[CLS]和[SEP]）应用MaxPooling和AvgPooling，然后将得到的两个向量拼接起来，作为整个序列的表示。这样做的话可以同时保留序列中最显著的特征（通过MaxPooling）和整体的，均衡的特征（通过AvgPooling）。

当然这种做法我本人并没有尝试过，或许也不是一个很好做的研究/工作方向。

**n.Bert中有哪些地方用到了mask?**

预训练任务Masked Language Model (MLM)

self-attention的计算

下游任务的decoder

**o.预训练阶段的mask有什么用？**

虽然MLM现在被主流LLM抛弃了，但是也是一项很重要的任务。

主要的思想是，把输入的其中一部分词汇随机掩盖，模型的目标是预测这些掩盖词汇。这种训练方式使得每个位置的BERT都能学习到其上下文的信息。

**p.attention中的mask有什么用？（BERT中）**

这是nlp任务很重要的问题，就是不同样本的seq\_len不一样。但是由于输出的seq\_len需要一致，所以需要通过补padding来对齐。而在attention中我们不希望一个token去注意到这些padding的部分，因为实际场景下它们是不存在的，所以attention中的mask就是来处理掉这些无效的信息的。

具体来说就是在softmax前每个都设为-inf（或者实际的场景一个很小的数就可以），然后过完softmax后"padding"部分的权重就会接近于零，query token就不会分配注意力权重了。

**t.Bert是如何处理传统方法难以搞定的溢出词表词(oov)的语义学习的？**

前面提到了，关键词是subword。

**u.中文是如何处理溢出词表词(oov)的语义学习的？**

subword处理中文都是字符级别的，所以就不会有词级别oov的问题了。

**x.为什么说GPT是单向的Bert是双向的？**

这也是decoder-only和encoder-only的区别。

decoder-only架构的生成模型在输出的时候只能看到当前位置前的tokens，也就是屏蔽了序列后面的位置，以适配NTP任务。

encoder-only架构的编码模型在输出的时候可以利用前后位置的tokens，以适配MLM任务。

具体的做法是self-attention加不加casual mask，也就是遮不遮住序列后面的内容。

**y.Bert如何处理一词多义？**

一词多义指的是在不同句子中token有不同的含义。

这正是self-attention解决的，搭配上MLM的任务，就可以让每个token会注意到上下文的其他token来得到自己的embedding。

**z.Bert中的transformer和原生的transformer有什么区别？**

其实很多，如果我们只讨论模型架构，也就是对比[Attention is All You Need](https://link.zhihu.com/?target=https://arxiv.org/abs/1706.03762" \t "https://zhuanlan.zhihu.com/p/_blank)的encoder和BERT的话，最重点的区别在于位置编码。

原生的transformer是最经典的Sinusoidal绝对位置编码。

而BERT中变成了可以学习的参数，也就是可学习位置编码。

变得可学了的话，只要模型学习能力强，数据量够，确实不会差。可以类比卷积核从手工变成了模型自己学。

关于位置编码，如果你有时间的话，建议从下面的链接一直往后看，苏神的内容质量都很高。位置编码确实大有可为，最近RoPE+NTK的方法来外推context length也挺让人热血沸腾的。  
[Transformer升级之路：1、Sinusoidal位置编码追根溯源 - 科学空间|Scientific Spaces](https://link.zhihu.com/?target=https://kexue.fm/archives/8231" \t "https://zhuanlan.zhihu.com/p/_blank)

**z+.Albert是通过什么方法压缩网络参数的？有什么问题？**

两个技巧，其一是参跨层数共享，其二是对嵌入参数化进行因式分解，也就是“不再将 one-hot 向量直接映射到大小为 H 的隐藏空间，先映射到一个低维词嵌入空间 E，然后再映射到隐藏空间”。

问题也是“模型压缩”通用的问题，网络表达能力和容量下降。然后推理速度也不会有很直观的提升。

**2.attention计算方式以及参数量，attention layer手写，必考。**

如果你找的工作是比较基础的，比如说本科生找llm相关实习，那基本会让你手写多头。

如果你想比较方便地一站对比各个Transformer模型的源码，可以来这个库：[GitHub - OpenBMB/ModelCenter](https://link.zhihu.com/?target=https://github.com/OpenBMB/ModelCenter" \t "https://zhuanlan.zhihu.com/p/_blank)

1. **Transformer模型的基本结构是什么?它是如何改变深度学习领域的?**

基本结构:Transformer模型由编码器和解码器组成，每个编码器包含多层自注意力和前馈网络，解码器增加了编码器-解码器注意力。模型中广泛使用了残差连接和层归一化。影响:Transformer引入了自注意力机制，使得模型能够并行处理序列数据，显著提高了长距离依赖项的处理能力，改变了序列建模和自然语言处理的主流方法。

