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。

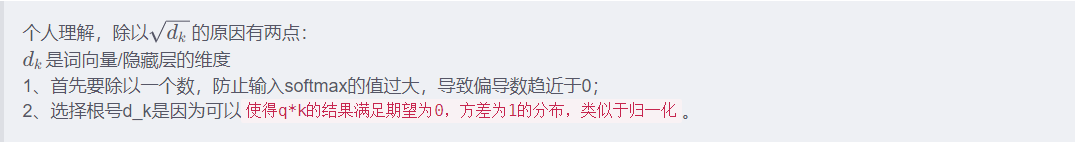
20、**Encoder和decoder是如何进行交互的?**

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

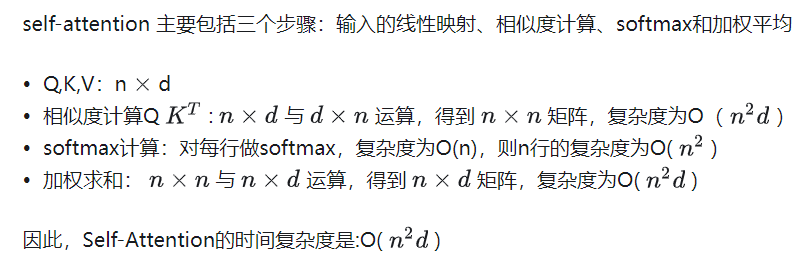
1. 基于attention有哪些代表性的改进方法?分别针对的是什么问题
2. 多头注意力（Multi-Head Attention）：在标准attention机制中，输入被缩放然后与权重相乘以产生输出。在多头attention中，输入首先被分为多个“头”，每个头独立计算attention权重，然后将结果拼接起来。这种方法可以使模型更好地理解和处理输入数据。
3. 自注意力（Self-Attention）：在许多任务中，输入数据的一部分与另一部分是高度相关的。自注意力机制让模型学习这种关系，从而提高性能。例如，在机器翻译任务中，句子中的单词可能会依赖于其他单词。通过让模型关注整个句子，而不是仅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

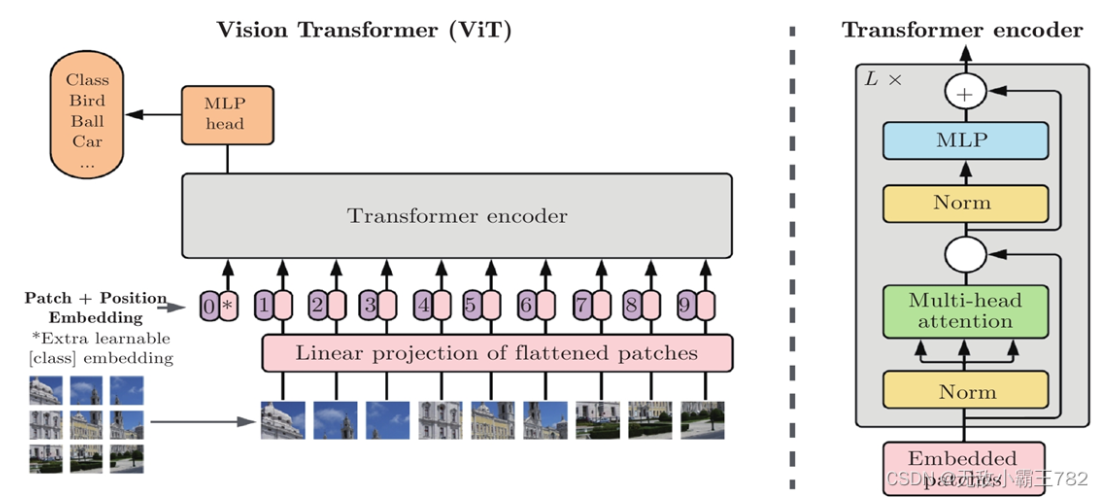
逻辑回归模型不用MSE作为损失函数的原因主要有两个，一个从背景意义来说，交叉熵函数更贴切“概率”的意义。另一方面是从求解的角度，交叉熵损失函数比MSE损失函数更易于求解。

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。

# **32、京东物流算法实习面试题7道**

### **1、GLM是什么？**

GLM（Generalized Linear Model）是一种广义线性模型，用于建立变量之间的关系。它将线性回归模型推广到更广泛的数据分布，可以处理非正态分布的响应变量，如二项分布（逻辑回归）、泊松分布和伽玛分布等。GLM结合线性模型和非线性函数，通过最大似然估计或广义最小二乘估计来拟合模型参数。

### **2、SVM的原理？怎么找到最优的线性分类器？支持向量是什么？**

SVM（Support Vector Machine）是一种二分类模型，其原理是在特征空间中找到一个最优的超平面，能够最大化不同类别之间的间隔。SVM的目标是找到一个能够将两个类别分开的超平面，并且使得离超平面最近的样本点到该超平面的距离最大化，这些离超平面最近的样本点称为支持向量。

找到最优的线性分类器的过程包括：

* 寻找最大间隔超平面：通过求解一个凸优化问题，使得间隔最大化。
* 转化为对偶问题：通过转化为对偶问题，可以更高效地求解最优化问题。
* 核技巧（Kernel Trick）：如果数据不是线性可分的，可以使用核函数将数据映射到高维特征空间，使其在高维空间中线性可分。

支持向量是训练过程中对于确定超平面位置最关键的样本点，它们位于各个类别之间的边界上，决定了超平面的位置和形状。

### **3、介绍一下CNN？**

CNN（Convolutional Neural Network，卷积神经网络）是一种深度学习模型，主要用于图像识别和计算机视觉任务。CNN通过卷积层、池化层和全连接层等组件，自动学习图像的特征表示。

CNN的主要包括：

* 卷积层：通过卷积操作，提取图像的局部特征，保留空间结构信息。
* 池化层：通过降采样操作，减少特征图的维度，提取更加抽象和稳定的特征。
* 全连接层：将特征图转化为向量，并通过全连接层进行分类或回归。

### **4、CNN中的卷积到底指什么？举个例子？**

在CNN中，卷积（Convolution）指的是一种特殊的数学操作，用于提取图像的局部特征。卷积操作基于滤波器（也称为卷积核或特征检测器），它通过滑动窗口在输入图像上进行计算。

具体来说，卷积操作将滤波器与输入图像的小区域进行逐元素相乘，并将结果相加，从而得到一个新的特征值。通过将滤波器在整个图像上滑动，可以获得一个特征图，其中每个位置都对应一个特征值。卷积操作可以保留图像的空间结构信息，并在不同位置共享参数，减少模型的参数量。

举个例子，假设有一个3x3的滤波器和一个5x5的输入图像。卷积操作首先将滤波器的每个元素与输入图像的对应区域的像素值进行相乘，然后将乘积结果相加，得到一个新的特征值。然后，滤波器在输入图像上移动一个像素，并重复这个过程，直到覆盖整个输入图像。最终，得到一个3x3的特征图，其中每个位置对应一个特征值。

卷积操作在CNN中的重要性在于，通过多个卷积层的堆叠，可以提取不同层次的图像特征，从低级的边缘和纹理特征到高级的形状和物体特征，以支持图像分类、目标检测等计算机视觉任务。

### **5、介绍决策树、信息熵？**

决策树（Decision Tree）是一种基于树形结构的机器学习算法，用于分类和回归任务。它通过对输入特征进行一系列的二分切分，构建一个树结构来进行决策。

决策树的构建过程如下：

* 选择最佳切分特征：通过计算特征的纯度或不纯度指标，选择最佳的特征来切分数据集。
* 切分数据集：根据选定的切分特征和阈值，将数据集划分为两个子集，每个子集将作为下一层决策树的输入。
* 递归构建子树：对每个子集，重复上述步骤，递归构建子树，直到满足终止条件，如达到最大深度或样本数量小于阈值。
* 预测和分类：根据构建好的决策树，对新的样本进行预测或分类。通过沿着树的路径根据特征值进行判断，最终到达叶子节点并输出相应的预测结果。

信息熵（Information Entropy）是衡量数据集纯度或不确定性的度量指标。在决策树算法中，常用的纯度指标之一是信息熵。信息熵的计算公式如下：

H(X) = -Σ p(x) \* log2(p(x))

其中，H(X)代表数据集X的信息熵，p(x)代表样本属于类别x的概率。

信息熵的值越大，表示数据集的不确定性越高，纯度越低。决策树通过选择能够最大程度降低信息熵的特征来进行切分，从而提高子集的纯度。当数据集的信息熵为0时，表示数据集完全纯净，所有样本属于同一类别。

决策树算法的目标是通过选择最佳切分特征和阈值来构建一个具有高纯度的决策树模型，以实现对新样本的预测和分类。信息熵在此过程中起到了衡量数据纯度的重要作用。

### **6、随机森林“随机”二字体现在什么地方？**

在随机森林（Random Forest）中，“随机”二字体现在以下几个方面：

* 随机采样：随机森林通过从原始数据集中有放回地随机采样形成多个训练子集（bootstrap samples），用于构建不同的决策树。这样做可以使得每个决策树使用的样本有所不同，增加模型的多样性。
* 随机特征选择：在每个决策树的节点切分时，随机森林只考虑部分特征的子集进行切分。这样可以确保每个决策树对特征的使用略有差异，增加模型的多样性，并减少特征之间的相关性。
* 随机森林是一种集成学习方法，它通过同时构建多棵决策树并基于集体投票或平均来做出最终预测。由于每棵决策树都是通过随机性的方式构建的，随机森林能够减少过拟合风险，并在处理大量数据和高维特征时表现良好。

### **7、介绍一下XGBoost，与GBDT相比有什么不同？**

XGBoost（eXtreme Gradient Boosting）是一种梯度提升树（Gradient Boosting Decision Trees）的优化实现。它在GBDT的基础上进行了改进和优化，提供了更高的性能和可扩展性。

与传统的GBDT相比，XGBoost具有以下不同之处：

1. 正则化：XGBoost引入了正则化项，包括L1和L2正则化，以控制模型的复杂度，减少过拟合的风险。
2. 列采样：XGBoost支持对特征进行列采样，即在每棵树的每次分裂时随机选择一部分特征。这样可以减少特征之间的相关性，并增加模型的多样性。
3. 高效的并行处理：XGBoost通过使用多线程进行模型训练，同时支持在分布式环境中进行并行训练，以提高训练速度和效率。
4. 特征重要性评估：XGBoost提供了一种可靠的方法来评估特征的重要性，根据特征在模型中的使用频率和它们对预测结果的贡献程度进行排序。
5. 提升性能：XGBoost通过使用二阶导数（Hessian）近似来更好地优化损失函数，并采用分位数损失函数（Quantile Loss）等来应对不平衡数据和异常值。

总之，XGBoost是GBDT的一种优化实现，通过正则化、列采样、并行处理和特征重要性评估等手段，提供了更高的性能和可扩展性。它在机器学习和数据竞赛中广泛应用，具有出色的性能和鲁棒性。

# **荣耀NLP算法工程师面试题8道**

### **1、讲一下transformer**

Transformer是一种用于自然语言处理和机器翻译的神经网络模型。它引入了自注意力机制，能够捕捉输入序列中的长距离依赖关系。Transformer由编码器和解码器组成，每个模块都由多个层堆叠而成。编码器用于将输入序列映射到一系列连续表示，解码器则将这些表示转化为输出序列。

### **2、transformer怎么调优**

Transformer的调优可以包括以下几个方面：

调整模型架构：可以尝试增加或减少编码器和解码器层的数量，调整隐藏单元的维度等。

学习率调度：使用学习率调度策略，如逐渐减小学习率、使用预热步骤等。

正则化：使用Dropout、权重衰减等正则化技术，防止过拟合。

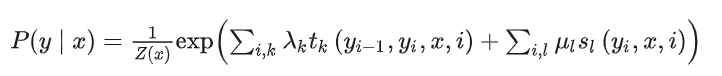
批量大小和训练迭代次数：调整批量大小和训练迭代次数，以获得更好的训练效果。

初始化策略：选择合适的参数初始化方法，如Xavier初始化、高斯初始化等。

梯度裁剪：为了防止梯度爆炸，可以对梯度进行裁剪，限制梯度的最大范数。

### **3、讲一下CRF，公式是什么**

条件随机场（CRF）是一种用于序列标注任务的统计模型。CRF可以建模输入序列和输出序列之间的依赖关系。其公式如下：

其中，P(y|x)是给定输入序列x的条件下输出序列y的概率，Z(x)是归一化因子，λ\_k和u\_l是特征函数的权重，t\_k和s\_l是特征函数，y\_i和y\_i-1分别表示输出序列的第i个和第(i-1)个标签，N是输出序列的长度，K是标签的数量。

### **4、讲讲word2vec和word embedding区别**

Word2Vec是一种用于将单词表示为连续向量的技术，它通过学习上下文信息来为每个单词生成固定维度的向量表示。Word2Vec基于分布式假设，即将上下文相似的单词嵌入到相似的向量空间中。Word2Vec有两种模型：Skip-gram和CBOW（Continuous Bag-of-Words）。Word embedding是指将单词映射到低维度的向量空间的一般术语。Word2Vec是一种用于生成word embedding的具体方法的技术。Word2Vec通过训练一个神经网络模型，从大规模文本语料中学习单词的分布式表示。

区别在于，Word2Vec是一种具体的算法，它是生成word embedding的一种方法。而word embedding是指将单词映射到低维度向量空间的技术，可以使用不同的方法来实现，而不限于Word2Vec。Word2Vec是基于上下文信息的分布式表示方法之一，而word embedding是一个更广泛的概念，涵盖了多种生成单词向量表示的方法，如GloVe、FastText等。

### **5、gpt3和gpt2的区别**

GPT-3和GPT-2是由OpenAI开发的两个语言模型。它们的区别主要在于规模和功能上的不同。GPT-3是目前最大的语言模型，具有1750亿个参数，而GPT-2则有15亿个参数。

由于GPT-3规模更大，它在自然语言处理任务上的表现更好，并且能够生成更连贯、更具逻辑性的文本。GPT-3还支持零样本学习，即可以在没有对特定任务进行显式训练的情况下执行各种语言任务。

另一个区别是GPT-3在文本生成方面的能力更强大，可以生成更长的文本，而GPT-2的生成长度有一定的限制。此外，GPT-3的使用需要更高的计算资源和成本。

### **6、讲讲生成模型和判别模型的区别**

生成模型和判别模型是两种常见的机器学习模型类型。

生成模型（Generative Model）是指能够学习数据的联合概率分布，并从中生成新的样本。生成模型能够对输入数据进行建模，并学习数据的统计特性，例如生成图像、生成文本等。常见的生成模型包括变分自编码器（VAE）和生成对抗网络（GAN）等。

判别模型（Discriminative Model）则是指通过学习输入数据与标签之间的条件概率分布，来对不同类别进行分类或预测。判别模型关注的是输入与输出之间的关系，能够进行分类、回归等任务。常见的判别模型包括逻辑回归、支持向量机（SVM）和深度学习中的卷积神经网络（CNN）和循环神经网络（RNN）等。

简而言之，生成模型关注数据的生成过程，能够生成新的样本，而判别模型则关注数据的分类或预测问题。

### **7、讲讲LDA算法**

LDA（Latent Dirichlet Allocation）是一种用于主题建模的概率图模型。它可以自动地将文本集合划分为多个主题

LDA（Latent Dirichlet Allocation）是一种用于主题建模的概率图模型。它可以自动地将文本集合划分为多个主题，并为每个文档分配主题的概率分布，同时为每个主题分配单词的概率分布。

LDA的基本思想是假设每个文档包含多个主题，并且主题决定了文档中单词的生成过程。具体来说，LDA认为文档的生成过程可以分解为两个步骤：首先，从主题分布中选择一个主题；然后，从选择的主题的单词分布中选择一个单词。通过这种生成过程，可以获得文档中单词的分布，从而推断出主题的分布。

LDA的目标是通过观察到的文本数据来估计主题和单词分布的参数。通常使用基于变分推断或Gibbs采样等方法进行参数估计。一旦得到了主题和单词分布，就可以使用LDA模型来进行主题分析、文本分类、信息检索等任务。

总结来说，LDA是一种无监督学习算法，用于从文本数据中学习主题的概率模型。它可以揭示文本数据的隐含主题结构，并在许多文本处理任务中发挥重要作用。

### **8、有做过NER吗，讲讲prompt learning**

"NER"代表命名实体识别（Named Entity Recognition），是指从文本中识别和提取特定类别的命名实体，如人名、地名、组织机构等。"Prompt learning"是一种训练模型的方法，它通过为模型提供一些示例输入和输出对（prompt）来引导模型学习特定的任务。

在NER任务中，prompt learning可以用于训练模型以执行命名实体识别。通常的做法是为模型提供一些示例文本和相应的NER标签作为prompt，然后使用这些prompt进行有监督的训练。通过这种方式，模型可以学习识别文本中的命名实体，并生成相应的标签。

Prompt learning在NER任务中的一个重要应用是零样本学习（zero-shot learning）。零样本学习指的是在没有对特定实体进行显式训练的情况下，通过提示样本的学习，使模型能够识别和分类以前未见过的实体。通过合理设计的prompt，模型可以学习到通用的命名实体识别能力，并在新的实体上进行推断和预测。

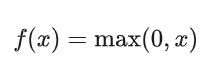
总结来说，NER是一项从文本中识别命名实体的任务，而prompt learning是一种训练模型的方法，可以用于引导模型学习NER任务，并在零样本学习中发挥重要作用。

## **[快手广告算法工程师面试题15道](https://zhuanlan.zhihu.com/p/630121638" \t "https://www.zhihu.com/people/_blank)**

### **1、ReLU，LeakyReLU**

ReLU和LeakyReLU都是常见的激活函数。

ReLU的表达式为



LeakyReLU的表达式为



其中α是一个小于1的常数。

### **2、ReLU的优缺点**

ReLU的优点在于它具有快速收敛和计算的优势，同时还能够缓解梯度消失的问题。ReLU的缺点是当输入为负数时，该激活函数的导数为0，从而造成部分神经元失活的问题。因此，应用ReLU激活函数时需要注意，确保输入数据的均值为0。

### **3、ReLU激活函数是如何解决梯度消失和梯度爆炸问题的？**

ReLU的梯度在正区间恒定为1，而负区间恒定为0。这使得无论神经元的输出值如何，都不会导致梯度消失或梯度爆炸的问题。因此，ReLU激活函数能够有效地解决梯度消失和梯度爆炸问题。

### **4、什么是梯度消失和梯度爆炸？什么单元更容易出现梯度消失梯度爆炸的问题？**

梯度消失指的是在反向传播过程中，权重和激活值都小于1，导致梯度不断缩小，最终导致梯度消失的现象。梯度爆炸则是指在反向传播过程中，权重和激活值都大于1，导致梯度不断增大，最终导致梯度爆炸的现象。其中，输出值小于0的sigmoid神经元更容易出现梯度消失的问题。

### **5、ReLU如何解决梯度消失问题？**

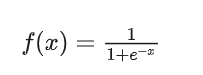
ReLU激活函数对于非负的输入值，其梯度恒定为1，不会缩小，因此能够有效地避免梯度消失。

### **6、ReLU之前常用的激活函数？**

以前常用的激活函数包括sigmoid函数、tanh函数、以及softmax函数等。

### **7、写一下sigmoid函数的表达式。sigmoid函数存在的问题。**

sigmoid函数的表达式为



它的问题是当输入值过大或过小时，其导数值非常小，从而导致梯度消失和梯度爆炸的问题。

### **8、写一下逻辑回归的损失函数。**

### 

### **9、讲一下交叉熵损失函数，与KL散度的关系 ，在二分类和多分类问题中的区别。**

交叉熵损失函数是二分类问题中最常用的损失函数，由于其定义出于信息学的角度，可以泛化到多分类问题中。KL散度是一种用于衡量两个分布之间差异的指标，交叉熵损失函数是KL散度的一种特殊形式。在二分类问题中，交叉熵函数只有一项，而在多分类问题中有多项。

### **10、卷积神经网络中常见的层有哪些？**

卷积神经网络中常见的层包括卷积层(Convolutional layer)、池化层(Pooling layer)、全连接层(Fully connected layer)，以及批标准化层(Batch normalization layer)等。

### **11、卷积层的参数计算，以及计算量计算。**

卷积层的参数数量取决于卷积核大小、输入通道数量和输出通道数量。计算量计算则取决于卷积核大小、输入图像大小、输出通道数量和步长大小。

### **12、m\*n的图像，卷积核为k\*k，步长为1，最后生成的图像大小。**

对于m×n的图像，卷积核为k×k，步长为1时，卷积后生成的图像大小为(m-k+1)×(n-k+1)。

### **13、m\*n\*3的图像输入进去，输出会有变化吗？**

输入的m×n×3的图像经过卷积操作后输出的通道数可能会有所变化，但是图像的宽高不会发生变化。

### **14、卷积核的k值的取值，为什么取这个值**

卷积核的大小k一般为奇数，这是因为奇数大小的卷积核有中心像素，可以更好地对称地影响图像的每个位置。同时，选择奇数大小的卷积核还可以保证最终卷积后的图像不会出现偏差。

### **15、ResNet中的Block，怎么用的**

ResNet中的Block是指由复杂的卷积和激活层组成的模块，可以有效地解决反向传播中出现的梯度消失和梯度爆炸的问题。ResNet中的每个Block都包含有多个卷积层以及一个Skip Connection，可以逐渐构建出一个非常深的神经网络。当网络层数较深时，通过增加Skip Connection的结构可以更好地保留梯度信息，从而提高模型的性能。

# **联想算法工程师面试题9道**

### **1、分类问题的交叉熵是什么？**

分类问题的交叉熵（cross-entropy）是一种用来衡量分类模型输出与真实标签之间差异的指标。在二分类问题中，交叉熵可以表示为以下公式：

其中，p 表示真实标签，q 表示模型预测的标签，N 表示样本数量。该公式可以看作是一个基于概率分布的比较方式，即将真实标签看做一个概率分布，将模型预测的标签也看做一个概率分布，然后计算它们之间的交叉熵。

### **2、分类问题是否可以用MSE？**

分类问题通常不能使用均方误差（MSE）作为损失函数，因为分类问题中的标签通常是离散的，而 MSE 适用于连续变量的回归问题。当使用 MSE 作为损失函数时，模型的输出可能会超过 1 或小于 0，这是因为 MSE 的计算方式不适用于概率值的范围。在分类问题中，常用的损失函数包括交叉熵、对数损失函数等。

### **3、推荐系统中，相比于余弦相似度，是否可以用欧几里得距离判断相似度？**

在推荐系统中，通常使用余弦相似度来度量用户或物品之间的相似度。因为余弦相似度考虑的是向量之间的夹角，而不是向量的长度，因此它对于不同大小的向量比较稳健。而欧几里得距离则是考虑向量之间的长度，因此对于不同大小的向量比较敏感。在推荐系统中，通常物品或用户的特征向量长度不同，因此欧几里得距离不适用于推荐系统中的相似度度量。

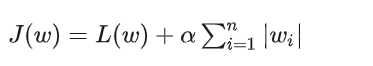
### **4、过拟合怎么处理？**

过拟合是指模型在训练集上表现良好，但在测试集上表现较差的现象。过拟合的原因通常是模型过于复杂，参数过多，导致模型在训练集上过度拟合。常见的处理方法包括以下几种：

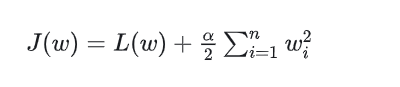
* 增加数据量：通过增加训练数据来降低模型对于训练集的过度拟合。
* 简化模型：减少模型的参数量，简化模型的结构，降低模型的复杂度。
* 正则化：
* 正则化：通过在损失函数中添加正则化项，限制模型参数的大小，从而避免模型过度拟合。常见的正则化方法包括 L1 正则化和 L2 正则化。
* 提前停止训练：通过监控模型在验证集上的表现，当模型在验证集上的表现不再提升时，提前停止训练，避免模型在训练集上过度拟合。

### **5、L1、L2正则化的效果、区别、原理？**

L1 正则化和 L2 正则化都是正则化方法，目的是通过限制模型参数的大小，降低模型的复杂度，防止过拟合。L1 正则化会使一部分参数变为 0，从而实现特征选择的效果，适合处理稀疏数据。L2 正则化则会让所有参数都趋向于较小的值，但不会使参数为 0。

L1 正则化的损失函数可以表示为： 其中，L(w) 是模型在训练集上的损失函数，w\_i 是模型的第 i 个参数，\alpha 是正则化强度的超参数。

L2 正则化的损失函数可以表示为：

其中，L(w)、w\_i 和 \alpha 的含义与 L1 正则化相同。

### **6、Dropout的原理、在训练和测试时的区别？**

Dropout 是一种常用的正则化方法，其原理是在每次迭代中随机将一部分神经元的输出置为 0，从而减少神经元之间的共适应性，防止过拟合。

在训练时，Dropout 会随机将一定比例的神经元的输出置为 0，从而产生多个不同的子网络，每个子网络都只学习了部分信息，强制使网络更加鲁棒。在测试时，为了保持网络的稳定性，Dropout 通常被关闭，而是使用训练时每个子网络的加权平均作为最终的预测结果。

### **7、SGD、Adam、动量优化的SGD？**

SGD（Stochastic Gradient Descent）是一种基本的梯度下降算法，它使用单个样本的梯度来更新模型参数，从而实现对损失函数的最小化。SGD 的缺点是可能会陷入局部最优解，并且收敛速度较慢。

动量优化的 SGD（Momentum SGD）是在 SGD 的基础上引入了动量的一种优化算法。动量可以看作是给梯度增加一个惯性项，使得在更新参数时，当前的梯度不仅考虑当前时刻的梯度，还会考虑之前的梯度信息，从而使得更新方向更加平稳，收敛速度更快。

Adam 是一种基于梯度的优化算法，结合了 RMSProp 和动量优化的思想。Adam 通过维护一个梯度的指数加权移动平均和梯度平方的指数加权移动平均来自适应地调整每个参数的学习率，从而实现快速的收敛。

### **8、Adam和动量优化的SGD效率上的区别？**

相对于动量优化的 SGD，Adam 有以下几个优点：

* 自适应性：Adam 能够自适应地调整每个参数的学习率，从而使得模型在不同的参数空间下能够更加快速地收敛。
* 速度快：Adam 在训练过程中能够快速地找到最优解，因此通常能够在较少的迭代次数内达到更好的性能。
* 可靠性高：Adam 对超参数的选择不太敏感，因此更容易调整到最优参数。

但是，Adam 也有一些缺点。例如，Adam 对于噪声较大的梯度可能会表现不稳定，并且在处理非凸问题时可能会出现性能下降的情况。

### **9、推荐系统中，如何进行负采样？**

在推荐系统中，负采样是一种重要的技术，用于构造负样本，以便训练推荐模型。一般而言，推荐系统的正样本很少，负样本很多，因此负采样可以大大加快训练的速度和提高模型的准确率。

负采样的过程通常可以分为以下几个步骤：

* 计算每个物品的权重：对于每个物品，根据其流行度或其他相关因素计算其权重，权重越高的物品越容易被采样为负样本。
* 根据权重进行采样：根据每个物品的权重进行随机采样，以得到一定数量的负样本。采样的过程通常可以使用一些概率分布，如指数分布或幂律分布等。
* 去除已有的正样本：从采样得到的负样本中去除已有的正样本，以确保训练集中不会出现重复样本。
* 控制负采样比例：为了保持正负样本之间的平衡，通常需要控制负采样的比例，即每个正样本对应的负样本数量。
* 对于多个用户和物品的推荐问题，可以通过对每个用户和物品分别进行采样来生成一定数量的正负样本对。

# **华为NLP算法工程师面试题9道** 2023-04-28 16:11

### **1、NLP中常见的分词方法有哪些？**

常见的中文分词方法包括基于规则、基于统计和基于深度学习的方法。其中，基于规则的方法根据预先定义的规则对文本进行切分；基于统计的方法通过统计某个词在语料库中出现的概率来进行分词；基于深度学习的方法则利用深度神经网络模型从大规模语料中学习分词模型。

### **2、讲一下BERT的结构？**

BERT（Bidirectional Encoder Representations from Transformers）是一种基于Transformer模型的预训练语言模型。其结构由多层Transformer编码器组成，其中每层包含多头自注意力机制和前馈神经网络。BERT还采用了双向训练策略，使得模型能够在不同层次、不同粒度下理解输入序列中的上下文信息。

### **3、自然语言处理有哪些任务？**

自然语言处理任务包括文本分类、命名实体识别、情感分析、机器翻译、文本生成、问答系统等。

### **4、L1，L2正则化的区别，岭回归是L1正则化还是L2正则化？**

L1正则化和L2正则化都是用来约束模型的复杂度，以避免过拟合。L1正则化通过将模型参数的绝对值之和作为正则项，使得一些参数变为0，从而达到特征选择的目的；L2正则化通过将模型参数的平方和作为正则项，使得参数的值都变小，从而使模型更加稳定。岭回归是一种使用L2正则化的线性回归模型。

### **5、怎么处理类别不平衡？**

类别不平衡问题可以通过过采样、欠采样、生成新样本、集成学习等方法来解决。过采样方法包括随机过采样、SMOTE等；欠采样方法包括随机欠采样、Tomek Links等；生成新样本方法包括GAN、VAE等；集成学习方法包括Bagging、Boosting等。

### **6、模型提速的方法有哪些？**

模型提速的方法包括模型压缩、剪枝、量化、分布式训练等。模型压缩包括权重共享、低秩近似、深度可分离卷积等方法；剪枝包括通道剪枝、结构剪枝、权重剪枝等；量化包括权重量化、激活量化等；分布式训练则是利用多个计算节点共同完成模型训练任务，加速训练过程。

### **7、了解数据挖掘的方法嘛？**

数据挖掘包括数据预处理、特征工程、模型构建和模型评估等步骤。其中，数据预处理包括数据清洗、数据集成、数据变换和数据规约；特征工程包括特征提取、特征选择和特征构建；模型构建包括选择合适的模型和模型参数调优；模型评估包括模型效果评估和模型泛化能力评估。

### **8、了解对比学习嘛？**

对比学习是一种无监督学习方法，通过训练模型使得相同样本的表示更接近，不同样本的表示更远离，从而学习到更好的表示。对比学习通常使用对比损失函数，例如Siamese网络、Triplet网络等，用于学习数据之间的相似性和差异性。

### **9、说一下广度优先遍历和深度优先遍历？**

广度优先遍历（BFS）是一种图形搜索算法，从起点开始，依次访问与起点相邻的所有节点，再访问与这些节点相邻的所有未访问过的节点，直到找到目标节点或者所有节点都被访问。广度优先遍历使用队列来保存访问过的节点。

深度优先遍历（DFS）是一种图形搜索算法，从起点开始，一直访问相邻节点，直到达到最深的节点，再返回上一级节点，继续访问其他未访问过的节点，直到找到目标节点或者所有节点都被访问。深度优先遍历使用栈来保存访问过的节点。与广度优先遍历相比，深度优先遍历更适用于搜索深度较深的图形

# **SHEIN算法工程师暑期实习面试题8道**

### **1、如何理解交叉熵的物理意义**

交叉熵是一种用于比较两个概率分布之间的差异的指标。在机器学习中，它通常用于比较真实标签分布与模型预测分布之间的差异。

### **2、过拟合如何去解决？L1正则为什么能够使得参数稀疏，从求导的角度阐述。**

过拟合的解决方法有很多：

数据的角度：获取和使用更多的数据（数据集增强）；

模型角度：降低模型复杂度、L1\L2\Dropout正则化、Early stopping（提前终止）

模型融合的角度：使用bagging等模型融合方法。

L1正则化在损失函数中加入参数的绝对值之和，可以使得一些参数变得非常小或者为零，从而使得模型更加稀疏，减少过拟合的风险。从求导的角度上看，L1正则化的导数在某些点处为零，从而使得参数变得非常小或者为零，达到了稀疏的效果。

### **3、类别不平衡是如何去处理的？如果进行采样，策略是什么？**

类别不平衡可以通过对数据进行采样来处理。一种常用的采样策略是欠采样，即随机从多数类中选择一部分样本使得样本平衡。另一种常用的采样策略是过采样，即从少数类中复制一些样本使得样本平衡。还有一种策略是生成新的少数类样本，比如SMOTE算法，其中通过对少数类样本进行随机插值来生成新的样本。

### **4、介绍一下attention，做过哪些尝试和改进。**

Attention是一种机器学习中的技术，主要用于提取输入序列中的关键信息。在自然语言处理和图像处理中，Attention机制已经成为了重要的技术。对于Attention的一些实现方法和改进，一种常见的Attention实现方法是Soft Attention，它可以用于提取序列数据中的重要信息。另外，还有一些改进方法，比如Multi-Head Attention和Self-Attention等，可以进一步提高Attention的性能。

### **5、对于一个时间顺序的推荐数据，如何划分训练集和验证集，能不能随机？**

对于时间顺序的推荐数据，通常可以使用时间轴来划分训练集和验证集。具体地，可以选取一段时间作为训练集，另一段时间作为验证集。如果数据量足够大，也可以将数据随机划分为训练集和验证集。但是，需要注意的是，在时间序列数据中，训练集和验证集应该按照时间顺序进行划分，以保证模型的泛化能力。

### **6、欠拟合如何去解决，训练过程不收敛如何去解决？**

欠拟合的解决方法有很多，其中一个是增加模型的复杂度。可以增加模型的参数数量、增加网络层数、使用更复杂的模型结构等来提高模型的拟合能力。另外，还可以尝试调整学习率、修改损失函数、增加训练数据等方法。如果训练过程不收敛，可能是学习率过大或者网络结构不合理导致的。可以尝试减小学习率、使用不同的优化器、增加网络层数等方法来解决这个问题。

### **7、正则化和最大似然的关系。**

正则化和最大似然有一定的关系。最大似然是一种用于估计模型参数的方法，其目标是找到使得观测数据出现的概率最大的模型参数。正则化是一种对模型参数进行限制的方法，可以使得模型参数更加稳定和泛化能力更强。在最大似然估计中，通过添加正则化项可以达到类似的目的，即防止模型过拟合。常见的正则化方法包括L1正则化和L2正则化。

### **8、Leetcode：数组中第K大的元素。**

难度：【中等】

三种思路：一种是直接使用sorted函数进行排序，一种是使用小顶堆，一种是使用快排（双指针 + 分治）。

**方法一：直接使用sorted函数进行排序**

代码如下：

class Solution:

def findKthLargest(self, nums: List[int], k: int) -> int:

return sorted(nums, reverse = True)[k-1]

**方法二：使用堆**

维护一个size为 k 的小顶堆，把每个数丢进去，如果堆的 size > k，就把堆顶pop掉（因为它是最小的），这样可以保证堆顶元素一定是第 k 大的数。

代码如下：

class Solution:

def findKthLargest(self, nums: List[int], k: int) -> int:

heap = []

for num in nums:

heappush(heap,num)

if len(heap) > k:

heappop(heap)

return heap[0]

时间复杂度：O(nlogk)

空间复杂度：O(k)

**方法三：双指针 + 分治**

partition部分

定义两个指针left 和 right，还要指定一个中心pivot（这里直接取最左边的元素为中心，即 nums[i]）

不断将两个指针向中间移动，使得大于pivot的元素都在pivot的右边，小于pivot的元素都在pivot的左边，注意最后满足时，left是和right相等的，因此需要将pivot赋给此时的left或right。

然后再将中心点的索引和 k - 1 进行比较，通过不断更新left和right找到最终的第 k 个位置。

代码如下：

class Solution:

def findKthLargest(self, nums: List[int], k: int) -> int:

left, right, target = 0, len(nums)-1, k-1

while True:

pos = self.partition(nums, left, right)

if pos == target:

return nums[pos]

elif pos > target:

right = pos -1

else:

left = pos + 1

def partition(self, nums, left, right):

pivot = nums[left]

while left < right:

while nums[right] <= pivot and left < right:

right -=1

nums[left] = nums[right]

while nums[left] >= pivot and left < right:

left +=1

nums[right] = nums[left]

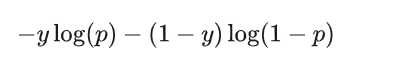
nums[left] = pivot

return left

# **趣玩科技公司推荐算法工程师面试题9道**

### **1、二分类的分类损失函数？**

二分类的分类损失函数一般采用交叉熵（Cross Entropy）损失函数，即CE损失函数。二分类问题的CE损失函数可以写成：



其中，y是真实标签，p是预测标签，取值为0或1。

### **2、多分类的分类损失函数(Softmax)？**

多分类的分类损失函数采用Softmax交叉熵（Softmax Cross Entropy）损失函数。Softmax函数可以将输出值归一化为概率分布，用于多分类问题的输出层。Softmax交叉熵损失函数可以写成：

−∑�=1���log⁡(��)

其中，n是类别数，yi是第i类的真实标签，pi是第i类的预测概率。

### **3、关于梯度下降的sgdm,adagrad，介绍一下。**

SGD（Stochastic Gradient Descent）是最基础的梯度下降算法，每次迭代随机选取一个样本计算梯度并更新模型参数。SGDM（Stochastic Gradient Descent with Momentum）在SGD的基础上增加了动量项，可以加速收敛。Adagrad（Adaptive Gradient）是一种自适应学习率的梯度下降算法，它根据每个参数的梯度历史信息调整学习率，可以更好地适应不同参数的变化范围。

### **4、为什么不用MSE分类用交叉熵？**

MSE（Mean Squared Error）损失函数对离群点敏感，而交叉熵（Cross Entropy）损失函数在分类问题中表现更好，因为它能更好地刻画分类任务中标签概率分布与模型输出概率分布之间的差异。

### **5、yolov5相比于之前增加的特性有哪些？**

YOLOv5相比于之前版本增加了一些特性，包括：使用CSP（Cross Stage Partial）架构加速模型训练和推理；采用Swish激活函数代替ReLU；引入多尺度训练和测试，以提高目标检测的精度和召回率；引入AutoML技术，自动调整超参数以优化模型性能。

### **6、可以介绍一下attention机制吗？**

Attention机制是一种用于序列建模的技术，它可以自适应地对序列中的不同部分赋予不同的权重，以实现更好的特征表示。在Attention机制中，通过计算查询向量与一组键值对之间的相似度，来确定每个键值对的权重，最终通过加权平均的方式得到Attention向量。

### **7、关于attention机制，三个矩阵KQ,KV,K..的作用是什么？**

在Attention机制中，KQV是一组与序列中每个元素对应的三个矩阵，其中K和V分别代表键和值，用于计算对应元素的权重，Q代表查询向量，用于确定权重分配的方式。三个矩阵K、Q、V在Attention机制中的具体作用如下：

K（Key）矩阵：K矩阵用于计算每个元素的权重，是一个与输入序列相同大小的矩阵。通过计算查询向量Q与每个元素的相似度，确定每个元素在加权平均中所占的比例。

Q（Query）向量：Q向量是用来确定权重分配方式的向量，与输入序列中的每个元素都有一个对应的相似度，可以看作是一个加权的向量。

V（Value）矩阵：V矩阵是与输入序列相同大小的矩阵，用于给每个元素赋予一个对应的特征向量。在Attention机制中，加权平均后的向量就是V矩阵的加权平均向量。

通过K、Q、V三个矩阵的计算，Attention机制可以自适应地为输入序列中的每个元素分配一个权重，以实现更好的特征表示。

### **8、介绍一下文本检测EAST。**

EAST（Efficient and Accurate Scene Text）是一种用于文本检测的神经网络模型。EAST通过以文本行为单位直接预测文本的位置、方向和尺度，避免了传统方法中需要多次检测和合并的过程，从而提高了文本检测的速度和精度。EAST采用了一种新的训练方式，即以真实文本行作为训练样本，以减少模型对背景噪声的干扰，并在测试阶段通过非极大值抑制（NMS）算法进行文本框的合并。

### **9、编程题(讲思路)：给定两个字符串s,t，在s字符串中找到包含t字符串的最小字串。**

给定两个字符串s、t，可以采用滑动窗口的方式在s中找到包含t的最小子串。具体做法如下：

（1）定义两个指针left和right，分别指向滑动窗口的左右边界。

（2）先移动right指针，扩展滑动窗口，直到包含了t中的所有字符。

（3）移动left指针，缩小滑动窗口，直到无法再包含t中的所有字符。

（4）记录当前滑动窗口的长度，如果小于之前记录的长度，则更新最小长度和最小子串。

（5）重复（2）到（4）步骤，直到right指针到达s的末尾为止。

# **知乎NLP算法工程师面试题7道**

## **1、生成式模型与判别式模型的区别？**

**生成式模型**先对数据的联合分布 进行建模，然后再通过贝叶斯公式计算样本属于各类别的后验概率 。

**判别式模型**直接进行条件概率建模，由数据直接学习决策函数 或条件概率分布作为预测的模型。**判别方法不关心背后的数据分布，关心的是对于给定的输入，应该预测什么样的输出。**

用一句话总结就是生成模型估计的是**联合概率分布**，判别式模型估计的是**条件概率分布**。

## **2、模型的方差和偏差是指什么？**

偏差（Bias）：表示模型的预测值和真实值之间的差异程度。用所有可能的训练数据集 训练出的所有模型的输出的平均值与真实模型的输出值之间的差异。如果模型的偏差很高，说明模型无法捕捉到数据中的所有信息，因此模型的预测值和真实值之间的差异会很大。例如，在房屋预测的例子中，如果模型只考虑了房屋的面积这一个特征，那么模型的偏差就会很高，因为模型无法考虑到其他特征（比如卧室数量、浴室数量等）对房价的影响。

方差（Variance）：表示模型对于不同的训练集的预测结果分散程度。如果模型的方差很高，说明模型对于不同的训练集的预测结果差异很大，也就是说，模型无法捕捉到数据中的一般规律，而过度拟合了训练数据。例如，在房屋预测的例子中，如果我们使用非常复杂的模型（比如高阶多项式回归），那么模型的方差就会很高，因为模型过度拟合了训练数据，而无法捕捉到一般规律。

## **3、二分类模型的评估指标有哪些？**

在二分类模型中，我们通常将样本分为两类：正例和反例。在评估模型性能时，我们需要考虑以下四个指标：

准确率（Accuracy）：预测正确的样本占总样本数的比例。

精确率（Precision）：预测为正例的样本中，真正为正例的比例。

召回率（Recall）：真正为正例的样本中，被预测为正例的比例。

F1值：精确率和召回率的调和平均数，常用于综合评估模型性能。

此外，我们还可以使用ROC曲线和AUC来评估模型的性能。ROC曲线是以假正率（False Positive Rate, FPR）为横坐标，真正率（True Positive Rate, TPR）为纵坐标绘制的曲线，用于衡量模型在不同阈值下的性能。AUC则是ROC曲线下的面积，用于综合评估模型的性能。AUC的取值范围为0.5到1，值越大，模型性能越好。