1. **Transformer为何能够有效地处理长距离依赖问题?与传统RNN和LSTM相比有哪些优势?**

长距离依赖处理:Transformer通过自注意力机制直接计算序列中任意两点间的依赖关系，避免了RNN和LSTM中的逐步传播，因此能有效捕捉长距离依赖。

优势:相比RNN和LSTM，Transformer具有并行化处理的优势,缩短了训练时间。同时，

它避免了梯度消失问题，提高了对长序列的建模能力。

1. **多头注意力的作用是什么?**

作用:多头注意力允许模型同时从不同的表示子空间捕获信息，增强了模型对不同位置和语义信息的捕捉能力，提高了注意力机制的表达能力

1. 能不能手写下attention?



1. **Transformer模型如何平衡模型性能与计算资源的消耗?**

平衡策略:Transformer通过调整模型大小(层数、维度等)、使用注意力机制的有效变体(如稀疏注意力)和优化技术(如混合精度训练)来平衡性能和计算资源消耗

6、**Transformer模型的自注意力机制如何实现并行处理?**

在自注意力机制中，模型对输入序列的每个元素计算其与序列中所有其他元素之间的注意力得分。这一计算是矩阵乘法形式的，可以高效地利用现代硬件(如GPU或TPU)进行并行计算。这种并行化大大提升了模型的训练和推理速度，特别是对于长序列数据。

7、**在Transformer模型中，位置编码(Position Encoding)的作用是什么**

作用:位置编码赋予模型对输入序列中元素位置的感知能力，因为自注意力机制本身不

具备区分不同位置信息的能力。

1. **Transformer模型如何处理变长输入序列?**

处理方式:Transformer通过位置编码处理变长输入，配合掩码(masking)技术来处理不同长度的序列，确保模型在自注意力计算中只关注有效的输入部分。

1. **Transformer模型的缩放点积注意力(Scaled Dot-Product Attention)是什么，其重要性在哪里?**

定义:缩放点积注意力是一种计算注意力权重的方法，它通过对查询(Q)、键(K)的点积结果进行缩放，并应用softmax函数获取权重。

重要性:此机制允许模型在给定查询的情况下，动态地聚焦于关键的信息，缩放因子能避免在高维空间中点积结果过大，导致梯度消失问题。

1. **Transformer模型在实践中如何优化以处理超长序列?**

优化方法:针对超长序列，可以采用分块注意力、稀疏注意力、记忆压缩技术或者长序列专用的Transformer变体来降低计算复杂度

11、**Transformer模型在自注意力层中如何解决多尺度表示问题?**

解决方式:通过多头注意力设计，模型能够在不同的表示子空间中捕捉信息，从而同时考虑不同尺度的序列特征。

12.**Transformer模型中的自注意力机制在计算效率和表示能力之间是如何权衡的?**

自注意力机制通过并行处理序列数据提高计算效率，而多头注意力设计则提升了模型的表示能力。权衡通常通过调整头的数量和维度大小来实现。

1. **Transformer模型的参数共享策略对模型性能有何影响?**

影响:参数共享能减少模型参数量，避免过拟合，同时在多任务学习中可以提高模型的泛化能力。

14、**Transformer encoder和decoder的区别?**

区别:编码器负责处理输入序列，解码器则在此基础上增加了编码器-解码器注意力层，用于将编码器的输出与当前生成的序列相结合，进行序列生成。

1. **Transformer模型中的前馈网络(Feed-Forward Networks)的作用是什么?**

作用:前馈网络对自注意力层的输出进行非线性变换，增加了模型的表达能力，并可以捕捉局部特征。

1. Pre-norm和post-norm有什么区别?
2. **Transformer网络很深，是怎么避免过拟合问题的?**

Transformer网络采用以下机制以避免过拟合并促进深层结构的训练:

Dropout: 在自注意力、前馈层和嵌入层中随机抑制节点激活，提高泛化性。

权重衰减:引入L2正则化惩罚过大的权重参数，限制模型复杂度。

标签平滑:在损失函数中对真实标签分布进行平滑，避免模型对某些类别的过度自信。残差连接:通过跳跃连接，实现特征直传，缓解梯度消失问题并加速收敛。

1. Transformer的两个mask机制是什么?

两种掩码:

序列掩码:用于屏蔽输入序列中的填充

(padding)部分，确保这些位置不影响自注意力的计算。

查找掩码:用于解码器中防止未来信息泄露，

确保在预测下一个词时只能使用之前的词。

19、**Transformer为什么要用Layer norm?作用是什么**

层归一化(Layer

normalization)可以加速训练并提高稳定性，通过对输入的特征进行归一化，减少了不同初始化和批量数据分布差异带来的影响。

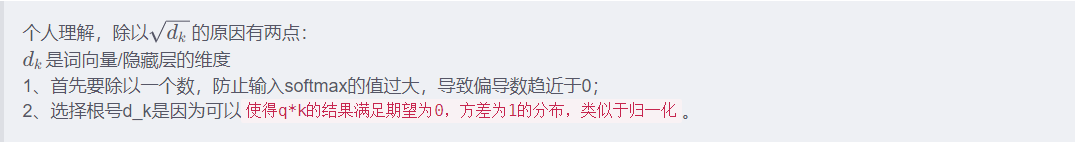
20、**Encoder和decoder是如何进行交互的?**

交互方式:在解码器中，编码器-解码器注意力层允许解码器的每个位置访问编码器的所有位置的输出。这种机制使解码器能够根据编码器的上下文信息生成输出序列。

进阶篇

注意力

1. 如何提高Transformer模型中自注意力机制的计算效率?
2. 在计算注意力分数时仅考虑部分词元，通过限制 Query-Key 对的数量，使计算复杂度与n呈线性关系，而非二次方关系。这类方法就称为稀疏注意力（Sparse Attention）机制。可以将稀疏化方法进一步分成两类：基于位置信息和基于内容。
3. Flash Attention 在绝大多数的神经网络中，都含有大量的Memory-bound操作，但是绝大多数Efficient Transformer把改进方法集中在降低模型的FLOPS上。这就导致这些方法的计算速度并没有显著降低。于是FlashAttention将优化重点放在了降低存储访问开销（MAC）上
4. 多查询注意力（Multi Query Attention）是一种多头注意力的变体，它在轻微牺牲模型质量的前提下显著减少计算成本。在多查询注意力中key-value对在不同的注意力头之间共享，即，所有注意力头使用同一个key投射和一个value投射，只单独保留了query。因此键和值的矩阵仅有一份，这大幅度减少了显存占用和解码所需的内存带宽需求。
5. 为什么self-attention要除以根号N?有方法不用处理根号N的吗?



有，只要能做到每层参数的梯度保持在训练敏感的范围内，不要太大，不要太小。那么这个网络就比较好训练。方式有，比较好的初始化方法，类似于google的T5模型，就在初始化把这个事情干了。

1. Transformer模型中注意力权重如何解释模型的决策?

这道题其实是在考察self-attention是如何实现

Transformer模型通过注意力权重，可以直观地理解模型在做决策时关注的区域，提高了模型的可解释性。

1. 如何在自注意力机制中平衡局部信息和全局信息的捕获?

这个其实是在考察自注意力的公式QKV如何计算，softmax

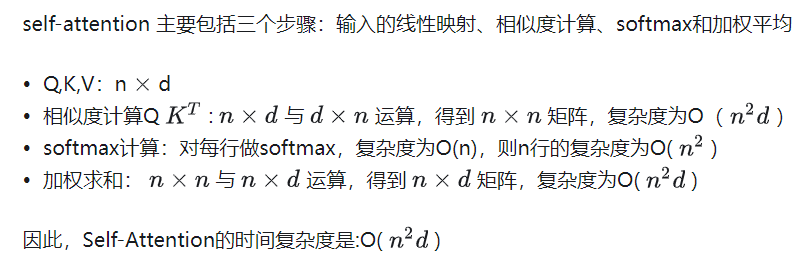
1. 基于attention有哪些代表性的改进方法?分别针对的是什么问题
2. 多头注意力（Multi-Head Attention）：在标准attention机制中，输入被缩放然后与权重相乘以产生输出。在多头attention中，输入首先被分为多个“头”，每个头独立计算attention权重，然后将结果拼接起来。这种方法可以使模型更好地理解和处理输入数据。
3. 自注意力（Self-Attention）：在许多任务中，输入数据的一部分与另一部分是高度相关的。自注意力机制让模型学习这种关系，从而提高性能。例如，在机器翻译任务中，句子中的单词可能会依赖于其他单词。通过让模型关注整个句子，而不是仅仅关注当前单词，可以提高翻译的准确性。
4. 局部注意力（Local Attention）：与全局注意力相反，局部注意力只关注输入的局部区域。这种方法可以减少计算量，并使模型更好地理解输入数据的结构。

4. 加权平均注意力（Scaled Dot-Product Attention with Optional Additional Heads）：在多头attention中，每个头的输出被缩放然后相加。加权平均注意力是对此方法的改进，它根据头的输出为每个头分配不同的权重。这可以进一步提高模型的性能。

26、如何设计更有效的注意力机制来处理层次化或结构化数据?