## **4、AUC刻画的什么？说明了什么意思？**

AUC（Area Under the ROC Curve）是机器学习领域中常用的一个性能度量指标，用于衡量分类器对样本的分类能力。ROC（Receiver Operating Characteristic）曲线是以假正率（False Positive Rate, FPR）为横坐标，真正率（True Positive Rate, TPR）为纵坐标绘制的曲线，用于衡量模型在不同阈值下的性能。AUC则是ROC曲线下的面积，用于综合评估模型的性能。AUC的取值范围在0.5到1之间，值越大，分类器性能越好。

具体来讲，AUC刻画了分类器对正负样本的分类能力，即对于一个随机的正负样本对，分类器给出正样本的预测值高于负样本的概率。AUC越高，说明分类器在区分正负样本时具有更好的能力。在实际场景中，例如医学诊断、金融风控等领域，AUC是评估分类器性能的重要指标之一。

需要注意的是，AUC并不是一个完美的指标，因为它并不能告诉我们分类器在不同的阈值下的表现，而且对于类别不平衡的数据集，AUC可能会给出误导性的结果。因此，在实际应用中，我们需要综合考虑多个性能度量指标来评估分类器的性能。

## **5、交叉熵函数刻画的什么？**

交叉熵函数是机器学习中常用的一种损失函数。它常用于评估两个概率分布之间的差异，尤其是在神经网络中用于度量模型输出与真实标签之间的差距。交叉熵函数的基本思想是将模型的输出概率与真实标签的概率进行比较，通过计算两个概率分布之间的交叉熵（cross-entropy）来衡量它们的相似程度。在分类任务中，我们通常将标签表示为一个one-hot向量，即只有一个元素为1，其余元素都为0。

## **6、对于非常大的分类类别，对于softmax有哪些优化方法？**

当分类类别非常大时，计算softmax的复杂度会变得非常高，因为softmax的计算复杂度是与类别数成正比的。因此，需要采用一些优化方法来加速softmax的计算。

一种常见的优化方法是基于交叉熵的方法，这包括hierarchical softmax和noise-contrastive estimation（NCE）。hierarchical softmax通过将所有的类别组织成一棵二叉树，并对每个节点定义一个概率值，从而将原问题转化为对一系列二分类问题的求解。这样就可以减少softmax计算的次数，从而提高运行速度。NCE是一种用于训练softmax模型的替代方法，它通过最大化正确标签和噪声标签之间的差异来训练模型。与传统的softmax方法不同，NCE不需要计算所有类别的概率，因此在类别数非常大时可以大大提高运行速度。

另一种优化softmax的方法是采用近似softmax方法，例如采用采样的方法进行训练。这包括采样自适应重要性抽样（sampled softmax）、target sampling和sparsemax等。sampled softmax通过随机采样一小部分类别来近似计算softmax概率，从而减少计算量。target sampling是一种基于重要性采样的方法，它根据类别的频率对类别进行采样，从而减少少数类别的影响。sparsemax是一种类似于softmax的激活函数，但是它产生的分布更加稀疏，因此可以减少计算量。

总之，当我们需要处理非常大的分类问题时，需要采用一些优化方法来加速softmax的计算，从而提高运行速度和效率。不同的优化方法有不同的适用场景，需要根据具体情况进行选择。

## **7、softmax除了作为激活函数，在深度学习中还有哪些用途？**

除了作为激活函数，还可以使用softmax进行软归一化，而且在transformer中，softmax在注意力机制中计算注意力，主要是软最大化。

# **携程NLP算法工程师面试题9道**

## **1、可以解释一下熵吗，它的公式怎么算的？**

熵可以用来描述信息的不确定性或信息的随机程度。

熵的公式为：

H = -Σ(p \* log(p))

其中，H表示熵，p表示每种可能事件发生的概率，log表示以2为底的对数。

## **2、BERT的base版本的原始模型，训练的时候，第一个epoch模型的判定结果很可能是错的，这个时候熵还可信吗？**

在训练BERT模型时，通常需要多个epoch来达到最佳性能。在初始训练过程中，模型的判定结果可能不是非常准确，因此第一个epoch的结果可能会存在误差。但即使在初始阶段，BERT模型的熵仍然是可信的。熵是衡量信息的不确定性的量，可以用来衡量模型的预测结果的置信度。

在第一个epoch，BERT模型的熵也可以帮助我们评估模型的性能和预测结果的置信度。当熵较低时，表示模型对于给定的输入文本的预测结果比较确定；当熵较高时，则表示模型对于该输入文本的预测结果不确定。因此，即使在第一个epoch，熵仍然是一种可信的评估模型性能的指标。

## **3、交叉熵和KL散度有什么关系？**

交叉熵和KL（Kullback-Leibler）散度都是用于度量两个概率分布之间的差异性的指标，它们有一些相似之处，但也有不同之处。

相同处：a. 都不具备对称性 b. 都是非负的

区别：KL散度是交叉熵与熵的差。

## **4、BERT的缺点？可以从预训练方法角度解答。**

训练时间长：BERT模型需要巨大的计算资源和时间来进行预训练，尤其是在较大的语料库上进行训练时。这使得BERT对于小规模数据集的任务可能不是很适合。

处理长文本困难：BERT对于长文本的处理存在困难，因为输入文本长度有限制，而且BERT的self-attention机制需要计算所有输入词的相互交互，这样的计算开销也很大。

## **5、RoBERTa相比BERT有哪些改进？**

预训练数据更多：RoBERTa使用了更多的训练数据，并且在预训练中采用了更多的数据增强技术，这使得RoBERTa在多个自然语言处理任务上的表现优于BERT。

去掉了NSP任务：RoBERTa将原本的下一句预测任务去掉了，只保留了遮蔽语言模型任务，这使得RoBERTa能够更好地处理单个句子的表示。

动态掩码：RoBERTa使用了一种动态遮蔽策略，通过在每次训练迭代中随机选择遮蔽哪些词语来增加模型的鲁棒性。

## **6、BERT的输入有哪几种Embedding？**

Token Embedding：将每个输入词转化为其对应的WordPiece嵌入向量。

Segment Embedding：对于一组输入文本，将其分为两个部分，并为每个部分分别分配一个segment ID，以区分不同的输入文本。

Position Embedding：为了捕捉输入词的相对位置，BERT使用了位置嵌入向量，对于不同的位置位置，使用不同的嵌入向量表示。

## **7、你有了解其他模型去尝试解决长度限制的方案吗？**

Bert模型的长度限制问题主要是由于Transformer结构中的自注意力机制（self-attention mechanism）和位置嵌入（position embeddings）所导致的。这些机制使得Bert对于较长的序列处理非常耗时，并且占用大量的内存，从而限制了Bert在处理长序列任务上的性能。

为了解决这个问题，一些研究人员提出了一些改进型的模型，包括：

Longformer：Longformer是一个基于Transformer结构的模型，它使用了一种新的自注意力机制，称为"Sliding Window Attention"，该机制可以在处理长序列时缓解Bert模型的计算和存储成本。

Reformer：Reformer是一个基于哈希注意力（Hashing Attention）的Transformer模型，该模型可以有效地处理长序列，并且在一些NLP任务上表现良好。

Performer：Performer是一种基于FFT（Fast Fourier Transform）的Transformer模型，该模型可以处理长序列，并且在一些NLP任务上表现良好。

Sparse Transformer：Sparse Transformer是一种使用稀疏注意力机制的Transformer模型，它可以减少Bert模型在处理长序列时的计算和存储成本。

## **8、BERT是怎么缓解梯度消失的？**

Layer Normalization（LN）：在每个Transformer模块中，BERT使用LN来规范化每个词嵌入的向量值，使其具有相同的均值和方差。这有助于提高梯度的流动性，减少梯度消失的可能性。

Residual Connections：BERT使用残差连接将每个Transformer模块的输入和输出相加，以便梯度能够更容易地传播到较早的层。这也有助于缓解梯度消失问题。

## **9、LN和BN的区别？**

Layer Normalization（LN）：在神经网络中，LN是一种用于规范化输入向量的技术。LN基于每个输入的样本进行标准化，而不是使用整个批次的统计信息，这使得LN对于小批量输入数据也能提供相对稳定的标准化。另外，LN通常应用在RNN或Transformer等网络结构中的每个层上，而不是在整个网络中的一层上。

Batch Normalization（BN）：BN是一种在神经网络中用于标准化输入向量的技术。与LN不同，BN使用整个批次的统计信息来标准化输入向量。BN通常应用在卷积层或全连接层上，以减少梯度消失问题并提高模型的鲁棒性。在训练过程中，BN会维护每个批次的均值和方差，用于标准化输入数据。在推理时，BN使用训练期间学习的统计信息进行标准化。

# **某公司NLP算法工程师面试题8道**

## **1、分类场景下 bert和gpt+prompt的方式哪种会有更好效果，为什么。**

在分类场景下，BERT比GPT更适合用于建模，因为BERT的结构中包含了双向的Transformer编码器，而GPT的结构中只包含单向的Transformer解码器。这使得BERT能够更好地捕捉文本中的双向上下文信息，从而在文本分类任务中表现更好。而GPT+prompt的方式则可以通过在输入文本前加入一些提示语来指导模型学习特定任务的表示，从而提高模型的泛化性能。

## **2、如何解决prompt泛化性**

* 使用多个不同的prompt，从而增加模型学习的样本多样性。
* 通过在prompt中添加随机噪声或变换，来增加数据集的丰富性，从而提高模型的泛化性能。
* 采用迁移学习或元学习等方法，从先前学习的任务中提取知识，并将其应用于新的任务中。

## **3、对COT(Chain-of-Thought Prompt)和Instruction Tuning的理解**

COT Prompt是一种用于自然语言生成的提示机制，它可以通过将一个文本片段（例如一段文章或一个问题）分解为一系列相关的语义单元，从而帮助自然语言生成模型更准确地理解文本的意义。这些语义单元可以按照它们的层次结构（例如句子、段落或章节）进行组织，并且可以用于为模型提供关于生成文本的结构和内容的提示。

Instruction Tuning是一种用于调整自然语言生成模型的参数的技术，它可以通过向模型提供来自人类编辑或专家的指令来指导模型的生成过程。这些指令可以涵盖各种不同的方面，例如语法、风格、结构和内容等。通过使用Instruction Tuning，可以在不改变模型架构的情况下对其进行微调，从而使其更好地满足特定的生成需求。

## **4、关于bert的后续改进工作，分别改进了哪些地方？**

1. BERT的后续改进工作主要包括以下方面：

* 基于BERT的预训练模型的改进，例如RoBERTa、ALBERT等。
* 通过调整BERT的架构和超参数来进一步优化模型性能，例如Electra、DeBERTa等。
* 改进BERT在特定任务上的应用方法，例如ERNIE、MT-DNN等。

## **5、对知识蒸馏知道多少，有哪些改进用到了？**

知识蒸馏是一种通过将一个复杂模型的知识转移到一个简单模型来提高简单模型性能的方法。这种方法已经被广泛应用于各种深度学习任务中。其中一些改进包括：

* 使用不同类型的损失函数和温度参数来获得更好的知识蒸馏效果。
* 引入额外的信息来提高蒸馏的效果，例如将相似性约束添加到模型训练中。
* 将蒸馏方法与其他技术结合使用，例如使用多任务学习和迁移学习来进一步改进知识蒸馏的效果。

## **6、attention计算复杂度以及如何改进。**

在标准的Transformer中，attention计算的时间复杂度为O(N^2)，其中N是输入序列的长度。为了降低计算复杂度，可以采用以下几种方法：

* 使用自注意力机制，减少计算复杂度。自注意力机制不需要计算输入序列之间的交叉关系，而是计算每个输入向量与自身之间的关系，从而减少计算量。
* 使用局部注意力机制，只计算输入序列中与当前位置相关的子序列的交互，从而降低计算复杂度。
* 采用基于近似的方法，例如使用随机化和采样等方法来近似计算，从而降低计算复杂度。
* 使用压缩注意力机制，通过将输入向量映射到低维空间来减少计算量，例如使用哈希注意力机制和低秩注意力机制等。

## **7、谈一下对模型量化的了解。**

1. 模型量化是一种将浮点型参数转换为定点型参数的技术，以减少模型的存储和计算复杂度。常见的模型量化方法包括：

* 量化权重和激活值，将它们转换为整数或小数。
* 使用更小的数据类型，例如8位整数、16位浮点数等。
* 使用压缩算法，例如Huffman编码、可逆压缩算法等。

模型量化可以减少模型的存储空间和内存占用，同时也可以加速模型的推理速度。但是，模型量化可能会对模型的精度造成一定的影响，因此需要仔细权衡精度和计算效率之间的平衡。

## **8、topk数组取值，尽可能多的方法。**

1. 对于一个长度为N的数组，获取其中前K个最大或最小值的方法有很多种，其中一些常见的方法包括：

* 直接排序，将整个数组排序，然后取前K个或后K个元素。
* 使用堆，维护一个大小为K的最大或最小堆，然后遍历数组并将元素插入到堆中，最后取出堆中的元素。
* 分治算法，将数组分成多个小的部分，然后对每个部分分别计算前K个或后K个元素，最后将结果合并。
* 快速选择算法，类似于快速排序，通过每次选择一个基准值来将数组分成两部分，然后根据基准值所在的位置递归地对其中一部分进行处理。使用快排变种算法，例如三向切分快排，通过将数组分成小于、等于和大于基准值的三部分来减少递归次数。
* 桶排序，对于已知取值范围的数组，可以使用桶排序来计算前K个或后K个元素。
* 部分排序算法，例如选择排序和插入排序，可以只对数组的一部分进行排序，从而减少计算量。
* 分层采样算法，将数组分成多个层次，每个层次采样一定比例的元素，然后通过分层采样的结果来估计整个数组的前K个或后K个元素。

对于长度较大的数组，使用基于排序的方法可能会非常慢，而使用堆和快速选择算法则可以在较短的时间内获得结果。

# **海康-CV算法工程师实习面试题6道**

## **1、YOLO的正负样本是什么？**

在YOLO算法中，目标物体被定义为一组边界框，也称为锚框（anchor box）。每个锚框都由其中心点坐标、宽度和高度来定义。对于每个锚框，计算它与所有目标物体的IoU（交并比）值。如果IoU大于某个阈值（如0.5），则将该锚框标记为正样本，否则将其标记为负样本。

## **2、模型压缩和加速的方法有哪些？**

参数剪枝（Parameter Pruning）：删除模型中冗余的参数，减少模型的大小。通常情况下，只有很少一部分参数对模型的性能贡献较大，其余参数对性能的贡献较小或没有贡献，因此可以删除这些冗余参数。

量化（Quantization）：将浮点型参数转换为更小的整数或定点数，从而减小模型大小和内存占用，提高计算效率。

知识蒸馏（Knowledge Distillation）：利用一个较大、较准确的模型的预测结果来指导一个较小、较简单的模型学习。这种方法可以减小模型的复杂度，提高模型的泛化能力和推理速度。

网络剪枝（Network Pruning）：删除模型中冗余的神经元，从而减小模型的大小。与参数剪枝不同，网络剪枝可以删除神经元而不会删除对应的参数。

蒸馏对抗网络（Distillation Adversarial Networks）：在知识蒸馏的基础上，通过对抗训练来提高模型的鲁棒性和抗干扰能力。

模型量化（Model Quantization）：将模型的权重和激活函数的精度从32位浮点数减少到更小的位数，从而减小模型的大小和计算开销。

层次化剪枝（Layer-wise Pruning）：对模型的不同层进行不同程度的剪枝，以实现更高效的模型压缩和加速。

低秩分解（Low-Rank Decomposition）：通过将一个较大的权重矩阵分解为几个较小的权重矩阵，从而减少计算开销。

卷积分解（Convolution Decomposition）：将卷积层分解成几个更小的卷积层或全连接层，以减小计算开销。

网络剪裁（Network Trimming）：通过对模型中一些不重要的连接进行剪裁，从而减小计算开销。

## **3、半精度是什么？**

半精度是指使用16位二进制浮点数（half-precision floating point）来表示数字的数据类型，可以加速计算和减小内存占用。

## **4、半精度的理论原理是什么？**

半精度使用16位二进制浮点数来表示数字，其中1位表示符号位，5位表示指数，10位表示尾数。相比于单精度（32位）和双精度（64位）的浮点数，半精度的表示范围和精度更小，但可以通过降低内存占用和加速计算来实现高效的运算。

## **5、你了解的知识蒸馏模型有哪些？**

FitNets：使用一个大型模型作为教师模型来指导一个小型模型的训练。

Hinton蒸馏：使用一个大型模型的输出作为标签来指导一个小型模型的训练。

Born-Again Network（BAN）：使用一个已经训练好的模型来初始化一个新模型，然后使用少量的数据重新训练模型。

TinyBERT：使用一个大型BERT模型作为教师模型来指导一个小型BERT模型的训练。

## **6、自监督、半监督、无监督的区别？**

自监督学习：使用输入数据的某些属性（例如，数据本身的结构或某些隐含信息）来作为监督信号，从而避免了手动标注的成本。例如，图像数据可以通过旋转、剪切等方式进行扩增，并使用数据自身的变换作为监督信号来训练模型。

半监督学习：是指使用有标注和无标注的数据来训练模型。通常情况下，有标注的数据只是无标注数据的一个子集。通过同时使用有标注和无标注数据进行训练，可以提高模型的性能和泛化能力。

无监督学习：是指在没有标注数据的情况下，通过分析数据本身的结构、模式和相关性来学习模型。无监督学习的目标是从数据中发现一些有用的结构，例如聚类、降维、密度估计等。常见的无监督学习方法包括自编码器、生成对抗网络、变分自编码器等。与监督和半监督学习不同，无监督学习不需要手动标注数据，因此可以处理大量未标注的数据，从而提高数据利用率和模型性能。

# **小米AIGC大模型实习面试题4道**

### **1、讲一下transformer的原理**

transformer的主要组成部分包括自注意力机制和位置编码。

* 自注意力机制（Self-Attention）：它允许模型在输入序列中的不同位置之间建立关联。通过计算每个位置与其他所有位置的相关性得分，模型可以学习到输入序列中各个部分之间的依赖关系。
* 位置编码（Positional Encoding）：Transformer通过将位置信息编码到输入向量中，使得模型能够区分不同位置的词语或标记。这样，模型能够利用位置信息更好地理解输入序列。
* 多头注意力机制（Multi-Head Attention）：为了增强模型对不同表示空间的关注，Transformer引入了多个注意力头，允许模型并行地关注不同的表示子空间，从而提高了模型的表征能力。
* 前馈神经网络（Feed-Forward Neural Network）：每个注意力模块之后都连接着一个前馈神经网络，它通过应用全连接层和激活函数来对特征进行变换和组合。

通过这些组件的组合和堆叠，Transformer模型能够捕捉长距离依赖关系，同时避免了传统循环神经网络中的序列依赖问题，使其在处理长序列时表现更好。

### **2、讲一下微调方法p-tuning v2的原理**

P-Tuning V2在P-Tuning V1的基础上进行了下述改进：

* 在每一层都加入了Prompts tokens作为输入，而不是仅仅加在输入层，这与Prefix Tuning的做法相同。这样得到了更多可学习的参数，且更深层结构中的Prompt能给模型预测带来更直接的影响。
* 去掉了重参数化的编码器。在 P-tuning v2 中，作者发现重参数化的改进很小，尤其是对于较小的模型，同时还会影响模型的表现。
* 针对不同任务采用不同的提示长度。提示长度在提示优化方法的超参数搜索中起着核心作用。在实验中，我们发现不同的理解任务通常用不同的提示长度来实现其最佳性能，这与Prefix-Tuning中的发现一致，不同的文本生成任务可能有不同的最佳提示长度。
* 可选的多任务学习。先在多任务的Prompt上进行预训练，然后再适配下游任务。一方面，连续提示的随机惯性给优化带来了困难，这可以通过更多的训练数据或与任务相关的无监督预训练来缓解；另一方面，连续提示是跨任务和数据集的特定任务知识的完美载体。

### **3、什么是bilstm-crf**

BiLSTM-CRF是一种用于序列标注任务的神经网络模型，结合了双向长短期记忆网络（BiLSTM）和条件随机场（CRF）。

* 双向长短期记忆网络（BiLSTM）：BiLSTM是一种循环神经网络变种，能够有效地捕捉输入序列中的长期依赖关系。通过同时考虑输入序列的正向和反向信息，BiLSTM可以更好地理解整个序列的语境。
* 条件随机场（CRF）：CRF是一种概率图模型，常用于序列标注任务。它能够对序列中的标记进行全局建模，利用标记之间的依赖关系来提高标注的一致性和准确性。

BiLSTM-CRF模型首先利用BiLSTM对输入序列进行特征提取，然后将提取的特征作为CRF模型的输入，利用CRF模型对序列进行标注。通过结合BiLSTM和CRF，BiLSTM-CRF模型能够更好地处理序列标注任务，同时充分利用上下文信息和标记之间的依赖关系。

### **4、代码题：给整数数组和target，找数组当中和为目标值的两个整数返回下标。**

使用一个字典来存储遍历过的数字及其下标，通过查找字典来判断是否存在与当前数字配对的数字，从而实现寻找和为目标值的两个数的功能。

def two\_sum(nums, target):

num\_dict = {} # 用字典存储数字和对应的下标

for i, num in enumerate(nums):

complement = target - num # 计算目标值与当前值的差值

if complement in num\_dict: # 如果差值在字典中，则找到了符合条件的两个数

return [num\_dict[complement], i]

num\_dict[num] = i # 否则将当前数字及其下标存入字典

return None # 如果未找到符合条件的两个数，则返回None

# 示例

nums = [2, 7, 11, 15]

target = 9

print(two\_sum(nums, target)) # 输出 [0, 1]，因为 nums[0] + nums[1] = 2 + 7 =

# **美团营销算法实习面试题9道**

## **1、如何解决过拟合和欠拟合。**

过拟合（Overfitting）：

* 增加数据量：通过增加训练数据来减少模型对特定数据的过度拟合。
* 简化模型：减少模型的复杂度，可以通过减少特征数量、降低多项式次数等方式。
* 正则化：引入正则化项，如L1或L2正则化，以惩罚模型复杂度。