层次化注意力机制（Hierarchical Attention）：层次化注意力机制在处理具有层次结构的输入时非常有效，如文档分类、句子级情感分析等。它首先将输入划分为不同的层次（如句子、段落等），然后在每个层次上计算注意力得分，并将结果汇总到最终的输出中。层次化注意力机制能够让模型在处理复杂输入时更好地关注到关键信息。

1. self-attention时间复杂度



1. 逻辑回归为什么用交叉熵不用mse

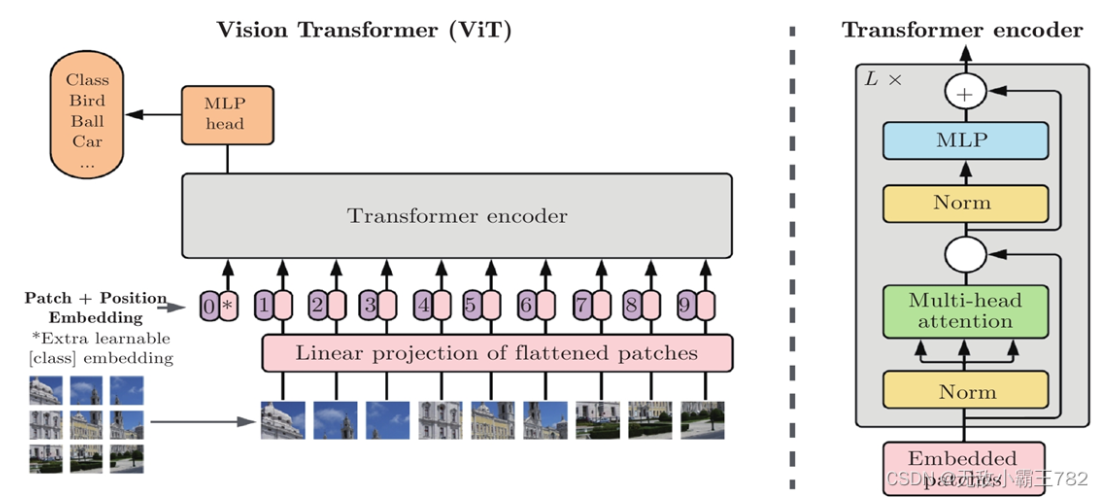
逻辑回归模型不用MSE作为损失函数的原因主要有两个，一个从背景意义来说，交叉熵函数更贴切“概率”的意义。另一方面是从求解的角度，交叉熵损失函数比MSE损失函数更易于求解。

1. 介绍一个transformer变体

ViT的首要意义就是将Transformer成功应用于计算机视觉任务，如图像分类、目标检测等。为研究人员提供了一种新的思路，即在计算机视觉任务中探索更多基于自注意力机制的模型设计

在分割任务中，ViT可以被用作特征提取器，将图像编码为特征向量，然后通过后续的分割头进行像素级别的预测，将图像中的每个像素分类到不同的语义类别中。

ViT会将输入图像分成一个个小的图像块（patches）,并将每个图像块展平为一个向量，然后利用Transformer编码器学习全局特征和依赖关系。这种方法的优势在于，ViT能够利用自注意力机制来处理整个图像，而不是仅关注局部区域，从而提高了图像理解的能力。



上图就是VIT模型的整体框架，由以下几个部分构成：

输入图像切分： 首先，将输入的图像分割成一系列的固定大小的图像块（通常是16x16或32x32的小图像块）。每个图像块将被看作一个序列元素。

嵌入层（Embedding Layer）： 每个图像块通过一个嵌入层进行编码，将每个像素的特征表示转换为更高维的嵌入向量。这些嵌入向量将作为输入序列送入Transformer模型。

位置编码（Positional Encoding）： 与传统Transformer类似，ViT模型在输入序列中引入位置信息。这是通过添加位置编码向量到每个图像块的嵌入向量中来实现的，从而使模型能够理解图像中的空间关系。

Transformer Encoder： 接下来，ViT模型使用一层或多层的Transformer编码器来处理输入序列。每个编码器由多头自注意力机制（Multi-Head Self-Attention）和前馈神经网络（Feed-Forward Neural Network）两部分组成。

多头自注意力机制： 用于捕捉序列中不同位置之间的关系，特别是在图像块之间建立联系。它允许模型在处理每个图像块时能够关注其他图像块的信息，从而捕获全局的语义信息。

前馈神经网络： 用于对每个图像块的特征进行非线性变换，以增强特征的表示能力。

池化和分类层： 在经过一系列Transformer编码器后，ViT模型通常会采用全局平均池化（Global Average Pooling）来聚合所有图像块的信息，生成整体图像的特征表示。最后，将这些特征送入一个全连接层进行分类预测。

ViT模型的关键创新在于将图像块转化为序列数据，并利用Transformer的自注意力机制来捕获图像中的全局和局部信息。这使得ViT能够在一定程度上摆脱传统卷积神经网络的限制，对遮挡、尺度变化和位置变换等问题具有一定的鲁棒性。

1. gat有什么缺点

尽管GAT模型在处理图结构数据时具有很高的表现能力，但它也存在一些缺陷。首先，GAT模型的计算复杂度较高。由于每个节点都需要与其邻居节点进行注意力计算，这导致了计算量的显著增加。其次，GAT模型对于大规模图结构的处理效果不佳。由于注意力机制需要考虑每个节点与其所有邻居节点之间的关系，当图的规模较大时，GAT模型的计算和存储开销将会非常大。此外，GAT模型的注意力机制可能会受到噪声节点的影响，从而导致模型性能下降。

1. mape有什么缺点

平均绝对百分比误差。优点在于以百分比形式表示预测值与真实值之间的相对误差，更关注相对误差，对于不同量级的预测问题更具可比性。此外，MAPE在金融领域中常用于评估投资组合风险模型的表现。然而，MAPE的缺点在于当真实值接近零时，计算会出现分母为零的情况，导致评价结果不可用。同时，MAPE对小的误差较为敏感，可能会放大真实值较小的样本的误差。

31、怎么处理冷启动问题

1、数据

首先思考数据，能够帮助我们了解现状，知道手上有哪些底牌。数据一般包括用户数据和物品数据。

按数据来源的不同，考虑：内部数据、外部数据。

1.1、内部数据

内部数据包括：本产品线的数据、其他产品线的数据。

注意，在冷启动问题中，对于数据是“缺乏”而非“没有”。这意味着我们手上可能还是有一些数据的。

● 对于用户冷启动问题，用户在注册时填写的信息（手机号、地址、性别、年龄等）和注册时的环境信息（IP地址、GPS），可以帮助我们做出粗粒度的推荐。例如可以根据专家意见或决策树模型建立一些针对于不同年龄段、不同性别的用户的个性化榜单，然后在用户完成注册后，根据注册时填写的信息进行推荐。

● 对于物品冷启动问题，物品的一些属性信息也同样可以起到作用。在酒店推荐的场景下，可以根据新上线酒店的位置、价格、面积等信息，为酒店指定聚类，找到相似酒店，利用相似酒店的推荐逻辑完成冷启动过程。

另外，如果公司还有其他业务线，那么其他业务线的数据也可以拿过来使用。例如用户在美团已经积累了外卖数据，可以根据消费金额、家庭地址等分析得出用户的消费水平，那么在用户第一次使用美团的酒店服务时，也可以推荐出符合消费习惯的酒店。

1.2、外部数据

常见获取数据的手段包括：爬虫、平台对接。

● 爬虫是近乎于零成本的方案，但是可能会有一些法律风险。平台之间互相告对方非法爬取数据的新闻屡见不鲜。

● 有些第三方 DMP（Data Management Platform，数据管理平台）也会提供用户信息。像国外的 BlueKai、Nielsen，国内的 Talking Data 等公司都提供匹配率非常高的数据服务，可以极大地丰富用户的属性特征。像腾讯、百度、网易、Google等企业都与这些 DMP 平台有合作。

那 DMP 的数据是哪里来的呢？数据交换。通过合作的方式，企业给 DMP 提供用户的一些基本数据，DMP 对数据进行分析、挖掘，给企业提供更加全方位的用户信息。这样一来，企业就能获取到本来完全得不到的用户兴趣、收入水平、广告倾向等一系列高阶特征。

2、算法

在梳理完数据现状之后，接下来考虑算法的问题。

推荐系统的目标就是推荐给用户正确的商品，评价方式可以是点击率、在线观看时长等。在解决冷启动问题的过程中，无论用什么算法，算法的优化目标都要与总体目标一致。

算法可以从实现方式的不同，分为 3 类：基于规则、基于ML/DL、探索与利用。

2.1、基于规则的算法

基于规则的算法，一般给出的都是榜单类型的推荐结果。

在用户冷启动场景下，可以使用“热门排行榜”、“最新流行趋势榜”、“最高评分榜”等作为默认的推荐列表，实现非个性化推荐。可以根据专家意见建立一些针对于不同年龄段、不同性别的用户的个性化榜单，然后在用户完成注册后，根据注册时填写的信息进行粗粒度的个性化推荐。另外，在 LBS（Location Based Services，基于位置的服务）场景下，可以根据用户在注册时填写的地址信息、GPS 信息，按一定规则推荐周围的店家/商品。