欠拟合（Underfitting）：

* 增加特征：添加更多有意义的特征，提高模型的表达能力。
* 增加模型复杂度：选择更复杂的模型，如增加层数、节点数等。
* 减小正则化：减小正则化的程度，以允许模型更好地适应数据。

## **2、L1正则化和L2正则化的区别。**

L1正则化：

* 增加的正则化项为权重向量的绝对值之和。
* 促使模型参数变得稀疏，即某些权重变为零，从而实现特征选择的效果。

L2正则化：

* 增加的正则化项为权重向量的平方和。
* 通过减小权重的同时保持它们都非零，对权重进行平滑调整。

区别：

* L1正则化倾向于产生稀疏权重，对于特征选择有利；
* L2正则化则更倾向于在所有特征上产生较小但非零的权重。

## **3、什么是交叉验证。**

一种评估模型性能的统计学方法，通常用于训练样本有限的情况。将数据集划分为多个子集，多次训练模型，并在每次训练中使用不同的子集作为验证集，其余作为训练集。常见的交叉验证方法包括k折交叉验证和留一法。

## **4、随机森林的原理和它的作用。**

原理：

* 由多个决策树组成，每个决策树基于随机抽取的数据和特征进行训练。
* 通过投票或平均等方式，集成多个树的结果来提高模型的泛化能力。

作用：

* 防止过拟合，提高模型的鲁棒性。
* 在处理大规模数据时，随机森林能够有效地进行并行化处理。
* 对于高维数据，具有较好的特征选择能力。

## **5、三种决策树的信息划分准则。**

ID3（信息增益）： 使用信息熵来度量不确定性，选择能够最大程度减少不确定性的特征进行划分。

C4.5（信息增益比）： 在ID3的基础上，引入对特征取值数目的惩罚，避免选择取值较多的特征。

CART（基尼系数）： 使用基尼系数来度量数据的纯度，选择能够最小化基尼系数的特征进行划分。

## **6、集成学习中的bagging和boosting算法以及两种的区别。**

Bagging（Bootstrap Aggregating）：

* 并行训练多个弱学习器，每个学习器使用随机抽样得到的数据集。
* 最终结果通过投票或平均等方式融合。

Boosting：

* 串行训练多个弱学习器，每个学习器尝试修正前一个学习器的错误。
* 每个学习器的权重与其表现相关，错误越大权重越大。

区别：Bagging的每个模型权重相等，Boosting的每个模型权重不同，取决于其性能。

## **7、统计学中的P值是什么含义，如何通俗地解释？**

含义： P值是在原假设为真的情况下，观察到统计量或更极端情况的概率。

通俗解释： P值表示了观察到的数据在假设成立的情况下产生的可能性大小。较小的P值表明数据与假设的不一致性较大。

## **8、GBDT的原理。**

梯度提升决策树（Gradient Boosting Decision Trees）：通过迭代训练弱学习器（一般是决策树），每次训练调整之前学习器的残差，以逐步改进模型性能。通过梯度下降优化损失函数。可以使用回归损失函数或分类损失函数，如均方误差或对数损失函数。

## **9、编程题：有1元、4元、5元的硬币，找到满足15元的最少硬币数量。**

思路：动态规划

代码：

def min\_coins(amount):

coins = [1, 4, 5]

dp = [float('inf')] \* (amount + 1)

dp[0] = 0

for coin in coins:

for i in range(coin, amount + 1):

dp[i] = min(dp[i], dp[i - coin] + 1)

return dp[amount]

# 测试

amount = 15

result = min\_coins(amount)

print(f"找零 {amount} 元的最少硬币数量为：{result}")

dp[i]表示找零i元所需的最少硬币数量。通过遍历硬币面额，不断更新dp数组，最终得到找零15元的最少硬币数量。

# **商汤智能座舱一面（社招）面试题7道**

### **1、Transformer结构和LSTM的区别和优势，Transformer怎么体现时序信息。**

区别：

* Transformer是基于注意力机制（self-attention）的模型，通过同时处理整个序列的信息，而LSTM（长短时记忆网络）则是基于递归的循环神经网络。
* LSTM依赖于序列上的递归计算，而Transformer使用注意力机制进行全局性的关联。

优势：

* Transformer并行计算更有效，因为每个位置的计算独立于其他位置，而LSTM需要按顺序计算。
* Transformer更易于训练，不容易出现梯度消失或爆炸的问题。
* Transformer能够学习长距离依赖关系，而LSTM的这个能力在序列很长时受到限制。

时序信息：

* Transformer通过引入位置编码（Positional Encoding）来处理时序信息。这是通过为输入序列的每个位置添加一个特定的向量表示其在序列中的位置，从而使模型能够感知到输入的时序结构。

### **2、Transformer Encoder和Decoder的输入输出和结构。**

Encoder：

* 输入： 序列的嵌入表示或者经过了位置编码的输入。
* 输出： 对输入序列的编码表示。
* 结构： 多头自注意力层（Multi-Head Self-Attention）和前馈神经网络层（Feedforward Neural Network）的堆叠。

Decoder：

* 输入： 编码器的输出以及目标序列的嵌入表示。
* 输出： 对目标序列的解码表示。
* 结构： 多头自注意力层、多头编码-解码注意力层（Multi-Head Encoder-Decoder Attention）和前馈神经网络层的堆叠。

### **3、介绍下BERT模型。**

* BERT（Bidirectional Encoder Representations from Transformers）是一种预训练的自然语言处理模型，基于Transformer架构。
* BERT的核心思想是通过在大规模语料库上进行无监督训练，从而学习丰富的语言表示。
* BERT采用双向（bidirectional）的注意力机制，能够更好地理解上下文信息。
* 在预训练之后，BERT可以通过微调应用于各种下游任务，如文本分类、命名实体识别等。

### **4、LayerNorm & BatchNorm的作用以及区别。**

作用：

* BatchNorm（批归一化）： 在训练时通过归一化每个小批次的输入，加速训练并有正则化效果。
* LayerNorm（层归一化）： 在每个层的输入上进行归一化，通常应用于逐层的归一化处理。

区别：

* BatchNorm： 对每个批次的数据进行归一化，考虑了批次内的样本间关系。
* LayerNorm： 对每个样本的所有特征进行归一化，独立于批次。
* BatchNorm适用于全连接和卷积层，而LayerNorm更适用于循环神经网络（RNN）和Transformer等序列模型。

### **5、BatchNorm更多用在视觉上，LayerNorm更多用在语言上，为什么。**

原因：

* BatchNorm： 在图像处理中，数据通常以批次的形式输入，BatchNorm可以充分利用批次内的数据分布进行归一化，有助于训练深层的视觉模型。
* LayerNorm： 在自然语言处理中，序列数据（如语言）的长度可能会变化，而LayerNorm独立于批次，更适用于处理变长序列，保持对每个样本的归一化一致性。

### **6、编程题：跳跃游戏（LeetCode 55）。**

解题思路：

尽可能到达最远位置（贪心）。

如果能到达某个位置，那一定能到达它前面的所有位置。

方法：

初始化最远位置为 0，然后遍历数组，如果当前位置能到达，并且当前位置+跳数>最远位置，就更新最远位置。最后比较最远位置和数组长度。

时间复杂度 O(n)，空间复杂度 O(1)。

代码：

class Solution:

def canJump(self, nums) :

max\_i = 0 #初始化当前能到达最远的位置

for i, jump in enumerate(nums): #i为当前位置，jump是当前位置的跳数

if max\_i>=i and i+jump>max\_i: #如果当前位置能到达，并且当前位置+跳数>最远位置

max\_i = i+jump #更新最远能到达位置

return max\_i>=i

### **7、编程题：跳跃游戏2（leetcode45）。**

思路：

从前往后先找如果到达最后一步，那么前一步最早能落在那里，找到前一步后，再把该步的下标当作目标，直到目标下标为0。

代码：

class Solution:

def jump(self, nums: List[int]) -> int:

n = len(nums)

aim = n

t = 0

while aim > 1:

for i in range(aim):

if i + nums[i] >= aim-1:

aim = i +1

t += 1

break

return t

# **快手算法实习面试题5道**

### **1、xgboost怎么进行预排序的**

在XGBoost中，预排序是通过构建一个树之前对数据进行预排序来提高训练速度的一种技术。它通过对每个特征的值进行排序，然后记录每个样本在排序后的位置。这样，在构建树时，可以快速地找到最佳的分割点。具体步骤如下：

* 对每个特征的取值进行排序。
* 记录每个样本在排序后的位置。
* 在训练树的过程中，根据排序后的位置来选择最佳的分割点。

### **2、xgboost特征重要性是如何做的**

XGBoost支持三种特征重要性计算方式：weight、gain和cover。 weight方式只计算特征在模型中被使用的次数，gain方式只计算特征的增益值，cover方式只计算特征的覆盖率。 一般来说，gain方式是最常用的计算方式，因为它能够更全面地评估特征的重要性。

### **3、Transformer结构介绍：**

Transformer是一种用于处理序列数据的模型，特别适用于自然语言处理任务。它由编码器和解码器组成，其中核心结构包括自注意力机制和前馈神经网络。

编码器： 包含多个相同的层，每个层都有两个子层，分别是多头自注意力机制和前馈神经网络。这些层之间通过残差连接和层归一化进行连接。

解码器： 同样由多个相同的层组成，每个层包含三个子层，分别是多头自注意力机制、多头注意力机制（用于处理编码器输出），和前馈神经网络。

### **4、场景题：如何判定看快手视频时，是否挂机**

判定是否挂机可以考虑以下几个方面：

* 用户交互： 检测用户是否有鼠标移动、键盘输入等交互行为，挂机通常不会有这些行为。
* 视频播放模式： 观察视频播放进度，如果一直在同一时间点停留，可能是挂机。
* 设备传感器： 利用设备传感器，如摄像头或麦克风，检测是否有用户存在。
* 行为模式分析： 使用机器学习模型，监测用户在观看视频时的行为模式，挂机可能表现出不规律或重复性较强的行为。

### **5、手撕螺旋矩阵**

螺旋矩阵是指按照顺时针螺旋的方式填充矩阵。以下是一个简单的Python代码，用于生成螺旋矩阵：

def generate\_spiral\_matrix(n):

matrix = [[0] \* n for \_ in range(n)]

top, bottom, left, right = 0, n-1, 0, n-1

num = 1

while top <= bottom and left <= right:

for i in range(left, right + 1):

matrix[top][i] = num

num += 1

top += 1

for i in range(top, bottom + 1):

matrix[i][right] = num

num += 1

right -= 1

if top <= bottom:

for i in range(right, left - 1, -1):

matrix[bottom][i] = num

num += 1

bottom -= 1

if left <= right:

for i in range(bottom, top - 1, -1):

matrix[i][left] = num

num += 1

left += 1

return matrix

# 例如，生成3x3的螺旋矩阵

result\_matrix = generate\_spiral\_matrix(3)

print(result\_matrix)

# **字节数据科学—模型开发实习面试题9道**

## **1、介绍一下Bert模型**

BERT（Bidirectional Encoder Representations from Transformers）是由Google在2018年提出的自然语言处理（NLP）领域的预训练模型。它采用了Transformer架构，通过无监督学习从大规模文本数据中学习语言表示。BERT在训练阶段使用了两个任务：Masked Language Model（MLM）和Next Sentence Prediction（NSP），使其能够更全面地理解上下文中的语言。可以在特定任务上进行微调，以适应不同的应用场景。

## **2、树模型是如何计算每个特征的重要性的**

树模型计算特征重要性主要是根据不用的指标来计算的。

* 基尼重要性（Gini Importance）： 基尼重要性衡量了一个特征在决策树中的节点分裂中对纯度改善的贡献程度。通过计算每个特征在不同节点上的基尼指数减小量，然后加权求和，得到特征的重要性。
* 信息增益（Information Gain）： 信息增益用于衡量一个特征在决策树节点分裂中对不确定性的减小程度。通过计算每个特征对目标变量的信息增益，可以评估其重要性。

## **3、特征选择中，前向选择和反向选择实现起来有什么区别，哪个效果更好**

* 前向选择（Forward Selection）： 从空特征集合开始，逐步添加最重要的特征，直到满足某个停止准则（如达到指定特征数量）。这种方法的优点是简单直观，但可能忽略一些特征之间的交互影响。
* 反向选择（Backward Elimination）： 从包含所有特征的集合开始，逐步剔除最不重要的特征，直到满足停止准则。这种方法考虑了特征之间的交互，但计算复杂度较高。

哪个效果更好取决于具体情况，通常需要通过实验验证。前向选择可能更适用于大规模数据，而反向选择可能在特征间存在相关性时更为有效。

## **4、如何构建多模态模型**

多模态模型结合了来自不同数据源或模态的信息，通常包括视觉、语言、音频等。构建多模态模型的一般步骤包括：

数据预处理： 将不同模态的数据统一格式，确保能够输入到模型中。

模型设计： 使用适当的深度学习架构，如融合型（Fusion-based）或并行型（Parallel-based），将不同模态的信息整合在一起。

训练和微调： 利用多模态数据进行模型训练，并通过微调来适应具体任务。

## **5、在多模态任务中，如果视觉模型的输出张量比语言模型的输出张量短很多，该进行什么操作**

可以尝试使用以下方法：

填充（Padding）： 在视觉模型的输出中添加适当数量的填充，使其长度与语言模型的输出一致。

剪裁（Trimming）： 在语言模型的输出中剪裁一部分，使其长度与视觉模型的输出一致。

使用注意力机制： 在模型设计中，可以使用注意力机制来动态地对不同模态的信息进行权重分配，从而处理不同长度的输出。

## **6、介绍一下transformer**

Transformer的核心思想是自注意力机制（self-attention），它允许模型在处理序列数据时关注输入序列中的不同位置，而无需依赖于序列的顺序。Transformer包括编码器（Encoder）和解码器（Decoder），每个编码器和解码器块都包含多头注意力机制和前馈神经网络。

## **7、transformer的输入和输出分别是什么**

输入： Transformer的输入是一个序列，如文本中的一句话。每个词（或标记）都会被嵌入成一个向量，并加上位置编码，形成输入序列。

输出： Transformer的输出是对输入序列中每个位置的预测，或者是整个序列的表示。在机器翻译等任务中，输出通常是目标语言的对应序列。

## **8、transformer的输出和bert有什么区别**

Transformer输出： 在标准的Transformer模型中，输出是由解码器的最终层产生的，通常是一个表示整个序列的向量。这个向量可以用于各种任务，如文本分类、生成等。

BERT输出： BERT（Bidirectional Encoder Representations from Transformers）是基于Transformer的预训练模型，主要用于学习丰富的上下文语境。BERT的输出不仅包含了整个序列的向量，还包括了每个输入词的上下文相关表示。BERT的预训练阶段包括两个任务：Masked Language Model（MLM）和Next Sentence Prediction（NSP）。

总的来说，BERT的输出更注重于每个词的上下文相关表示，而标准的Transformer输出更倾向于整个序列的表示。

## **9、介绍一下注意力机制**

注意力机制是深度学习模型中一种关键的机制，用于处理序列数据时更灵活地关注不同位置的信息。在Transformer中，注意力机制通过自注意力机制实现。

* 自注意力机制（Self-Attention）： 给定一个输入序列，自注意力机制通过对序列中每个元素计算权重，然后将这些权重加权求和得到输出。这个权重表示了模型对输入中不同位置的关注程度。具体来说，对于每个位置，通过计算与其他位置的相似度得到一个权重，最后将这些权重应用于输入序列上。这使得模型能够同时关注序列中的多个位置。
* 多头注意力机制（Multi-Head Attention）： 为了提高模型的表达能力，Transformer引入了多头注意力机制，允许模型学习多组不同的注意力权重。每个注意力头都产生一个输出，最后通过线性变换和拼接得到最终的多头注意力输出。

注意力机制的引入提高了模型处理序列数据的能力，使得模型能够更好地捕捉上下文信息。

# **好未来NLP算法工程师面试题9道**

## **1、lora的矩阵怎么初始化？为什么要初始化为全0？**

矩阵B被初始化为0，而矩阵A正常高斯初始化

如果B，A全都初始化为0，那么缺点与深度网络全0初始化一样，很容易导致梯度消失(因为此时初始所有神经元的功能都是等价的)。

如果B，A全部高斯初始化，那么在网络训练刚开始就会有概率为得到一个过大的偏移值Δ W 从而引入太多噪声，导致难以收敛。

因此，一部分初始为0，一部分正常初始化是为了在训练开始时维持网络的原有输出(初始偏移为0)，但同时也保证在真正开始学习后能够更好的收敛。

## **2、gpt源码past\_key\_value是干啥的？**

在GPT（Generative Pre-trained Transformer）中，past\_key\_value是用于存储先前层的注意力权重的结构。在进行推理时，过去的注意力权重可以被重复使用，避免重复计算，提高效率。

## **3、gpt onebyone 每一层怎么输入输出**

在GPT One-by-One中，每一层的输入是上一层的输出。具体而言，输入是一个序列的嵌入表示（通常是词嵌入），并通过自注意力机制和前馈神经网络进行处理，得到输出序列的表示。

## **4、模型输出的分布比较稀疏，怎么处理？**

可以采用一些方法来处理模型输出的分布稀疏，例如使用softmax函数的温度参数调节来平滑输出分布，或者引入正则化技术，如Dropout，以减少模型对特定类别的过度依赖。

## **5、决策树是如何做回归的**

决策树回归通过在数据空间中递归划分，每个叶子节点都对应一个具体的回归值。在训练过程中，决策树根据输入特征的不同划分数据，使每个叶子节点尽可能纯净（样本值相似）。叶子节点的回归值可以是该节点内所有训练样本的平均值。

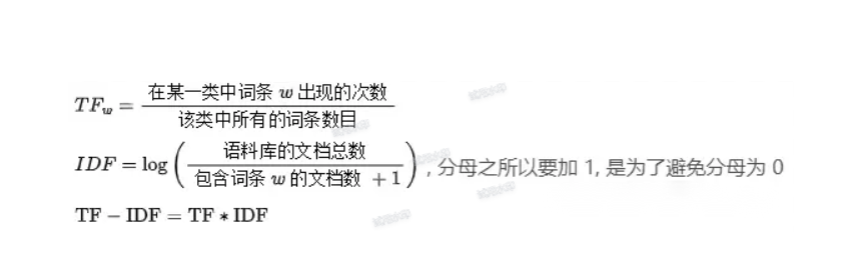
## **6、kl散度的公式和kl散度与交叉熵的区别**

KL（Kullback-Leibler）散度衡量了两个概率分布之间的差异。其公式为：

IMG_256

KL散度指的是相对熵，KL散度是两个概率分布P和Q差别的非对称性的度量。KL散度越小表示两个分布越接近。也就是说KL散度是不对称的，且KL散度的值是非负数。（也就是熵和交叉熵的差）

## **7、tfidf公式**



## **8、除了cosin还有哪些算相似度的方法**

除了余弦相似度（cosine similarity）之外，常见的相似度计算方法还包括欧氏距离、曼哈顿距离、Jaccard相似度、皮尔逊相关系数等。

## **9、cart树的分裂准则是啥**

CART（Classification and Regression Trees）树的分裂准则通常采用基尼不纯度（Gini impurity）或均方误差（Mean Squared Error，MSE）。在分类问题中，CART树寻找能最小化基尼不纯度的特征和阈值进行分裂；在回归问题中，它寻找能最小化均方误差的分裂方式。分裂过程持续递归，直到满足停止条件。

# **百度算法工程师面试题8道**

## **1、讲一下lora的原理**

Lora方法指的是在大型语言模型上对指定参数增加额外的低秩矩阵，也就是在原始PLM旁边增加一个旁路，做一个降维再升维的操作。并在模型训练过程中，固定PLM的参数，只训练降维矩阵A与升维矩阵B。而模型的输入输出维度不变，输出时将BA与PLM的参数叠加。用随机高斯分布初始化A，用0矩阵初始化B。

## **2、讲一下GBDT的原理**

GBDT（Gradient Boosting Decision Tree）是一种集成学习方法，它通过迭代训练决策树，每一次迭代都试图修正前一次迭代的错误。具体而言，GBDT通过拟合残差（目标值与当前模型的预测之差）来逐步改进模型。在每一轮迭代中，新的决策树被训练以捕捉残差中的模式。最终，所有树的预测被组合起来形成最终的预测模型。

## **3、翻译任务一般用什么架构**

翻译任务主要使用的是seq2seq架构，目前主流的翻译架构是基于Transformer模型的，它采用自注意力机制（self-attention）来捕捉输入序列中不同位置的关联信息，并通过编码器-解码器结构来实现翻译。

## **4、encoder-only, decoder-only, encoder-decoder的区别**

Encoder-only：只有编码器，用于将输入序列编码成固定维度的表示，常用于特征提取。

Decoder-only：只有解码器，用于从固定维度的表示生成输出序列，通常在生成式任务中使用。

Encoder-decoder：同时包含编码器和解码器，用于序列到序列的任务，如机器翻译。编码器将输入序列编码成上下文信息，解码器使用该信息生成输出序列。

## **5、讲一下transformer的结构**

Transformer模型由编码器和解码器组成，其核心是自注意力机制。每个编码器和解码器均由多个相同的层叠加而成，每一层包含自注意力子层和全连接前馈网络子层。输入序列通过多头自注意力机制进行处理，然后通过前馈网络。每个子层后都有残差连接和层标准化。编码器和解码器之间通过编码器-解码器注意力连接。