在物品冷启动场景下，可以按一定规则寻找相似商品进行绑定，完成推荐。

需要注意的是，基于规则的算法更多依赖的是专家对业务的洞察。因此在制定规则时，需要充分了解业务，充分利用已有数据，才能让冷启动规则合理且高效。

2.2、基于ML/DL

基于 ML/DL 的算法要解决的是用户冷启动或物品冷启动问题，而非系统冷启动问题。因此前提是，系统已经上线，同时也已经有了一定的数据积累。

机器学习（ML）的思路是，将基于规则的算法改造为机器学习模型，按学习方式的不同，又可以分为有监督学习和无监督学习（当然还有半监督学习，此处不展开）。

● 有监督学习：在前面的例子中，可以利用点击率目标构造一个用户属性的决策树，在每个决策树的叶节点建立冷启动榜单，然后新用户在注册后，根据其有限的属性信息，寻找到决策树上对应的叶节点榜单，完成冷启动过程。

● 无监督学习：例如使用聚类算法，来寻找相似物品，但要注意维度灾难问题。

需要注意的是，由于数据的缺乏，不能选用复杂的机器学习模型，否则容易造成过拟合问题。

而对于新用户，由于其特征非常的稀疏，使用基于深度学习（DL）的推荐系统效果会比较差，那有什么方法呢？可以考虑迁移学习和强化学习。

● 迁移学习如果有其他业务线的数据，也可以拿过来使用。冷启动问题本质上是某领域的数据或知识不足导致的，如果能够将其他领域的知识用于当前领域，那么冷启动问题自然迎刃而解。我们称这种做法为“迁移学习”，常见的做法是共享特征（在深度学习模型中就是共享 Embedding）或共享模型参数。例如将 CTR 模型中的用户 Embedding 和物品 Embedding 应用到 CVR 模型中，直接用于训练。Embedding 是一种高维特征到低维特征的映射，训练好的 Embedding 可以反映用于与隐变量、商品与隐变量之间的内在联系。

● 强化学习：所谓强化学习，就是指智能体（即模型）根据环境（即用户、物品等）的反馈（即点击或不点击）来采取行动（即推荐商品列表）并改变自身状态（更新模型参数），然后再获得反馈再采取行动再改变状态的循环过程。在一次次的迭代过程中，让推荐系统尽快度过冷启动状态。

2.3、运筹优化

运筹优化在推荐系统中的应用场景是多样的，而在冷启动问题里，主要是用于解决物品冷启动问题。当然，同样也可以用来解决系统冷启动的问题。

具体而言，就是是在“探索新数据”和“利用旧数据”之间进行平衡，使系统既能够利用旧数据进行推荐，达到推荐系统的商业目标，又能高效地探索冷启动的物品是否是“优质物品”，使冷启动物品获得曝光的倾向，快速收集冷启动数据。我们又称这个过程为“探索与利用”。

显然，这是一个多目标优化问题。

一个经典的解决办法是 UCB（Upper Confidence Bound，置信区间上界）。公式如下。其中 为观测到的第 个物品的平均回报（这里的平均回报可以是点击率、转化率等）， 是目前为止向该用户曝光第 个物品的次数， 是到目前为止曝光所有物品的次数之和。

在新物品刚上架的时候， 比较低，但是因为曝光次数 也比较小，所以 会比较大，最后 值会比较大，新物品的曝光机会较大。随着曝光次数的增加， 在公式中的相对值逐渐减小，最后 就主要取决于 了。也就是说，使用 UCB 方法进行推荐，推荐系统会倾向于推荐“效果好”或“冷启动”的物品。随着冷启动物品被有倾向性的推荐，能够快速收集反馈数据，最后快速通过冷启动阶段。

3、产品

最后讨论一下从产品的角度，要怎么帮助解决冷启动问题。

冷启动问题之所以出现，就是因为缺乏有价值的数据，那么在产品功能方面，就要尽量帮助收集数据。

● 用户冷启动：有些应用会在用户第一次登录时，引导用户输入一些冷启动特征。例如，一些音乐类产品会引导用于选择“音乐风格”；一些视频类产品会引导用户选择几部自己喜欢的电影。

● 物品冷启动：有些应用会以积分奖励的方式引导用户输入一些物品特征。像大众点评上的评论体系，淘宝上的评价系统，都是帮助商家、商品快速度过冷启动解决的利器。

31、怎么处理数据分布不均问题

* 重采样（Resampling）：

对数据进行过采样（增加少数类样本）或欠采样（减少多数类样本）以平衡类别。

* 合成数据生成（Synthetic Data Generation）：

使用技术如SMOTE（Synthetic Minority Over-sampling Technique）来生成少数类的合成样本。

* 改变损失函数（Modifying Loss Function）：

使用如加权交叉熵等损失函数，对不同类别的样本赋予不同的权重。

* 使用集成学习（Ensemble Learning）：

结合多个模型的预测结果，如随机森林或提升方法，可以帮助处理类别不平衡。

* 专注于评价指标（Focusing on Evaluation Metrics）：

使用像精确率-召回率曲线（Precision-Recall Curve）或F1分数等更适合处理不平衡数据的评价指标。

1. **大模型篇**

**进行领域大模型预训练应用哪些数据集比较好？**

通过分析发现现有的开源大模型进行预训练的过程中会加入书籍、论文等数据。主要是因为这些数据的数据质量较高，领域相关性比较强，知识覆盖率（密度）较大，可以让模型更适应考试。给我们自己进行大模型预训练的时候提供了一个参考。同时领域相关的网站内容、新闻内容也是比较重要的数据。

**领域数据训练后，通用能力往往会有所下降，如何缓解模型遗忘通用能力？**

如果仅仅使用领域数据集进行模型训练，模型很容易出现灾难性遗忘现象，为了解决这个问题通常在领域训练的过程中加入通用数据集。那么这个比例多少比较合适呢？目前还没有一个准确的答案。主要与领域数据量有关系，当数据量没有那么多时，一般领域数据与通用数据的比例在1:5到1:10之间是比较合适的。

**进行SFT操作的时候，基座模型选用Chat还是Base?**

在进行SFT实验的时候，大模型选用Chat还是Base作为基座，需要根据SFT的数据量进行决定。如果你只拥有小于10k数据，建议你选用Chat模型作为基座进行微调；如果你拥有100k的数据，建议你在Base模型上进行微调。

### **用于大模型微调的数据集如何构建？**

进行大模型微调时，数据是比较重要的，数据的高度决定模型效果的高度，因此数据的质量重要性大于数据的数量的重要性，因此对于构建微调数据时的几点建议如下所示：

（1）选取的训练数据要干净、并具有代表性。

（2）构建的prompt尽量多样化，提高模型的鲁棒性。

（3）进行多任务同时进行训练的时候，要尽量使各个任务的数据量平衡。

### **预训练和微调是哪个阶段注入知识的？**

知识是在预训练阶段注入的，而微调阶段是指在特定任务上的训练，以使预训练模型的通用知识和特定任务的要求相结合，使模型在特定任务上表现更好。

### **想让模型学习垂直领域的知识，是应该预训练还是微调？**

对于大模型来说是在预训练的阶段注入领域知识的，因此在训练行业大模型的时候最佳的方案是使用预训练与微调相结合的方案，先用篇章数据进行预训练以获取广泛的知识，再用问答数据进行微调，使模型更好的学习到特定领域的知识。

不过GPT系列大模型的预训练和微调，从实现方式上来讲是没有什么差别的，都是用decoder only的语言模型进行训练并更新参数，如果样本数量比较少，没有大量的篇章文档数据，个人认为只进行微调也能够注入知识，就不必再进行预训练了。如果特定的垂直领域跟预训练模型的分布差别不是很大，也不用再进行二次预训练。

### **微调后的大模型出现灾难性遗忘是什么原因？**

灾难性遗忘是指大模型在学习了某个行业的新知识后，遗忘掉了最初学习的通用知识，减缓该问题的出现可以从如下两方面进行着手：

（1）加入通用知识：在用行业数据进行训练的时候增添一些通用的数据一块训练。

（2）调整学习率LR：出现这种情况也有可能是训练参数调整导致的，微调初始学习率不要设置的太高，LR=2e-5或者更小，最好不要设置大于预训练时的学习率，能够缓解此问题。

**什么是LLM的复读机问题？**

大模型LLM的复读机问题是指大型语言模型在生成文本时出现的一种现象，也就是模型倾向于无限地复制输入的文本或者以过渡频繁的方式生成重复相同的句子或短语。这种现象使得模型的输出缺乏多样性和创造性，给用户带来了不好的体验。

**出现复读机问题的可能原因有哪些？**

通过分析发现导致大模型的复读机问题出现可能是以下几个方面的原因。

（1）数据偏差：大型语言模型通常是通过预训练阶段使用大量的无标签数据进行训练的。如果训练数据中含有大量的重复文本或者某些特定的句子或短语出现频率较高，模型在生成文本时就有可能倾向于复制这些常规的模式。

（2）训练目标的限制：大型语言模型的训练通常是基于自监督的学习方法，通过预测下一个词或掩盖词、短语来学习语言模型。这样的训练目标可能使得模型更倾向于生成与输入相似的文本，导致复读机问题的出现。

（3）缺乏多样性的训练数据：虽然大型语言模型能够处理大规模的数据，但如果训练数据中缺乏多样性的语言表达和语境，模型可能无法学习到足够的多样性和创造性，导致复读机问题的出现。

（4）模型结构和参数设置：大型语言模型的结构和参数也可能对复读机问题产生影响。比如，模型的注意力机制和生成策略可能导致模型更倾向于复制输入的文本。

**解决大模型复读机问题可用哪些策略？**

如何缓解大模型的复读机问题是一个复杂的任务，并没有一个通用的解决方案。不同的方法可能适用于不同的业务场景和任务，需要根据具体的情况进行选择和调整。下面是几种用于缓解大模型复读机问题的几种解决方案。