## **6、见一下Flash Attention的原理**

Flash Attention从attention计算的GPU memory的read和write方面入手来提高attention计算的效率。其主要思想是通过切块（tiling）技术，来减少GPU HBM和GPU SRAM之间的数据读写操作。

## **7、fp32和fp16的区别，混合精度的原理**

fp32：32位浮点数，占用4字节，提供高精度的数值表示。

fp16：16位浮点数，占用2字节，提供相对较低的数值表示精度，但在许多情况下仍足够。

混合精度指的是在模型训练中同时使用fp16和fp32，以在计算中提高效率。通常，模型参数使用fp16表示，而梯度和累积值使用fp32表示。这样可以减少内存占用和计算开销，加速训练过程。

## **8、Beam search的原理**

Beam search是一种在序列生成任务中选择最优输出序列的搜索算法。在每一步，它保留概率最高的k个候选，然后根据下一个词的预测继续扩展这些候选。这个过程一直进行到生成完整序列为止。Beam search允许在搜索空间中进行局部剪枝，以平衡搜索的广度和深度，提高生成的序列的质量。

# **淘天阿里妈妈算法工程师机器学习面试题9道**

## **1、描述下Transformer的结构**

Transformer是一种深度学习模型架构，特别适用于序列到序列的任务，如机器翻译。其主要结构包括以下几个部分：

* 编码器（Encoder）： 由多个相同的层组成，每个层包含两个子层，即多头自注意力机制（Multi-Head Self-Attention）和前馈神经网络（Feedforward Neural Network）。
* 解码器（Decoder）： 同样由多个相同的层组成，每个层包含三个子层，分别是多头自注意力机制、多头注意力机制（用于处理编码器输出的信息）和前馈神经网络。
* 嵌入层（Embedding）： 将输入序列的词嵌入到连续的向量空间中。
* 位置编码（Positional Encoding）： 为输入序列的每个位置添加一个表示其位置的向量，以保留序列的顺序信息。
* Layer Normalization和残差连接： 在每个子层的输出上应用Layer Normalization，并使用残差连接将输入添加到子层的输出。

## **2、为什么Transformer可以处理多种模态，它是怎么处理的**

Transformer之所以能够处理多种模态的数据，是因为其核心操作——自注意力机制（Attention Mechanism）是通用的，不依赖于数据的特定结构。自注意力机制能够在输入序列中动态地关注不同位置的信息，从而适应不同模态的输入。

对于多模态数据，例如图像和文本，可以通过将每个模态的数据分别表示为输入序列，然后将这些序列送入Transformer模型。这样，模型可以通过自注意力机制在不同模态之间建立关联，捕捉模态间的复杂关系。

## **3、BN的作用和好处**

* 作用： BN主要用于加速神经网络的训练过程。它在每个批次的数据上进行标准化，即将输入数据减去均值并除以标准差，然后通过缩放和平移进行线性变换。这有助于维持每个神经元激活的分布稳定，防止梯度消失或爆炸问题。
* 好处：
  + 加速训练： BN有助于减小训练过程中的内部协变量转移（Internal Covariate Shift），从而提高网络训练的稳定性和速度。
  + 降低对初始参数的敏感性： BN使网络对初始权重的选择不那么敏感，减少了需要进行精细调整的依赖。
  + 正则化效果： BN在一定程度上具有正则化效果，可以减轻过拟合问题。

## **4、Dropout的好处**

Dropout通过随机地将一部分神经元的输出置零，强制模型去学习更健壮和泛化的特征，从而降低过拟合风险。

## **5、梯度消失的原因**

梯度消失是指在深度神经网络中，梯度逐渐变得极小，导致底层网络参数几乎无法更新。主要原因包括：

* 链式法则： 在链式法则中，每个乘法项都会缩小梯度，多层网络中这些项的连续相乘导致梯度指数级地减小。
* 激活函数的选择： 一些传统的激活函数，如sigmoid和tanh，具有饱和区域，梯度在这些区域非常小，使得反向传播时梯度减小。
* 深度网络结构： 随着网络深度增加，梯度经过多次连续的权重矩阵相乘，导致梯度逐渐衰减。

## **6、Resnet为什么能减缓梯度消失的原因**

ResNet采用了残差连接（Residual Connection），通过将输入直接添加到网络层的输出，形成了一个“跳跃连接”。这种结构有助于缓解梯度消失问题的原因有：

* **梯度直达**： 在反向传播中，由于有了跳跃连接，梯度能够直接通过残差块传播到较早的层，减轻了梯度逐层衰减的问题。
* **易于学习恒等映射**： 残差连接使得网络可以更容易地学习恒等映射，即将输入直接映射到输出，这有助于减轻梯度消失问题。

## **7、在nums寻找最小的连续子数组使得它们的和大于所给的targets**

方法：前缀和+二分

def search(s, k, n, t):

t > s[n]:

return -1

l, r = k, n

res = n

while l < r:

mid = (l + r) >> 1

if s[mid] >= t:

res = mid

r = mid - 1

else:

l = mid + 1

return res

def func(targets, nums):

if targets < sum(nums):

return 0

n = len(nums)

s = [0] \* (n + 1)

for i in range(1, n+1):

s[i] = s[i - 1] + nums[i]

res = n

for i in range(1, n+1):

t = targets + s[i - 1]

d = search(s, i, n, t)

if d > i and d - i + 1 < res:

res = d - i + 1

return res

target = 7

nums = [2,3,1,2,4,3]

print(func(target, nums))

## **8、求一个数的N次幂**

def pow(k, n):

if k == 0 and n == 0:

return None

flag = 1

if n < 0:

flag = 0

res = 1

while n > 0:

if (n & 1) == 1:

res = res \* k

k \*= k

n >>= 2

if flag == 1:

return res

return 1.0 / res

## **9、全排列**

输入: 5,3

输出: [(1,2,3),(1,2,4),(1,2,5),(2,3,4),(2,3,5),(3,4,5),...]

import copy

def dfs(l, res, n, m, index, level, tmp):

if level > m:

return

if index > n:

return

if level == m:

t = copy.deepcopy(tmp)

res.append(tuple(t))

return

# print("index ", index, " n - m + level + 1 ", n - m + level + 1)

for i in range(index, n - m + level + 1):

# print("i ", i, " l[i] ", l[i])

tmp.append(l[i])

dfs(l, res, n, m, i+1, level+1, tmp)

tmp.pop()

def pailie(n, m):

l = range(1, n+1)

res = []

tmp = []

dfs(l, res, n, m, 0, 0, tmp)

return res

print(pailie(5, 3))

# **阿里智能互联机器学习算法工程师面试题7道**

## **1、介绍决策树和gbdt以及它们之间的关系**

决策树可以被看作是GBDT中的基本学习器之一。在GBDT中，每棵树都是通过迭代学习而来的，每一棵树都试图纠正前一棵树的预测误差。通常，GBDT使用回归树（Regression Tree）作为基学习器。GBDT通过将多棵决策树组合在一起来构建一个更强大的模型，这些树相互协作，纠正前一棵树的不足，从而提高整体性能。GBDT的目标是最小化损失函数。

## **2、常见的深度学习的优化方法有哪些**

(1) 随机梯度下降 SGD

(2) 动量 Momentum

(3) Adam 算法

(4) Adamw算法

不同的优化器实质上只是定义了不同的一阶动量和二阶动量公式。

详见: [https://zhuanlan.zhihu.com/p/604551577?utm\_id=0](https://link.zhihu.com/?target=https://gw-c.nowcoder.com/api/sparta/jump/link?link=https%3A%2F%2Fzhuanlan.zhihu.com%2Fp%2F604551577%3Futm_id%3D0" \t "https://zhuanlan.zhihu.com/p/_blank)

## **3、介绍C++的左值右值**

左值是是通常可以放在等号左边的表达式，右值是是通常只能放在等号右边的表达式。

左值是有标识符、可以取地址的表达式，最常见的情况有：变量、函数或数据成员的名字返回左值引用的表达式，如 ++x、x = 1、cout << ’ '字符串字面量如 "hello world"在函数调用时，左值可以绑定到左值引用的参数，如 T&。一个常量只能绑定到常左值引用，如 const T&。

右值是没有标识符、不可以取地址的表达式，一般也称之为“临时对象”。最常见的情况有：返回非引用类型的表达式，如 x++、x + 1、make\_shared(42)除字符串字面量之外的字面量，如 42、true

详见: [https://blog.csdn.net/qq\_42108501/article/details/114479140](https://link.zhihu.com/?target=https://gw-c.nowcoder.com/api/sparta/jump/link?link=https%3A%2F%2Flink.zhihu.com%2F%3Ftarget%3Dhttps%253A%2F%2Fblog.csdn.net%2Fqq_42108501%2Farticle%2Fdetails%2F114479140" \t "https://zhuanlan.zhihu.com/p/_blank)

## **4、指针与引用的区别**

(1)引用访问一个变量是直接访问，而指针是间接访问。

(2)引用是一个变量的别名，不额外占用内存空间，而指针是一个变量，有自己的内存空间。

(3)引用定义的时候必须赋值,并且赋值之后不可以改变，指针定义的时候可以不赋值，赋值后可以改变其所指的值。

(4)引用不可以为空，但是指针可以指向空值。

(5)引用是类型安全的，一般不会引起内存泄露的问题，指针可能会，一般尽可能用引用代替指针。

## **5、各种设计模型**

(1) 单例模式: 确保一个类只有一个实例，并提供一个全局访问点。

(2) 抽象工厂模式: 提供一个接口，用于创建一系列相关或相互依赖的对象，而不需要指定其具体类。

(3) 适配器模式: 将一个接口转换成另一个接口，以适应不同的客户端需求。

(4) 观察者模式: 定义对象间的一对多依赖关系，使得一个对象的状态发生变化时，其所有依赖对象都会收到通知并自动更新。

(5) 策略模式: 定义一系列算法，将每个算法封装起来，并使它们可以 访问者模式: 允许在不改变元素类的前提下定义作用于元素的新操作。

## **6、python生成器、装饰器**

生成器: 函数中包含yield关键字的语句将被视为生成器函数。当调用生成器函数时，它会立即返回一个生成器对象，但不执行函数体内的代码。yield用于生成下一个值，同时保存函数的状态，以便在下一次调用时从上一次暂停的位置继续执行。

装饰器是Python中的一种高级编程特性，用于修改函数或类的行为，而不需要对其本身进行修改。装饰器通常是函数，它可以接受一个函数或类作为参数，并返回一个新的函数或类，从而增加或修改它们的功能。装饰器通常用于代码重用、AOP(面向切面编程)和功能增强。

## **7、介绍模型量化**

模型量化（Model Quantization）是一种用于减小深度学习模型的存储空间和加速推理速度的技术。它通过减少模型权重和激活值的精度，通常将它们从浮点数转换为较低位的整数或定点数，来实现这一目标。

# **58同城算法工程师面试题11道**

## **1、Gbdt和xgboost的区别**

XGBoost是对GBDT的改进和扩展，它提供了更高的效率、更好的性能、正则化技术、内置特征选择等功能。

(1) 正则化:

GBDT使用基本的树模型，并在每一轮迭代中逐渐增加树的复杂性。它使用简单的正则化技术，如叶子节点的最小样本数限制，来防止过拟合。

XGBoost引入了正则化技术，包括L1和L2正则化，以减少过拟合风险。它还使用了二阶导数信息来提高训练的稳定性。

(2) 高效性:

XGBoost通过多线程和分布式计算提供了更高的训练效率。它实现了高度优化的数据存储和计算，以减少内存使用和加速训练过程。

GBDT通常以串行方式训练，训练时间可能较长，特别是在处理大规模数据时。

(3) 缺失值处理:

XGBoost能够自动处理缺失值，无需手动进行处理。

在GBDT中，需要在数据预处理阶段手动处理缺失值，通常通过填充或删除缺失值。

(4) 内置特征选择:

XGBoost具有内置的特征选择功能，它可以估计每个特征的重要性，并根据其重要性进行特征选择。GBDT通常需要手动进行特征选择或依赖其他特征选择方法。

(5) 求导优化:

GBDT只需要对目标函数求一阶导，xgboost要求二阶导。

## **2、Lstm的特点**

(1) 门控机制: LSTM引入了门控机制，包括遗忘门、输入门和输出门，这些门控制着信息的流动和保存。遗忘门决定哪些信息应该被遗忘，输入门控制哪些信息应该被添加到记忆单元，输出门控制什么信息应该传递到下一个时间步。这种机制有助于控制信息的流动，提高了模型的训练效率。

(2) 长期记忆: LSTM的主要特点是能够捕捉和维护长期依赖关系，它在处理序列数据中表现出色。传统的RNN存在梯度消失问题，导致难以学习长序列的依赖关系，而LSTM通过设计具有记忆单元的结构来解决这个问题，允许信息在长时间内保持不变。

(3) 平行化训练: LSTM具有良好的并行性，可以加速训练过程，特别是在GPU上进行训练。这有助于处理大规模数据和加速深度学习模型的训练。

## **3、Transformer的最重要的特点，对比CNN的效果**

最重要的特点是自注意力机制。

对比CNN，transformer更注重全局特征，特征之间能并行计算，CNN更注重局部特征，图像分类领域中，在图像数量充足的情况下，tranformer的效果通常比CNN好。

## **4、ReLU激活函数的优缺点，怎么改进**

优点: (1) 当特征值大于0时，可以避免梯度消失

(2) 计算简单

缺点: (1) 非零均值

(2) 当特征值大量小于0时，可能引起梯度消失

(3) 当特征值大于0时，非线性拟合能力可能下降

改进:

改用Leaky ReLU函数

## **5、Sort Linux命令的使用**

# file.txt

# int int string

1 2 str1

1 1 str2

2 2 str3

# 按第一列的倒序，第二列的正序排序输出

2 2 str3

1 1 str2

1 2 str1

答案:

sort -r 1 -k 2 file.txt

## **6、Coding 斐波那契数列**

# 1 1 2 3 5 ...

# input : n

# output: 第n位的值

# n = 3, => 3

def func(n):

if n == 0 or n == 1:

return 1

a, b = 1, 1

for i in range(2, n+1):

tmp = b

b = a + b

a = tmp

return b

## **7、简单介绍一下Map reduce**

它用于处理海量数据，其核心思想是将大规模数据集分为多个小的子集，然后并行处理这些子集，最后将结果进行合并。

## **8、C++的指针和引用的区别**

(1)引用访问一个变量是直接访问，而指针是间接访问。

(2)引用是一个变量的别名，不额外占用内存空间，而指针是一个变量，有自己的内存空间。

(3)引用定义的时候必须赋值,并且赋值之后不可以改变，指针定义的时候可以不赋值，赋值后可以改变其所指的值。

(4)引用不可以为空，但是指针可以指向空值。

(5)引用是类型安全的，一般不会引起内存泄露的问题，指针可能会，一般尽可能用引用代替指针。

## **9、样本不均衡的处理方法**

(1) 欠采样

(2) 过采样

(3) 平衡读取数据

(4) 设置权重，对样本较少的数据设置较高的训练权重

(5) 使用平衡损失函数，比如focal loss等

(6) 数据增强

## **10、介绍Focal loss**

Focal Loss 最初由物体检测领域的研究者提出，其主要目标是减轻模型在训练过程中对大多数背景类别的关注，从而更好地处理少数类别的样本。这种损失函数有助于提高模型对罕见类别的检测性能。

Focal Loss 的主要特点如下:

关注难分样本: Focal Loss 通过调整样本的权重，更加关注难以分类的样本。通常情况下，容易分类的样本(大多数属于背景类别)会降低其权重，而难分类的样本(属于少数类别)会增加其权重。

降低易分类样本的权重: 通过调整损失函数，Focal Loss 能够有效地降低容易分类的样本(背景类别、样本数量多的类别)的权重，这样模型将更加关注罕见类别，从而提高了模型在罕见类别的检测能力。

Focal Loss 的引入有助于提高目标检测模型对于罕见目标的检测性能，减轻了类别不平衡问题对模型训练的影响。

## **11、二叉搜索树和二叉堆的区别**

二叉搜索树的左子树的结点的值都比根结点小，右子树结点的值都比根结点大。一般情况下，插入删除搜索的时间复杂度是O(logn)，最坏情况下是O(n)

二叉堆分为最大堆、最小堆。如果是最大堆，只要保证根结点的值大于左右子树的结点的值。插入删除排序的时间复杂度是O(nlogn)

# **美团算法工程师面试题7道|**

## **1、常用的评估指标，AUC计算方式、TPR、FPR**

AUC (Area Under the ROC Curve)：AUC是用于评估二分类问题中模型性能的常见指标。ROC曲线是以真正例率（TPR，True Positive Rate）为纵轴，假正例率（FPR，False Positive Rate）为横轴绘制的曲线。AUC表示ROC曲线下的面积，取值范围在0到1之间，越接近1表示模型性能越好。

TPR (True Positive Rate)：也称为召回率，表示正类别样本中被正确分类为正类别的比例。计算方式：TPR= (TP+FN)/TP ，其中TP是真正例的数量，FN是假负例的数量。

FPR (False Positive Rate)：表示负类别样本中被错误分类为正类别的比例。计算方式：

FPR= (FP+TN)/FP，其中FP是假正例的数量，TN是真负例的数量。

## **2、uplift model评估指标**

Uplift Model是用于评估营销策略效果的模型，通常使用以下指标来评估：

* Uplift Score：Uplift Score是衡量不同营销策略对目标变量（如购买行为）的影响的分数。它可以通过比较不同策略的目标变量概率来计算。
* Conversion Uplift：Conversion Uplift是目标变量在使用策略后与未使用策略的相对差异。它表示了策略的有效性。

## **3、数据量很大怎么平衡AUC计算速度和准确性；**

当面对大规模数据集时，可以采取以下方法平衡AUC计算速度和准确性：

* 子采样（Subsampling）：对于大规模数据集，可以随机抽取子样本进行AUC计算，而不是使用整个数据集。这会显著降低计算成本，但可能会引入采样误差。
* 并行计算：利用多核处理或分布式计算架构，将AUC计算任务并行化，以提高计算速度。
* 近似计算：使用近似算法来估计AUC，如MinHash和HyperLogLog等。这些方法可以在一定程度上降低计算复杂度，但可能牺牲一些准确性。
* Online计算：如果数据流式传入，可以使用在线AUC计算方法，避免在内存中保存整个数据集。

## **4、xgb的优点**

XGBoost（Extreme Gradient Boosting）的优点：

* 高性能：XGBoost是一种高性能梯度提升框架，采用了多线程和稀疏感知技术，能够在大规模数据集上进行高效训练和预测。
* 可扩展性：XGBoost支持分布式计算，可以在分布式计算环境下训练大规模模型。
* 正则化：XGBoost引入了L1和L2正则化来防止过拟合，提高模型泛化能力。
* 自动处理缺失值：XGBoost能够自动处理数据中的缺失值，无需手动填充。
* 特征重要性分析：XGBoost提供了特征重要性分析工具，可以帮助选择最重要的特征进行建模。
* 支持多种损失函数：XGBoost支持多种损失函数，包括分类、回归和排序等任务。

## **5、Transformer结构**

Transformer是一种用于处理序列数据的深度学习模型，其核心结构包括：

编码器-解码器结构：Transformer通常由编码器和解码器组成，用于各种序列到序列任务。编码器用于将输入序列编码为表示，解码器用于生成输出序列。

* 自注意力机制（Self-Attention）：Transformer中的自注意力机制允许模型在编码和解码阶段关注序列中不同位置的信息，从而捕捉全局依赖关系。
* 多头注意力（Multi-Head Attention）：多头注意力允许模型关注不同子空间的信息，提高了模型的表示能力。
* 前馈神经网络（Feed-Forward Network）：用于每个编码器和解码器层的前馈神经网络，用于进行非线性变换。
* 残差连接（Residual Connection）：残差连接允许信息在层之间流通，有助于避免梯度消失问题。
* 位置编码（Positional Encoding）：为了将位置信息引入模型，Transformer引入了位置编码，使模型能够理解序列中不同位置的元素。

## **6、transformer为什么用layer norm**

Transformer中使用Layer Normalization来规范化每个子层的输入。Layer Normalization有以下优点：

与Batch Normalization不同，Layer Normalization不需要在小批量数据上计算统计信息，因此更适用于序列数据处理。在Transformer中，Layer Normalization有助于缩放输入向下层，以避免梯度爆炸或梯度消失问题，提高模型训练的效率和稳定性。

## **7、介绍xgb，介绍二阶展开**

XGBoost：XGBoost是一种梯度提升树（Gradient Boosting Tree）算法，它结合了梯度提升和决策树的强大性能，用于解决分类和回归问题。XGBoost具有高性能、可扩展性、正则化、特征重要性分析等优点。

二阶展开：在机器学习和优化中，二阶展开（Second-Order Expansion）是一种用于拟合和优化非线性函数的方法。它涉及到对目标函数进行泰勒级数展开，通常包括一阶导数和二阶导数。在XGBoost中，二阶展开用于构建损失函数的近似，以提高模型的训练效率。通过精确计算一阶导数和二阶导数，XGBoost能够更准确地确定每次迭代中的树节点分裂，从而提高了模型的性能。