（1）多样性训练数据：在训练阶段，使用具有多样性的语料进行训练模型，避免数据偏差和重复文本的问题。比如可以从不同来源、不同领域、不同风格的文本中获取数据。

（2）加入噪声：在文本生成时，可以引入一些随机性或噪声，以增加生成文本的多样性。

（3）温度参数调整：温度参数是用于控制生成文本多样性的一个参数，通过调整温度参数值，可以控制生成文本的独特性和多样性。较高的温度值能够增加随机性，从而减少复读机问题的出现。

（4）解码参数调整：目前在生成文本时常用的解码算法有Beam搜索算法，可以通过调整Beam大小和搜索宽度，控制文本的多样性和创造性。

（5）重复惩罚参数调整：repetition\_penalty用于减少生成文本中重复词汇的出现。它可以设置为大于1的值，以惩罚重复的词汇。这有助于生成更自然、不重复的文本。

（6）后处理和过滤：对生成的文本进行后处理和过滤，去除重复的句子或短语，以提高生成文本的质量和多样性。可以通过文本相似度计算方法或规则来检测和去除重复的文本。

1. **目前有哪几种流行的大模型架构**

BART (bi Encoder+casual Decoder，类bert的方法预训练)、T5(Encoder+Decoder，text2text预训练)、GPT(Decoder主打zero-shot)、GLM(mask的输入部分是双向注意力，在生成预测的是单向注意力)。

1. **prefix LM和casual LM的区别**

attention mask不一样，前者prefix的部分token可以相互看

到，而causal是严格自回归。

1. **RLHF流程讲一下**

a.Step1:语言模型根据问题生成答案或续写;

b.Step2:使用函数/模型/人类反馈或它们的某种组合来评估

问题和答案。并映射至一个分数，即奖励reward;

c.Step3:在PPO优化步骤中，<问题+回复>pair用于计算序

列中标记的对数概率。经过训练的模型(actor)和参考模型(critic) 分别得到new\_logit和old\_logit，参考模型是上一阶段的指令微调 模型。两个输出之间的KL 散度用作附加奖励信号，以确保生成的响 应不会偏离参考语言模型太远。然

后使用 PPO算法训练语言模型

1. **instruction tuning和prompt learning 的区别**

instruction tuning和prompt learning的目的都是去挖掘语言模型本身具备的知识。不同的是Prompt是激发语言模型的补全能力，例如根据上半句生成下半句，或是完形填空等(few- shot)。Instruct是激发语言模型的理解能力，它通过给出更明显的指令，让模型去做出正确的行动(zero-shot)。

1. **LoRA怎么做的，讲一下?**

即在原始的预训练模型旁边增加一个新的通路，通过前后两个矩阵A,B相乘，做一个降维再升维的操作。外挂层和预训练模型层维度都为d,A会先将维度d降维到r，B再升回d。一般是针对Q/K/V的投影矩阵W分解成低秩矩阵BA作为外挂，B一般

以0初始化，A以高斯分布初始化

1. **为什么可以用LoRA ?**

专有任务的微调权重与初始预训练权重之间的差异往往表现出“低固有秩(low intrinsicrank)”差异，这意味着它可以很好地近似为一个低秩矩阵。即微调权重和初始预训练权重之间的这种差距可以表示为两个较小矩阵的乘积**。**

1. **LoRA的参数**

a. rank秩8:选择更高秩的分解矩阵将抵消LoRA的效率优

势，而且设到16也差别不大。

b. Alpha:16。Alpha用于对学习到的权重进行扩展。通常建议

固定Alpha的值为16，不调节。

c.目标模块:所有密集层。初始只对QV，但应用所有有效，

且更接近全参数。

d.基础学习率:1e-4。只有当训练不稳定时降到3e-5。

**8.LoRA和全参微调的对比**

a.对普通文本任务如生成sql差别不大，对小模型推理影响很

多，但到70B的时候差别不大。此外，数据偏离分布外太远可能会导致LoRA难以处理，有提示对LoRA训练更稳定，没有提示用特殊词元代替(需训练比如4个特殊 start/end等)

b. LoRA虽然在内存方面十分高效，但可能会影响模型达到收

敛的速度。

1. **国外开源的LLaMA的词表实际上兼容中文效果可能会大打折扣，那么扩充词表该怎么做?**

a.准备一份中文训练预料，用sentencepiece训练切词，得到

扩增的中文词表，然后增加到模型原来的词表中

b. embedding矩阵随机初始化(或均值初始化)新增的token

对应的向量

c.进一步做pretraining或者sft