# **深信服算法二面面试题6道**

### **1、LSTM和Transformer优缺点**

LSTM 优点：

* 能够处理序列数据，具有记忆性，适合处理自然语言处理任务。
* 相对较少的参数，适用于小数据集。
* 直观的顺序处理方式，容易理解和可解释。

缺点：

* 串行计算，不适合高度并行化，因此在GPU上运行效率较低。
* 对长序列的建模效果有限，容易出现梯度消失或梯度爆炸问题。
* 难以捕捉全局依赖关系，对于一些复杂的任务性能不如Transformer。

Transformer优点：

* 并行计算，适合在GPU/TPU等硬件上高效运行。
* 能够捕捉全局依赖关系，适用于自然语言处理和计算机视觉等多领域。
* 可扩展性强，可以通过增加层数来适应不同任务。

缺点：

* 参数较多，需要大量数据进行训练，不适用于小数据集。
* 相对难以解释和理解，特别是在处理自然语言时。

### **2、如何解决Transformer长度限制的问题**

* 截断和填充（Truncation and Padding）：一种常见的方法是截断较长的序列或填充较短的序列，使它们具有相同的长度。这种方法可以通过添加特殊的填充标记（通常是零向量）来实现。然后，可以使用掩码（mask）来告诉模型忽略填充的部分。
* 分层编码（Hierarchical Encoding）：这种方法将输入序列分层编码，每一层处理一部分序列，从而处理较长的序列。这样可以降低计算复杂度。
* 局部注意力（Local Attention）：考虑到全局自注意力可能计算复杂度高，一些变种的Transformer引入了局部自注意力，使模型只关注序列中的局部区域，而不是全局区域。这有助于处理长序列。
* 稀疏注意力（Sparse Attention）：一些研究工作提出了稀疏的注意力机制，例如在BERT中的随机掩码注意力，以减少计算复杂度，同时仍能够捕捉远距离依赖关系。

### **3、CNN和Transformer相比区别**

CNN（卷积神经网络）:

* 主要用于图像处理，可以有效捕捉空间特征。
* 局部连接和参数共享，适用于平移不变性的任务。
* 通过卷积核提取特征，层次结构深，适合处理具有层次结构的数据。

Transformer:

* 主要用于序列数据处理，如自然语言处理。
* 基于自注意力机制，能够捕捉全局依赖关系。
* 更适用于处理不同距离之间的依赖关系，如语言翻译、文本生成等任务。

### **4、Swin-Transformer的结构**

Swin-Transformer是一种用于图像处理的模型，结合了Transformer的思想和CNN的特点。其核心结构包括：

**Swin Block**：Swin-Transformer引入了Swin Block，其中包括两部分：

* Local Window Partition：将输入图像分割为局部窗口，以减小计算复杂度。
* Shifted Window Attention：在每个窗口内使用局部自注意力机制。

**Layered Architecture**：Swin-Transformer由多个Swin Block组成，具有多层结构，以逐级提取图像特征。

**Hierarchical Patch Embedding**：将图像划分为不同级别的分辨率，然后对每个级别进行嵌入，以捕捉不同尺度的特征。

Swin-Transformer的结构允许处理大尺寸图像，同时具有较低的计算复杂度。

### **5、检测中的IOU-GIOU等演变流程**

在目标检测任务中，IoU（Intersection over Union）是常用的评估标准，但它存在一些问题，如对目标旋转不敏感，无法很好地处理边界框的精确度等。因此，研究人员提出了一些改进的评估标准，如GIoU（Generalized IoU）和CIoU（Complete IoU）。

* GIoU（Generalized IoU）：GIoU考虑了目标的对齐性，更准确地衡量了目标之间的重叠程度。它可以解决IoU在边界框不完全对齐时的问题。
* CIoU（Complete IoU）：CIoU进一步考虑了目标的长宽比和中心点的偏移，更准确地衡量了目标的重叠情况。它对于处理不同尺寸的目标和检测结果的精确度更高。

这些改进的IoU衡量方法可以提高目标检测任务的性能和准确度，尤其在处理复杂场景和不完全对齐的目标时更有优势。

### **6、AutoAssign方法介绍，以及发展流程**

AutoAssign是一种目标检测任务中的自动分配方法，用于自动为每个预测的边界框分配真实目标。发展流程如下：

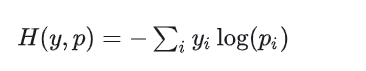
* **基本负责分配**：在AutoAssign的初期版本中，通常采用了一种基本的负责分配策略，例如IoU阈值分配，其中预测框与真实目标的IoU大于某个阈值的被分配为正例，其余的被分配为负例。这种方法存在一些问题，如难以处理多目标、不完全重叠等情况。
* **改进负责分配**：随后，研究人员提出了一些改进的负责分配方法，如Focal Loss，CornerNet等。这些方法更灵活，能够处理不同场景下的负责分配问题。例如，Focal Loss通过降低易分类的负例的权重来处理难负例问题。
* **自动权重分配**：AutoAssign的核心思想是自动权重分配。它通过学习一个权重向量，来平衡正例和负例的分配，从而更好地适应不同数据分布和难易样本。这种方法可以提高目标检测的性能，特别是在类别不平衡或难样本问题中。
* **联合训练和一致性损失**：AutoAssign还可以与其他目标检测技术一起使用，如Anchor-based或Anchor-free方法。它提供了一种一致性损失，使预测框与分配的真实目标之间保持一致，有助于提高目标检测的精确度。

总之，AutoAssign方法通过不断的发展和改进，为目标检测任务提供了更精确、灵活和高效的分配策略，有助于提高模型的性能和鲁棒性。

# **知乎算法面试题7道**

## **1、交叉熵损失函数写一下，物理意义是什么**

交叉熵损失函数（Cross-Entropy Loss Function）是用于度量两个概率分布之间的差异的一种损失函数。在分类问题中，它通常用于衡量模型的预测分布与实际标签分布之间的差异。



物理意义：交叉熵损失函数可以用来衡量实际标签分布与模型预测分布之间的“信息差”。当两个分布完全一致时，交叉熵损失为0，表示模型的预测与实际情况完全吻合。当两个分布之间存在差异时，损失函数的值会增加，表示预测错误程度的大小。

## **2、了解Bert模型吗**

Bert模型全名为Bidirectional Encoder Representations from Transformers。它是一种预训练的语言模型，通过阅读大量文本数据来学习词汇和语法结构，然后可以在各种NLP任务中进行微调。Bert模型引入了双向注意力机制，可以理解上下文信息，从而在各种自然语言处理任务中取得了显著的性能提升。

## **3、讲一下如何避免模型过拟合**

* 增加数据：增加训练数据量可以帮助模型更好地泛化，减少过拟合的风险。
* 使用正则化技巧：L1正则化和L2正则化可以降低模型复杂性，减少过拟合。
* 交叉验证：将数据分为训练集和验证集，以便及时检测模型的性能并进行调整。
* 早停法：根据验证集的性能，及时停止训练，以避免过度拟合。
* 减少模型复杂度：减少模型的层数、神经元数量或特征的维度可以降低过拟合风险。
* 集成学习：使用多个模型的预测结果进行集成，以提高泛化性能。

## **4、扩散模型了解吗**

扩散模型（Diffusion Model）是一种用于生成图像的生成模型。它基于一种称为“扩散过程”的数学方法，通过逐渐从噪声图像中生成目标图像。扩散模型在生成图像时可以控制图像的质量和内容，逐渐生成更清晰的图像。

## **5、最近关注的论文，多模态视觉大模型(CLIP,DALLE)**

多模态视觉大模型是指可以处理多种感知模态数据（如图像和文本）的大型深度学习模型。CLIP和DALL·E都是这方面的重要研究。CLIP（Contrastive Language-Image Pretraining）模型能够将图像和文本嵌入空间连接在一起，使得模型可以理解图像和文本之间的语义关系。DALL·E是一个生成模型，可以根据文本描述生成与之相关的图像。

## **6、手写kmeans**

import numpy as np

def kmeans(X, k, max\_iters=100):

# 随机初始化k个中心点

centroids = X[np.random.choice(range(len(X)), k, replace=False)]

for \_ in range(max\_iters):

# 分配每个数据点到最近的中心

distances = np.linalg.norm(X[:, np.newaxis, :] - centroids, axis=2)

labels = np.argmin(distances, axis=1)

# 更新中心点

new\_centroids = np.array([X[labels == i].mean(axis=0) for i in range(k)])

# 如果中心点没有改变，算法收敛

if np.array\_equal(centroids, new\_centroids):

break

centroids = new\_centroids

return centroids, labels

# 示例用法

data = np.array([[1, 2], [5, 6], [1, 3], [4, 7], [6, 8]])

k = 2

centroids, labels = kmeans(data, k)

print("最终中心点：", centroids)

print("聚类结果：", labels)

## **7、topK问题，两种方法**

方法一：排序法

* 对数据进行排序。
* 选择前K个元素即可。

方法二：堆（Heap）法

* 维护一个大小为K的最小堆（最小堆的根节点是最小值）。
* 遍历数据，将每个元素与堆中的最小值比较。
* 如果元素大于最小值，则将最小值出堆，将当前元素入堆。
* 遍历结束后，堆中的K个元素即为Top-K。

方法二的堆法在处理大规模数据时效率更高，因为它不需要对整个数据集进行排序。

# **字节电商算法实习面试题5道**

### **1、大模型微调 p\_tuning和传统fine tuning有什么区别 讲讲lora**

传统fine tuning会对模型所有的参数或大部分参数进行微调，对于大模型的庞大参数量来说，会消耗更多的时间和资源等，而P-tuning通过只优化几个token的参数，在有限算力下也可微调大型预训练语言模型。

LoRA的实现思想很简单，就是冻结一个预训练模型的矩阵参数，并选择用A和B矩阵来替代，在下游任务时只更新A和B。

### **2、讲讲transformer的架构**

Transformer本身是一个典型的encoder-decoder模型，Encoder端和Decoder端均有6个Block，Encoder端的Block包括两个模块，多头self-attention模块以及一个前馈神经网络模块；Decoder端的Block包括三个模块，Masked多头self-attention模块，多头Encoder-Decoder attention交互模块，以及一个前馈神经网络模块；需要注意：Encoder端和Decoder端中的每个模块都有残差层和Layer Normalization层。

### **3、transformer输出层的softmax有什么用**

Transformer输出层的softmax用于将模型的输出转换为概率分布，以便进行分类任务。它将模型的最后一层的隐藏状态映射到一个多类别分类任务上的概率分布，使得模型可以计算每个类别的概率，并选择概率最高的类别作为预测结果。

### **4、bert和gpt有什么区别**

BERT（Bidirectional Encoder Representations from Transformers）和GPT（Generative Pre-trained Transformer）是两种不同类型的预训练语言模型。主要区别在于：

BERT是一个双向编码器，它预测输入序列中的缺失部分，因此可以用于多种任务，如文本分类、命名实体识别等。

GPT是一个单向解码器，它生成文本的下一个单词，因此主要用于生成型任务，如文本生成、对话生成等。

### **5、一个数组 返回所有和为k的组合**

给定一个数组，返回所有和为k的组合可以通过回溯法或动态规划来实现。以下是一个示例的伪代码，演示如何使用回溯法来找到和为k的组合：

def find\_combinations(nums, k):

def backtrack(start, target, path):

if target == 0:

result.append(path)

return

for i in range(start, len(nums)):

if nums[i] <= target:

backtrack(i, target - nums[i], path + [nums[i]])

result = []

backtrack(0, k, [])

return result

# **招银网科算法面试题9道**

## **1、Python迭代器是什么，构成是怎样的？**

Python迭代器是一个对象，它允许你逐个访问一个序列中的元素，而不需要提前知道整个序列的大小。迭代器由两个主要方法组成：\_\_iter\_\_()和\_\_next\_\_()。\_\_iter\_\_()方法返回迭代器对象自身，而\_\_next\_\_()方法返回序列中的下一个元素。当没有元素可迭代时，\_\_next\_\_()方法会引发StopIteration异常。

## **2、介绍下yield**

yield是一个关键字，用于定义生成器函数。生成器函数在每次调用时不会执行完整的函数体，而是在遇到yield语句时暂停并将值返回给调用者。这个生成器函数的状态会被保留，以便在下次调用时继续执行。这种方法可以在迭代大型数据集或计算无限序列，占用较少的内存。

## **3、介绍下with**

with是Python的上下文管理器，用于处理资源的分配和释放，如文件、数据库连接等。通过使用with语句，你可以确保在块内退出时资源会被自动关闭或释放，而无需手动处理异常或资源管理。

## **4、深拷贝和浅拷贝**

深拷贝（deep copy）会复制对象及其内部包含的所有对象，创建一个完全独立的副本。

浅拷贝（shallow copy）只会复制对象本身，而不会递归复制内部包含的对象。内部对象的引用仍然指向原始对象的内部对象。

## **5、深度学习样本不均衡怎样处理：**

处理不均衡样本问题的方法包括重采样、数据增强、使用不同的损失函数、生成合成样本、调整类别权重等。具体的方法需要根据问题和数据集的特点而定。

## **6、过拟合怎样处理**

处理过拟合问题的方法包括增加更多的训练数据、减小模型复杂度、使用正则化技巧（如L1和L2正则化）、使用早停策略、数据增强等。

## **7、L1和L2正则化**

L1正则化：通过在损失函数中添加权重的绝对值之和，鼓励模型的权重趋向于稀疏，促使一些特征的权重变为零，从而实现特征选择。

L2正则化：通过在损失函数中添加权重的平方和，鼓励模型的权重趋向于较小的值，以防止过拟合。

## **8、BN层放在什么位置**

批量归一化（Batch Normalization，BN）通常在神经网络的每个隐藏层的激活函数之前应用。这有助于加速训练过程并提高模型的稳定性，使其更容易收敛到较好的结果。

## **9、10亿个参数的模型，部署后占用多大显存？**

目前模型的参数绝大多数都是float32类型, 占用4个字节。所以一个粗略的计算方法就是，每10亿个参数，占用约4G显存(实际应该是10^9\*4/1024/1024/1024=3.725G）。

# **百度秋招计算机视觉面试题12道**

### **1、Maxpooling的反向传播：**

Maxpooling的反向传播是为了计算池化层的梯度以便于更新前一层的权重。在反向传播过程中，只有最大值所在的位置会传递梯度，其他位置的梯度被设为零。具体步骤如下：

* 找到前向传播时最大值所在的位置。
* 在相同位置设置梯度为反向传播过来的梯度。
* 所有其他位置的梯度都设置为零。
* 将得到的梯度传递给上一层的相应位置。

### **2、L1和L2的区别：**

L1范数是指向量中各个元素的绝对值之和，也称为曼哈顿距离。L1正则化会倾向于产生稀疏权重，即某些权重变为零，从而进行特征选择。

L2范数是指向量中各个元素的平方和的平方根，也称为欧几里得距离。L2正则化会倾向于使权重分散在各个特征上，避免特定权重过大，有助于防止过拟合。

### **3、几种常见的图像特征：**

颜色特征：包括RGB颜色直方图、颜色矩等。

纹理特征：包括灰度共生矩阵、Gabor滤波器响应等。

形状特征：包括边缘检测、轮廓描述子等。

角点特征：包括Harris角点、FAST角点等。

SIFT特征和SURF特征：基于局部特征的描述子。

HOG特征：用于目标检测和行人识别。

### **4、深度可分离卷积是什么？**

深度可分离卷积是一种卷积神经网络中的卷积操作，它将标准卷积操作分解为两个独立的步骤：深度卷积和逐点卷积。

深度卷积：在这一步骤中，每个输入通道上的卷积操作是独立进行的，产生了一组中间特征图。

逐点卷积：在这一步骤中，对中间特征图的每个通道应用1x1卷积，以组合它们并生成最终的输出特征图。

这种分解可以大大减少参数数量，减小计算量，同时保持网络性能，特别适用于轻量级和移动设备上的深度学习任务。

### **5、CNN中参数量和计算量怎么算？**

参数量：计算每个卷积核中的参数数量，然后将其与卷积核的数量相乘，并考虑到所有层的参数。

计算量：计算每个卷积层的计算量，通常是输入特征图的大小乘以卷积核的大小乘以通道数量，然后将其与层的数量相乘，并考虑到所有层的计算量。

### **6、深度可分离卷积的参数量和计算量是多少？**

深度可分离卷积通常具有更少的参数量和计算量，具体的量取决于网络的架构和超参数的选择。一般来说，深度可分离卷积的参数量会比标准卷积少很多，而计算量也会大大减少。

### **7、了解Linux的管道命令吗？**

是的，Linux的管道命令允许将一个命令的输出作为另一个命令的输入，从而实现多个命令的串联操作。例如，可以使用管道将一个文件的内容传递给另一个命令进行处理，如将grep命令用于文本搜索，然后将结果传递给sed命令进行替换。

### **8、AUC和ROC是什么？**

ROC（Receiver Operating Characteristic）曲线是用于评估二元分类器性能的图形工具。它以不同的阈值设置下绘制了真正例率（True Positive Rate，TPR）和假正例率（False Positive Rate，FPR）之间的关系曲线。

AUC（Area Under the Curve）是ROC曲线下的面积，用于度量分类器在不同阈值设置下的性能。AUC的取值范围在0到1之间，AUC值越接近1表示分类器性能越好，越接近0.5表示性能越差，0.5表示随机分类器。

### **9、Precision和Recall是什么？**

Precision（精确率）是用于评估分类器性能的指标，它表示被分类为正例的样本中有多少是真正例。计算公式为 Precision = TP / (TP + FP)，其中TP表示真正例，FP表示假正例。

Recall（召回率）是用于评估分类器性能的指标，它表示所有真正例中有多少被正确分类为正例。计算公式为 Recall = TP / (TP + FN)，其中TP表示真正例，FN表示假负例。

### **10、Transfomer是什么？**

Transformer是一种用于自然语言处理（NLP）和其他序列到序列任务的深度学习架构。Transformer的主要特点是它不使用循环神经网络（RNN）或卷积神经网络（CNN），而是依赖自注意力机制（self-attention）来处理输入序列，从而实现了并行计算，提高了训练速度。

Transformer的核心组件包括：

* 自注意力机制：允许模型在处理输入序列时动态地关注不同位置的信息，从而更好地捕捉序列中的依赖关系。
* 多头注意力：通过多个不同的自注意力头并行处理输入，增强了模型对不同关系的建模能力。
* 位置编码：用于为模型提供输入序列中各个位置的信息，因为Transformer没有固定的顺序信息。
* 堆叠的编码器和解码器层：多层的编码器用于处理输入序列，多层的解码器用于生成输出序列（在机器翻译等任务中）。
* 残差连接和层归一化：用于帮助训练深层网络的技巧。

Transformer已经成为了NLP领域的标配架构，包括BERT、GPT（Generative Pre-trained Transformer）、T5（Text-to-Text Transfer Transformer）等模型都基于Transformer架构，并在各种NLP任务中取得了巨大成功。

### **11、手撕代码：找出数组中第K大的值。**

def findKthLargest(nums, k):

# 使用快速选择算法，类似于快速排序的思想

left, right = 0, len(nums) - 1

while left <= right:

pivot\_index = partition(nums, left, right)

if pivot\_index == k - 1:

return nums[pivot\_index]

elif pivot\_index < k - 1:

left = pivot\_index + 1

else:

right = pivot\_index - 1

def partition(nums, left, right):

pivot = nums[left]

low, high = left, right

while low < high:

while low <= right and nums[low] >= pivot:

low += 1

while high >= left and nums[high] < pivot:

high -= 1

if low < high:

nums[low], nums[high] = nums[high], nums[low]

nums[left], nums[high] = nums[high], nums[left]

return high

### **12、Python的深拷贝和浅拷贝的区别？赋值时浅拷贝还是深拷贝？**

* 浅拷贝：浅拷贝创建一个新对象，但是它只复制了原对象的引用，而不复制内部的对象。在Python中，可以使用copy模块的copy()函数或者列表、字典等容器对象的切片操作来创建浅拷贝。赋值操作也属于浅拷贝，即将一个对象的引用赋值给另一个变量。
* -深拷贝：深拷贝创建一个新对象，并且递归地复制原对象及其所有内部对象。在Python中，可以使用copy模块的deepcopy()函数来创建深拷贝。
* 赋值操作和浅拷贝都只复制了引用，多个变量会共享同一个对象。
* 深拷贝会创建一个完全独立的新对象，递归地复制所有内部对象，因此修改原对象不会影响深拷贝的对象。

# **SHEIN提前批NLP面试题5道**

## **1、Transformer的多头注意力的作用**

从GCN的角度就是探索更多的权重矩阵。原文的解释是MultiHead-Attention 提供了多个“表示子空间”，可以使模型在不同位置上关注来自不同“表示子空间”的信息。 即通过MultiHead，模型可以捕捉到更加丰富的特征信息。这里的多个表示子空间其实就是GCN里的多个权重矩阵。

## **2、无序数组的中位数**

方法一：快排一次后，检查k落在哪个区域，然后对那个区域再进行一次快排。如此反复，可得答案。

方法二：同样使用快排，但是对基准数不再随机，而是尽可能找出让两段区域长度相等的划分。（把原来的数组分成五份，然后找中位数，然后再在这些中位数里找出中位数作为基准）

## **3、一个元素在一个有序数组的第一次出现位置？**

要找到一个元素在一个有序数组中第一次出现的位置，你可以使用二分查找算法的变种。

def first\_occurrence(arr, target):

left, right = 0, len(arr) - 1

result = -1 # 初始化结果为-1，表示未找到元素

while left <= right:

mid = left + (right - left) // 2 # 计算中间索引

if arr[mid] == target:

result = mid # 找到目标元素，更新结果

right = mid - 1 # 继续在左侧查找第一次出现的位置

elif arr[mid] < target:

left = mid + 1

else:

right = mid - 1

return result

# 示例用法

arr = [1, 2, 2, 2, 3, 4, 4, 5]

target = 2

first\_index = first\_occurrence(arr, target)

if first\_index != -1:

print(f"元素 {target} 第一次出现的位置是 {first\_index}")

else:

print(f"元素 {target} 在数组中未找到")

这段代码首先初始化左右指针，然后在循环中执行二分查找，查找目标元素的第一次出现位置。如果找到了目标元素，则更新结果为中间索引，然后继续在左侧查找以找到第一次出现的位置。如果未找到目标元素，返回结果为-1。

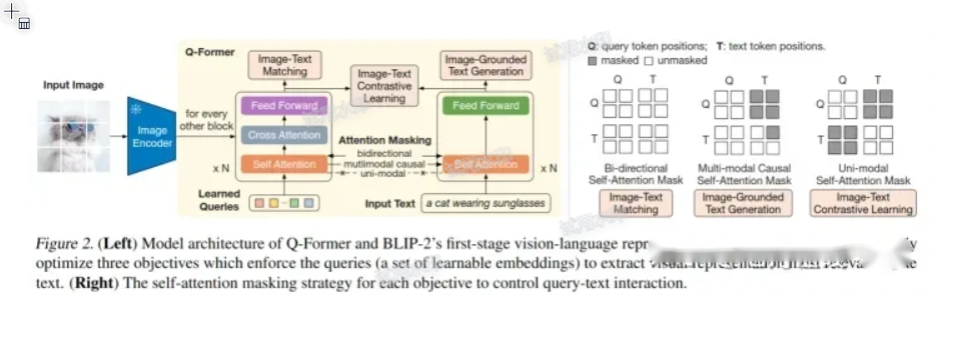
## **4、blip2的架构，优势和之前多模态模型的区别？**

blip2是图像-语言多模态模型的预训练方法。这个架构是2023年才提出的，也看出来面试紧跟时事了。

blip2的一个常见模式是输入一张图片，输出这张图片的描述。

bilp2是在冻结的图像模型（负责从图像中提取特征，比如vit）和冻结的语言模型（负责生成语言）中间放入一个Q-Former，我们的目标就是训练这个Q-Former。Q-Former包含图像Transformer和语言Transformer，图像Transformer包含CA和SA，SA和语言Transformer共享参数，CA只接受图像模型提取的图像特征，图像模型的输入是一个查询值，这个查询值将在SA中和自己交互，在CA中和图像特征交互。最后图像Transformer输出一个综合图像特征的向量，同时语言Transformer输入一个文本，进行encode，得到一个文本的向量。然后根据具体的任务选择不同的方式对这两个向量进行操作。最后，Q-former把得到的向量传给冻结的语言模型。语言Transformer训练的时候做解码器，预测的时候是解码器。

训练的时候先训练Q-Former和图像模型的交互，然后把Q-Former的结果和语言模型连接（中间可以加入全连接，前缀词等操作）。如下图



## **5、知识蒸馏和无监督样本训练？**

知识蒸馏是利用大模型把一个大模型的知识压缩到一个小模型上。具体来说你在一个训练集上得到了一个非常好的较大的模型，

然后你把这个模型冻结，作为Teacher模型也叫监督模型，然后你再造一个较小参数的模型叫做Student模型，我们的目标就是利用冻结的Teacher模型去训练Student模型。

A.离线蒸馏：Student在训练集上的loss和与Teacher模型的loss作为总的loss，一起优化。

B.半监督蒸馏：向Teacher模型输入一些input得到标签，然后把input和标签传给Student模型

还有个自监督蒸馏，直接不要Teacher模型，在最后几轮epoch，把前面训练好的模型作为Teacher进行监督。

目前知识蒸馏的一个常见应用就是对齐ChatGPT。

然后这个无监督样本训练，我看不懂意思。如果是传统的无监督学习，那就是聚类，主成分分析等操作。如果是指知识蒸馏的话，就是离线蒸馏的方式，只不过损失只有和Teacher的loss。

# **美团机器学习/数据挖掘面试题12道**

## **1、给出中序遍历和后序遍历，构建树**

中序遍历（Inorder）：左根右

后序遍历（Postorder）：左右根

构建树的过程可以通过递归算法来完成，首先找到后序遍历的最后一个节点作为根节点，然后在中序遍历中找到根节点的位置，将中序遍历分成左子树和右子树，然后递归构建左子树和右子树。

## **2、介绍树模型：GBDT和XGBoost**

GBDT（Gradient Boosting Decision Trees）和XGBoost（Extreme Gradient Boosting）都是集成学习方法，基于决策树的模型。它们通过迭代训练多个决策树，每棵树都尝试修正前一棵树的错误。

## **3、介绍LR（Logistic Regression）**

LR是一种用于二分类和多分类问题的线性模型。它使用逻辑函数将输入特征与输出概率之间建立关联，通常用于概率建模和分类任务。

## **4、XGB和LR的区别，各适用哪些场景**

区别：XGBoost是一种基于决策树的集成学习方法，具有强大的非线性建模能力，适用于复杂数据集。LR是线性模型，适用于简单数据集和线性可分问题。

适用场景：XGBoost通常在Kaggle等数据竞赛中表现出色，适用于高维、非线性和大规模数据集。LR适用于简单分类任务，易于解释和部署。

## **5、Lovain算法是什么算法**

Lovain算法是一种用于图的社区检测的聚类算法，通过不断合并具有相似邻居的节点来划分图中的社区。

## **6、准确率有什么缺点和问题**

准确率（Accuracy）在不平衡数据集中容易误导，因为它不考虑类别之间的不平衡。在类别分布不均匀时，准确率可能不是一个合适的评估指标，需要考虑其他指标如精确度、召回率、F1分数等。

## **7、AUC是什么**

AUC（Area Under the Curve）是用于衡量二分类模型性能的指标。它表示ROC曲线下的面积，范围在0到1之间，AUC值越高，模型性能越好。

## **8、常见的优化算法**

常见的优化算法包括梯度下降法（包括随机梯度下降和批量梯度下降）、Adam、RMSprop、Adagrad等。这些算法用于调整模型参数以最小化损失函数。

## **9、常见的激活函数**

常见的激活函数包括Sigmoid、ReLU（Rectified Linear Unit）、Tanh、Leaky ReLU、ELU（Exponential Linear Unit）等，用于神经网络中的非线性变换。

## **10、常见的特征提取方法**

常见的特征提取方法包括主成分分析（PCA）、线性判别分析（LDA）、单词嵌入（Word Embeddings）、卷积神经网络（CNN）特征提取等，用于从原始数据中提取有用的特征。

## **11、CNN和MLP区别，CNN的优势**

CNN（卷积神经网络）和MLP（多层感知机）都是神经网络模型，但CNN在处理图像和空间数据时具有优势。CNN使用卷积层和池化层可以捕捉局部特征和空间结构，减少了参数数量，并且在图像处理等领域表现出色。

## **12、RNN和LSTM，优缺点**

RNN（循环神经网络）是一种适用于序列数据的神经网络，但它存在梯度消失和梯度爆炸的问题。LSTM（长短时记忆网络）是RNN的一种变体，通过门控机制解决了梯度问题，可以更好地捕捉长期依赖性。然而，LSTM相对复杂，训练和计算成本较高。

# **得物NLP算法方向面试题8道**

### **1、在自然语言处理模型训练中，评价指标是怎样设定的？**

在自然语言处理模型训练中，评价指标通常根据任务的性质而定。常见的评价指标包括精确度、召回率、F1分数、BLEU分数、ROUGE分数、困惑度等。选择评价指标时需要考虑任务的具体要求，例如分类任务可以使用准确度，生成任务可以使用困惑度等。

### **2、自然语言处理中对低质量数据做数据清洗的方法？**

自然语言处理中对低质量数据的数据清洗方法包括去除重复数据、去除异常值、拼写校正、停用词去除、标记化和词干化等。此外，可以使用机器学习模型来自动检测和过滤低质量数据。

### **3、LSTM和RNN有什么区别？解决什么问题？**

LSTM（长短时记忆网络）和RNN（循环神经网络）都是序列模型，但LSTM在解决长序列依赖性问题上更有效，因为它通过门控单元可以更好地捕捉和管理长期依赖。RNN存在梯度消失和梯度爆炸的问题，而LSTM通过门控机制部分解决了这些问题。

### **4、transformer中，attention矩阵的计算为什么要除以根号dk**

在Transformer中，将注意力矩阵的计算结果除以根号dk的目的是为了控制注意力分布的尺度，使其更稳定。这有助于防止梯度爆炸和更好地训练模型，因为较小的注意力权重可以更均匀地分布在不同位置。

### **5、多任务学习各loss差异过大怎样处理？**

多任务学习中，如果各任务的损失差异过大，可以通过动态调整损失权重、使用任务特定的损失函数、改变模型架构或引入正则化等方法来处理。目标是平衡各任务的贡献，以便更好地训练模型。

### **6、多模态融合后，怎样知道最终结果受哪种模态影响更大？**

在多模态融合后，了解最终结果受哪种模态影响更大可以使用特征重要性分析方法，如SHAP值、Permutation Importance等。这些方法可以帮助识别每个模态对最终结果的贡献程度。

### **7、过拟合应该怎样处理？**

处理过拟合的方法包括增加训练数据、减小模型复杂度、使用正则化技术（如L1和L2正则化）、提前停止训练、使用更简单的模型架构、数据增强等。

### **8、BN层在训练和推理过程中有什么不一样？**

BN（批量归一化）层在训练和推理过程中有不同的计算方式。在训练过程中，BN层根据每个批次的统计信息来归一化输入数据。而在推理过程中，BN层通常使用训练时计算的均值和方差来进行归一化，而不是每个批次重新计算。这有助于保持模型在推理时的稳定性。

# **字节CV搜索提前批面试题7道**

### **1、Stable Diffusion里是如何用文本来控制生成的，Unet的Attention模块具体是如何实现的**

Stable Diffusion是一种潜在扩散模型，主要通过自动编码器（VAE），U-Net以及文本编码器三个核心组件完成用文本来控制生成的图像。Unet的Attention模块Latent Feature和Context Embedding作为输入，将两者进行Cross Attenetion操作，将图像信息和文本信息进行了融合，整体上是一个经典的Transformer流程。

### **2、Cross Attention与softmax**

Cross Attention：在Transformer模型中，Cross Attention通常用于“编码器-解码器”结构中的解码器，允许解码器关注编码器的输出。解码器可以使用这种注意力权重来决定从编码器输出中关注哪些部分。

Softmax：Softmax函数是一种将向量转化为概率分布的函数。给定一个实数向量，它会输出一个相同维度的新向量，其中每个值都是在0和1之间，并且所有值的总和为1。具体地说，softmax函数的第i个元素的值是原向量的第i个元素的指数与所有元素的指数之和的比值。

### **3、交叉熵**

交叉熵用于衡量两个概率分布之间的差异。在机器学习中，通常一个分布是真实的标签分布，另一个分布是模型的预测。数学上，交叉熵定义为

### **4、Focal Loss**

Focal Loss是为了解决目标检测任务中正负样本不均衡问题而提出的。在传统的交叉熵损失中，容易出现大量的负样本主导损失，导致模型无法学习到稀有或困难的正样本。Focal Loss通过给予那些容易分类的样本更小的权重来减少这种影响。

### **5、Dropout**

Dropout是一种正则化技巧，其在训练期间随机“关闭”神经网络中的一部分神经元。这有助于防止模型过拟合。

在训练时，Dropout层会随机地将其输入单位的一部分设置为0，其余部分不变。而在推理（或测试）时，Dropout层不会修改其输入，但可能会根据丢弃率对其输出进行缩放。

### **6、排序算法的复杂度**

快速排序：平均时间复杂度是O(n log n)，但在最坏的情况下是O(n^2)。

冒泡排序：时间复杂度在最好、平均和最坏的情况下都是O(n^2)。

插入排序：最好的情况下时间复杂度是O(n)，平均和最坏的情况下是O(n^2)。

### **7、堆排序**

堆排序是一种基于二叉堆数据结构的排序算法。

建堆的过程：可以从一个无序的数组开始，并将其转化为一个最大堆（父节点的值大于子节点）或最小堆（父节点的值小于子节点）。常用的方法是“下沉”方法，从最后一个非叶子节点开始，确保所有节点都满足堆的性质。

复杂度：建堆的时间复杂度是O(n)，而堆排序的总时间复杂度是O(n log n)。

# **快手机器学习算法岗面试题6道**

## **1、Kmeans与Kmeans++的区别：**

Kmeans：是一种迭代的聚类方法，用于将n个点分为k个聚类。其初始的质心通常是随机选择的，这可能导致算法陷入局部最优解。

Kmeans++：对Kmeans的初始化进行了优化，以期望得到一个比传统Kmeans更好的、全局的初始解。在Kmeans++中，第一个质心是从数据点中随机选择的，之后的质心是在数据点中按一个特定的概率分布选择的，其中距离当前已选择质心集更远的点有更高的概率被选择。

## **2、SVM和Logistic Regression的区别：**

SVM (支持向量机)：是一种分类和回归的算法。对于分类问题，SVM试图找到一个超平面来最大化两个类别之间的边界。关键点是支持向量，即距离超平面最近的那些点。

Logistic Regression：是一个用于估计类别概率的统计方法。输出是两个类之间的概率，通常使用sigmoid函数。当这个概率超过某个阈值(如0.5)时，将其归为某一类。

## **3、常见的距离函数：**

欧氏距离：二维或多维空间中两点之间的“直线”距离。

曼哈顿距离：在一个格子化的平面上，两点之间的距离是沿格子边界行走的距离。

切比雪夫距离：两点之间的距离是它们坐标差的最大值。

余弦相似度：衡量两个向量的方向的相似度，但不考虑它们的大小。

还有其他许多距离函数，具体的选择取决于应用场景。

## **4、介绍XGBoost：**

XGBoost是“Extreme Gradient Boosting”的缩写，是一个优化的分布式梯度增强库，用于实现机器学习中的提升树算法。XGBoost提供了并行树提升和特定硬件优化功能，使其训练速度快于许多其他提升树实现。

XGBoost的特点包括处理缺失值、剪枝、交叉验证、正则化等。

## **5、Bagging与Boosting的区别：**

Bagging (Bootstrap Aggregating)：主要的思想是从原始数据集中随机地抽取子集，并在每个子集上训练模型。所有模型的预测结果会被平均（回归问题）或投票（分类问题）来得到最终的预测结果。代表算法：随机森林。

Boosting：是一个迭代的过程，其中每一步都增加了一个新的模型，这个模型试图纠正前一个模型的错误。Boosting通常会赋予被错误分类的样本更高的权重，以便下一个模型能更加关注这些样本。代表算法：AdaBoost, Gradient Boosting, XGBoost等。

## **6、L1和L2正则化：**

L1正则化：也称为Lasso正则化，将权重向量的绝对值之和添加到损失函数中。这可以导致某些特征的权重为0，从而实现了特征选择。

L2正则化：也称为Ridge正则化，将权重向量的平方值之和添加到损失函数中。这防止权重变得过大，但通常不会使权重变为0。

# **字节跳动风控算法方向面试题7道**

### **1、过拟合的解决方法：**

过拟合是机器学习中常见的问题，它指的是模型在训练数据上表现很好，但在新数据上表现较差的现象。以下是几种解决过拟合问题的方法：

* 数据集扩增：通过增加更多的训练数据，可以减少过拟合。数据扩增可以通过各种方法，如旋转、裁剪、翻转、缩放等来实现。
* 正则化（L1和L2正则化）：L1和L2正则化是在损失函数中添加对模型参数的惩罚，以防止其过大。L1正则化通过添加参数绝对值的惩罚，可以使得某些参数为零，从而实现特征选择的效果；而L2正则化通过添加参数平方的惩罚，可以让参数较小，但不为零。
* 交叉验证：使用交叉验证来评估模型的性能，可以更好地了解模型在新数据上的表现，从而避免过拟合。
* 特征选择：通过选择最重要的特征，去除冗余特征，可以减少模型的复杂度，从而降低过拟合的风险。
* Dropout：在神经网络中应用Dropout层，以一定的概率随机丢弃一部分神经元的输出，可以减少神经网络中的过拟合。

### **2、Dropout训练和测试的区别：**

Dropout是一种用于解决神经网络过拟合问题的方法。在训练阶段和测试阶段，Dropout的行为是不同的。

* 在训练阶段，Dropout起作用，随机地使一些神经元输出变为零。这样做的目的是为了让网络不依赖于特定的神经元，从而增加网络的泛化能力。在训练过程中，通过多次迭代，每次迭代中随机选择不同的神经元丢弃，使得网络能够学习到多个不同的子网络。
* 在测试阶段，Dropout不起作用，所有神经元的输出都保留。这是因为在测试阶段，我们需要获得确定性的预测结果，而不是通过多个子网络的平均结果来进行预测。

### **3、优化器有哪些，详细讲一讲**

优化器是机器学习和深度学习中用于更新模型参数以最小化损失函数的算法。以下是一些常见的优化器：

* 随机梯度下降（SGD）：每次使用一个样本来更新参数，容易陷入局部最小值，训练过程可能不稳定。
* 动量优化器（Momentum）：引入动量项来在更新中考虑之前的梯度方向，可以加速收敛，并减少梯度更新的方差。
* AdaGrad（自适应梯度算法）：自适应地调整学习率，对于稀疏特征的数据效果较好，但可能造成学习率过小问题。
* RMSprop（均方根传播）：对AdaGrad进行改进，通过引入移动平均来解决学习率过小的问题。
* Adam（自适应矩估计）：结合了动量优化器和RMSprop，综合了两者的优点，被广泛应用于深度学习中。

### **4、特征筛选：**

特征筛选是指从原始特征集合中选择最相关、最有用的特征，以提高模型性能并减少计算成本。常见的特征筛选方法包括：

* 方差选择：选择方差较大的特征，适用于二元特征和数值特征。
* 相关性分析：通过计算特征与目标变量之间的相关性，选择与目标变量相关性较高的特征。
* 特征重要性：对于树模型等算法，可以利用特征重要性指标来筛选重要特征。
* 嵌入法：在模型训练过程中，根据特征的权重或系数来选择特征。
* Wrapper方法：通过训练模型，并在每次训练迭代中选择特征子集来评估性能，选择效果最好的特征子集。

### **5、机器学习模型了解：**

机器学习模型是一种用于学习输入数据与输出标签之间的关系的算法。以下是两种常见的机器学习模型：

* 树模型：决策树和随机森林是常见的树模型。决策树是一种基于树结构的分类和回归模型，它将数据划分为不同的节点，并根据特征值做出预测。随机森林是多个决策树的集成，通过投票或平均预测结果来提高准确性和稳定。
* 线性模型：线性模型是一种简单而有效的模型，其基本形式是输入特征与一组权重的线性组合。对于分类问题，常用的线性模型有逻辑回归，它使用sigmoid函数将线性预测值映射到概率输出；对于回归问题，线性回归是最常见的线性模型，通过最小化预测值与实际值之间的均方差来拟合数据。

### **6、题：如何识别抖音刷赞行为**

识别抖音刷赞行为是一种反作弊的任务，旨在检测用户是否使用机器或脚本等方式刷赞，以维护抖音平台的公平性和正常使用。这个问题可以通过以下方法来解决：

* 用户行为特征：分析用户在平台上的行为特征，比如点赞频率、点赞时间分布、点赞的对象、点赞行为与其他行为的关联等。刷赞行为通常会表现出异常的行为模式，与正常用户的行为有所不同。
* 异常检测：使用异常检测算法来识别异常点赞行为。这些算法可以基于统计方法、聚类方法或者深度学习方法。对于已知的刷赞模式，可以将其视为异常样本进行检测。
* 模型监测：建立监测模型，定期监测用户的行为，并与正常行为进行对比。如果发现某个用户的行为与正常用户明显不符，则可能涉及刷赞行为。
* 用户认证：对于涉嫌刷赞的用户，可以采取进一步的认证措施，比如要求用户进行验证码验证、手机验证等，以确保用户是真实的。
* 反作弊技术：应用一些反作弊技术，如设立人工智能审核系统、设立用户举报渠道等，以监测和严惩刷赞行为。

### **7、非递归进行中序遍历的代码：**

中序遍历是二叉树遍历的一种方式，其顺序为左子树 -> 根节点 -> 右子树。非递归进行中序遍历可以使用栈来实现，以下是Python代码示例：

class TreeNode:

def \_\_init\_\_(self, val=0, left=None, right=None):

self.val = val

self.left = left

self.right = right

def inorderTraversal(root):

if not root:

return []

result = []

stack = []

current = root

while current or stack:

# 将当前节点的所有左子节点压入栈中

while current:

stack.append(current)

current = current.left

# 弹出栈顶节点，加入结果列表，并处理右子节点

current = stack.pop()

result.append(current.val)

current = current.right

return result

# **哪吒规控算法实习生面试/10道**

## **1、深度学习的激活函数有哪些：**

1. 激活函数在深度学习中被用于引入非线性特性，常见的激活函数包括：

* Sigmoid函数（Logistic函数）
* Tanh函数（双曲正切函数）
* ReLU函数（线性整流函数）
* Leaky ReLU函数
* Parametric ReLU函数（PReLU）
* ELU函数（指数线性单元函数）
* Swish函数
* GELU函数（高斯误差线性单元函数）
* Softmax函数（用于多类别分类任务的激活函数）

## **2、深度学习训练的时候爆显存的原因有什么：**

1. 显存爆满通常是因为模型和数据导致的。主要原因包括：

* 模型大：使用过于庞大的模型，参数量太大，导致显存占用过多。
* 批量大小大：在训练过程中使用过大的批量大小，导致显存占用增加。
* 多样本输入：数据样本过多，导致在一次前向传播中需要处理的数据量过大。
* 多层级网络：深度较大的网络层级增加了计算量和显存需求。

## **3、Python的深浅拷贝：**

* 浅拷贝（Shallow Copy）：创建一个新的对象，但是只复制原始对象的引用而不复制其子对象。新对象与原对象共享子对象。
* 深拷贝（Deep Copy）：创建一个全新的对象，并递归复制原始对象及其所有子对象。新对象与原对象及其子对象完全独立。

在Python中，可以使用copy模块来实现深拷贝和浅拷贝。

## **4、三次握手**

1. 三次握手是TCP协议建立可靠连接的过程，确保客户端和服务器之间可以正确地发送和接收数据。其步骤如下：
2. 第一次握手（SYN=1，seq=x）：客户端发送一个带有SYN标志的数据包给服务器，并随机生成一个序列号x，表示请求建立连接。
3. 第二次握手（SYN=1，ACK=1，ack=x+1，seq=y）：服务器收到客户端的请求后，发送一个带有SYN和ACK标志的数据包给客户端，确认收到请求，并为自己也随机生成一个序列号y。
4. 第三次握手（ACK=1，ack=y+1）：客户端收到服务器的确认后，再发送一个带有ACK标志的数据包给服务器，表示连接已经建立。

经过三次握手，TCP连接建立完成，双方可以开始进行数据传输。

## **5、HTTP、HTTPS：**

* HTTP（Hypertext Transfer Protocol）：是一种用于传输超文本的协议。它是Web通信的基础，通过在客户端和服务器之间交换请求和响应来传输数据。然而，HTTP是明文协议，数据传输不加密，安全性较差。
* HTTPS（Hypertext Transfer Protocol Secure）：是HTTP的安全版本，通过使用SSL/TLS协议对HTTP的通信进行加密和认证，提供了更高的安全性。HTTPS在数据传输过程中对数据进行加密，确保通信过程中数据不被窃听或篡改。

## **6、TCP、UDP：**

* TCP（Transmission Control Protocol）：是一种面向连接的可靠传输协议。在数据传输之前，TCP会通过三次握手建立连接，并提供数据传输的流量控制、拥塞控制和错误校验等功能，确保数据可靠地传输到目标地址。
* UDP（User Datagram Protocol）：是一种面向无连接的不可靠传输协议。UDP在数据传输时不需要建立连接，数据包发送后不会进行确认，也不会进行重传。因此，UDP传输速度快，但不保证数据传输的可靠性。

## **7、数据链路层的作用：**

* 数据链路层是OSI模型中的第二层，负责在直连的网络节点之间传输数据帧。它的主要作用包括：
* 封装成帧：将网络层传递下来的数据包封装成数据帧，添加帧头和帧尾，便于在物理媒介上传输。
* 物理地址寻址：使用物理地址（MAC地址）来标识和定位设备，帮助数据帧在局域网内正确传送到目标设备。
* 媒体访问控制：协调多个设备共享同一物理媒介，避免冲突和碰撞。
* 差错检测：在传输过程中检测数据帧是否发生错误，并进行纠正。

## **8、进程与线程的区别：**

进程和线程都是程序执行的基本单元，但有以下区别：

进程：是独立的执行单位，拥有自己的独立内存空间和系统资源，进程间通信需要特殊的IPC机制，相互之间不会影响。

线程：是进程内的执行单位，共享所属进程的内存空间和资源，线程间通信相对容易，但也需要注意同步和资源竞争问题。

区别：

* 调度：在多线程环境下，线程的切换由操作系统的线程调度器负责。在多进程环境下，进程的切换由操作系统的进程调度器负责。线程切换比进程切换开销小。
* 并发性：多个线程可以在同一进程内并发执行，共享进程的资源。而多个进程之间的并发执行需要通过进程间通信（IPC）来实现。
* 安全性：由于多个线程共享同一进程的资源，必须注意线程同步和资源竞争问题。多进程之间不共享资源，相对更安全，但需要通过IPC进行通信。
* 灵活性：线程之间的通信和数据共享相对简单，但需要考虑线程同步问题。进程间通信相对复杂，但可以更好地实现模块化和隔离。

总的来说，进程适用于多个任务需要相对独立运行且安全性要求高的情况。线程适用于需要共享大量数据且并发执行的情况，适合用于提高程序的执行效率和响应性能。

## **9、什么是堆？什么是完全二叉树？**

* 堆：堆是一种特殊的数据结构，是一种经过排序的完全二叉树。在堆中，对于任意节点的值总是不小于（或不大于）其子节点的值，分别称为最大堆和最小堆。堆通常用于实现优先队列等数据结构。
* 完全二叉树：完全二叉树是一种二叉树，除了最后一层外，其他层的节点都被完全填满，并且最后一层的节点都靠左对齐。在完全二叉树中，从根节点到倒数第二层的节点都是满的，最后一层可能不满，但节点都尽可能地靠左排列。

## **10、路径规划算法有哪些？**

路径规划算法是指寻找从起始点到目标点的最优路径或者避开障碍物的路径。常见的路径规划算法包括：

* Dijkstra算法：用于求解单源最短路径问题，适用于权值非负的图。
* A\*算法（A star）：结合了Dijkstra算法和启发式搜索，可以在权值非负的图中找到最短路径，并通过启发式函数（估计函数）加速搜索。
* BFS算法（广度优先搜索）：用于无权图的最短路径搜索。
* DFS算法（深度优先搜索）：用于图的遍历，不一定能找到最短路径。
* RRT算法（Rapidly-exploring Random Tree）：用于高维空间的路径规划，常用于机器人运动规划等。
* PRM算法（Probabilistic Roadmap）：基于采样的方法，适用于高维空间的路径规划。

这些算法在不同的应用场景下有不同的优势和适用性。路径规划算法的选择取决于具体问题的约束条件和需求。

# **科大讯飞 飞星计划深度学习框架方向面试题11道**

### **1、cuda的block,grid,thread的关系**

CUDA编程模型中的线程、块和网格的关系: CUDA程序把代码执行的任务分成一个个线程(thread)，这些线程被组织成一个个块(block)，然后块再被组织成一个个网格(grid)。每个线程内执行相同的指令，但是操作不同的数据。每个块中的线程可以通过共享内存和同步机制进行通信，而不同块之间的线程不能直接进行通信。网格就是所有的块的集合，其实就是程序的总体执行框架。

### **2、cuda的共享内存了解吗，如何理解内存墙**

CUDA的共享内存是指一个线程块内的线程共享的内存。它有限且非常宝贵，但其访问速度快，常用来存放线程块内线程频繁访问的数据。内存墙是指线程并行运行时，由于内存访问模式不同，可能导致一些线程访问内存延迟较大，从而形成一个“内存墙”。

### **3、对c++封装和继承的理解**

C++中的封装是指把数据和操作数据的函数绑定在一起，形成一个“对象”，而对象的内部数据只能通过对象的方法来操作。继承则是子类继承父类的属性和方法，实现代码复用。

### **4、static和const的理解**

在C++中，static关键字可以用来定义静态成员，该成员不依赖于任何对象独立存在。const关键字用来定义常量，表示其值不能被改变。

### **5、c++ inline函数的作用**

inline函数是C++中的一种优化技术。通过在函数前加上inline关键字，编译器会在编译时将函数的调用直接替换为函数体的代码，避免了函数调用的开销。

### **6、10000个数字选前100个数用什么方法**

对于10000个数字选出前100个数，可以使用最小堆。首先对前100个数建立最小堆，然后遍历后面的数字，如果比堆顶元素大就替换堆顶元素并调整堆，最后堆中的元素就是最大的100个数。

### **7、堆排序 快排的时间复杂度 最坏情况的复杂度 什么情况会是最坏情况**

堆排序和快速排序的平均时间复杂度都是O(nlogn)。最坏情况下，堆排序的时间复杂度仍是O(nlogn)，而快速排序的时间复杂度变为O(n^2)。快速排序最坏的情况是输入数组已经完全有序或完全逆序。

### **8、宏定义和inline函数区别**

宏定义在预处理阶段就将所有的宏替换掉，而inline函数在编译阶段进行替换。宏定义只是简单的文本替换，没有类型检查，而inline函数有完全的函数特性，包括类型检查、作用域规则等。

### **9、c++智能指针理解**

C++智能指针是一种对象，它能保证在任何情况下都能正确地删除对象。它可以自动管理内存，防止内存泄漏。常见的有unique\_ptr, shared\_ptr, weak\_ptr等。

### **10、c++多态实现方式有哪些**

C++的多态主要通过虚函数实现，包括纯虚函数(定义接口)和普通虚函数(允许子类覆盖父类的实现)。通过指向基类的指针或引用，可以调用到子类的方法，实现动态绑定。

### **11、如何判断链表是否有环**

快慢指针法：设置两个指针，一个快指针和一个慢指针，它们都从链表头部开始走，快指针每次走两步，慢指针每次走一步，如果链表中存在环，那么快指针和慢指针最终一定会相遇；如果链表中不存在环，那么快指针会首先到达链表尾部。

# **万物心选算法实习面试题7道|**

### **1、Adam优化器和SGD的区别：**

Adam优化器和随机梯度下降（SGD）是两种常用的优化算法。它们的主要区别在于更新参数的方式和对梯度的处理方式。

Adam优化器使用了自适应学习率的方法，并结合了动量的概念。它维护了每个参数的自适应学习率，并使用动量来加速参数更新。Adam通过计算梯度的一阶矩估计（均值）和二阶矩估计（方差）来调整学习率。这种自适应学习率的调整可以帮助Adam更好地适应不同参数的特性，并且通常能够更快地收敛。

相比之下，SGD仅使用固定的学习率来更新参数。它直接使用当前的梯度来更新参数，而没有考虑其他信息。这种简单的更新方式可能导致收敛速度较慢，特别是在参数空间存在不同尺度的情况下。

总的来说，Adam相对于SGD来说更加智能化和自适应，能够更快地收敛到局部最优解，并且通常能够在训练过程中保持较小的学习率。

### **2、分类问题为什么用交叉熵损失函数不用均方误差（MSE）：**

交叉熵损失函数通常在分类问题中使用，而均方误差（MSE）损失函数通常用于回归问题。这是因为分类问题和回归问题具有不同的特点和需求。

分类问题的目标是将输入样本分到不同的类别中，输出为类别的概率分布。交叉熵损失函数可以度量两个概率分布之间的差异，使得模型更好地拟合真实的类别分布。它对概率的细微差异更敏感，可以更好地区分不同的类别。此外，交叉熵损失函数在梯度计算时具有较好的数学性质，有助于更稳定地进行模型优化。

相比之下，均方误差（MSE）损失函数更适用于回归问题，其中目标是预测连续数值而不是类别。MSE损失函数度量预测值与真实值之间的差异的平方，适用于连续数值的回归问题。在分类问题中使用MSE损失函数可能不太合适，因为它对概率的微小差异不够敏感，而且在分类问题中通常需要使用激活函数（如sigmoid或softmax）将输出映射到概率空间，使得MSE的数学性质不再适用。

综上所述，交叉熵损失函数更适合分类问题，而MSE损失函数更适合回归问题。

### **3、决策树的生成过程：**

决策树是一种常见的机器学习算法，用于解决分类和回归问题。下面是决策树的生成过程的简要介绍：

1. 选择最佳特征：从训练数据集中选择一个最佳的特征来作为当前节点的分裂标准。通常使用一些评价指标（如信息增益、基尼指数等）来衡量特征的好坏。
2. 分裂节点：根据选择的特征将当前节点分裂成多个子节点，每个子节点对应特征的一个取值或一个值的范围。
3. 递归生成子树：对于每个子节点，重复步骤1和步骤2，递归地生成子树，直到满足终止条件。终止条件可以是节点中的样本数量小于某个阈值，或者节点中的样本属于同一类别，或者达到了树的最大深度等。
4. 构建决策树：通过递归生成子树，最终构建出完整的决策树。
5. 剪枝（可选）：为了避免过拟合，可以进行剪枝操作，去掉一些决策树的分支或节点。

决策树的生成过程基于对特征空间的划分，通过选择最佳特征来使得每个子节点的样本更加纯净，即属于同一类别。这样生成的决策树可以用于预测新样本的类别或回归值。

### **4、什么是信息增益**

信息增益是在决策树算法中用于选择最佳特征的一种评价指标。在决策树的生成过程中，选择最佳特征来进行节点的分裂是关键步骤之一，信息增益可以帮助确定最佳特征。

信息增益衡量了在特征已知的情况下，将样本集合划分成不同类别的纯度提升程度。它基于信息论的概念，使用熵来度量样本集合的不确定性。具体而言，信息增益是原始集合的熵与特定特征下的条件熵之间的差异。

在决策树的生成过程中，选择具有最大信息增益的特征作为当前节点的分裂标准，可以将样本划分为更加纯净的子节点。信息增益越大，意味着使用该特征进行划分可以更好地减少样本集合的不确定性，提高分类的准确性。

### **5、动态规划的核心思想**

动态规划是一种解决多阶段决策问题的优化方法，其核心思想是将原问题分解为多个重叠子问题，并通过保存子问题的解来避免重复计算，从而提高算法的效率。

动态规划通常适用于具有重叠子问题和最优子结构性质的问题。最优子结构指的是原问题的最优解可以通过子问题的最优解来构造。而重叠子问题指的是在问题的求解过程中，同一个子问题会被多次计算。

### **6、直观地解释一下Transformer注意力机制**

Transformer是一种用于序列建模的深度学习模型，广泛应用于自然语言处理等领域。其中的注意力机制是Transformer的核心组成部分。

在Transformer中，注意力机制用于在序列中建立全局的关联性，将每个位置的表示与其他位置的表示进行交互。直观地解释注意力机制如下：

假设有一个输入序列，其中包含多个位置，每个位置都有一个表示向量。注意力机制通过计算每个位置与其他位置的关联权重，从而决定每个位置在表示时的重要性。

通过注意力机制，Transformer能够在序列中捕捉到全局的关联性，可以将每个位置的表示与其他位置的表示进行交互和融合，从而更好地捕捉序列中的重要信息和依赖关系。

### **7、一维卷积的作用**

* 在不改变特征图尺寸的前提下去改变通道数（升维降维）；
* 增强了网络局部模块的抽象表达能力；
* 在不增加感受野的情况下，让网络加深，从而引入更多的非线性。

# **网易NLP大模型实习面试题8道**

### **1、文本生成的几大预训练任务？**

* GPT（Generative Pre-trained Transformer）系列：包括GPT、GPT-2、GPT-3等。这些模型使用Transformer架构进行预训练，在大规模语料上学习语言模型，能够生成连贯、具有语义的文本。
* BART（Bidirectional and Auto-Regressive Transformer）：BART是一种基于Transformer的生成式预训练模型。它通过自回归解码器实现文本生成，通过自编码器预训练目标来重构输入文本，能够生成流畅、连贯的文本。
* T5（Text-to-Text Transfer Transformer）：T5是一种通用的文本生成模型，使用了编码器-解码器结构。它将不同的自然语言处理（NLP）任务转换为文本到文本的转换任务，可用于机器翻译、摘要生成、问题回答等多个NLP任务。
* XLNet：XLNet是一种基于Transformer架构的预训练模型，采用了自回归和自编码器的组合方式进行训练。它在语言建模任务上引入了全局的上下文信息，能够生成更加准确和连贯的文本。
* UniLM（Unified Language Model）：UniLM是一种多任务学习的预训练模型，将不同的自然语言处理任务转化为统一的生成式任务。它可以用于文本摘要、问答系统、机器翻译等多个任务。

### **2、多模态中常见的SOTA模型有哪些？**

* Vision Transformer (ViT): 将自注意力机制引入计算机视觉领域，通过将图像划分为图像补丁并应用Transformer模型，实现了在图像分类和目标检测等任务上的出色表现。
* CLIP (Contrastive Language-Image Pretraining): 结合了图像和文本的对比学习，通过训练一个模型，使其能够根据图像和文本之间的相互关系进行推理，实现了图像与文本之间的联合理解和表示学习。
* UNITER (UNiversal Image-Text Representation): 使用Transformer架构，联合学习图像和文本表示，提供了一个通用的图像和文本特征提取框架，适用于多个视觉和语言任务。
* LXMERT (Cross-Modal Transformer): 结合了视觉和语言信息，通过Transformer模型对图像和文本进行交互学习，可以用于视觉问答、图像描述生成等任务。

### **3、介绍一下stable diffusion的原理。**

stable diffusion是一种生成模型，其原理基于Langevin动力学和扩散过程。其核心思想是通过多次迭代，逐渐将噪声信号演化为目标分布所对应的样本。具体原理如下：

* 初始化噪声信号为服从高斯分布的随机向量。
* 通过一系列的演化步骤，将噪声信号迭代地转化为目标分布的样本。每一步中，将当前噪声信号与目标分布的梯度信息结合，通过Langevin动力学方程进行更新，使噪声信号逐渐接近目标分布。
* 迭代的次数越多，噪声信号越接近目标分布，并最终生成目标分布的样本。

stable diffusion通过合理的选择演化步长和迭代次数，可以在生成样本的过程中平衡样本质量和生成速度。

### **4、instructGPT的原理，讲讲rlhf和reward。**

instructGPT是一种基于强化学习的文本生成模型，其核心原理涉及两个概念：RLHF（Reinforcement Learning from Human Feedback）和reward shaping（奖励塑造）。

* RLHF：在训练instructGPT时，首先使用有人类生成的示例对模型进行预训练。然后，通过与人类评估者进行交互，收集评估结果，以创建一个用于强化学习的数据集。该数据集包含了人类评估者对生成结果的评分或反馈，用于指导模型的强化学习训练。
* Reward shaping：为了更好地引导模型的训练，reward shaping用于调整模型的奖励信号。通过将人类评估者的反馈与模型生成的文本进行比较，可以计算出一个差异度量，用作奖励信号的一部分。这样，模型可以根据这个奖励信号进行训练，并进行强化学习的训练。模型根据当前的状态（对话历史）生成文本，并通过奖励信号来评估生成文本的质量。模型的目标是最大化预期累积奖励，从而生成更高质量的文本。

通过RLHF和reward shaping的结合，instructGPT能够通过人类评估者的反馈指导模型的生成过程，并逐步提升生成文本的质量和一致性。

### **5、讲讲T5和Bart的区别，讲讲bart的DAE任务。**

T5（Text-to-Text Transfer Transformer）和Bart（Bidirectional and Auto-Regressive Transformer）是两个常见的预训练模型，它们之间的区别如下：

* T5是一种基于Transformer的通用文本生成模型。T5的训练目标是将不同的自然语言处理（NLP）任务统一为文本到文本的转换任务。它采用了编码器-解码器结构，通过输入一个自然语言文本，输出另一个相关的自然语言文本，可以应用于机器翻译、摘要生成、问题回答等多个NLP任务。
* Bart是建立在T5模型基础上的一个变种，它专注于生成式任务。Bart模型使用了自回归解码器，通过训练一个自编码器来重构原始文本，同时采用了标准的语言模型预训练目标，从而使得生成的文本更加流畅和连贯。Bart的主要应用领域包括文本生成、摘要生成、对话系统等。

在任务类型上，T5更加通用，适用于多种NLP任务的文本转换，而Bart则更加专注于生成式任务，并且在生成文本的质量和连贯性上有所优化。

关于Bart的DAE（Denoising AutoEncoder）任务，它是Bart模型的一种预训练目标。DAE任务要求模型从输入的有噪声的文本中恢复原始的无噪声文本。通过在训练过程中向输入文本中添加噪声，并要求模型重建无噪声的文本，Bart可以学习到更好的文本表示和重构能力，从而提高生成文本的质量和准确性。

### **6、讲讲Bart和Bert的区别。**

Bart和Bert是两个不同的预训练模型，它们之间的区别如下：

* Bart是一种基于Transformer的生成式预训练模型，主要应用于文本生成、摘要生成、对话系统等任务。Bart采用了自回归解码器，通过自编码器预训练目标来重构输入文本，从而生成流畅、连贯的文本。
* Bert（Bidirectional Encoder Representations from Transformers）是一种双向的预训练模型，用于生成文本的上下文表示。与Bart不同，Bert采用了双向的Transformer编码器，通过将上下文的信息融合到表示中，提供了更全面的语境理解能力。Bert主要应用于词嵌入、文本分类、命名实体识别等任务。

总体上说，Bart侧重于生成式任务和文本生成，而Bert侧重于上下文表示和语境理解。它们在模型结构和应用场景上存在一定的差异。

### **7、对比学习负样本是否重要？负样本构造成本过高应该怎么解决？**

对比学习中负样本的重要性取决于具体的任务和数据。负样本可以帮助模型学习到样本之间的区分度，从而提高模型的性能和泛化能力。然而，负样本的构造成本可能会较高，特别是在一些领域和任务中。

为了解决负样本构造成本过高的问题，可以考虑以下方法：

* 降低负样本的构造成本：通过设计更高效的负样本生成算法或采样策略，减少负样本的构造成本。例如，可以利用数据增强技术生成合成的负样本，或者使用近似采样方法选择与正样本相似但不相同的负样本。
* 确定关键负样本：根据具体任务的特点，可以重点关注一些关键的负样本，而不是对所有负样本进行详细的构造。这样可以降低构造成本，同时仍然能够有效训练模型。
* 迁移学习和预训练模型：利用预训练模型或迁移学习的方法，可以在其他领域或任务中利用已有的负样本构造成果，减少重复的负样本构造工作。

### **8、介绍一下lora的原理和ptuning的原理。**

Lora方法的核心是在大型语言模型上对指定参数增加额外的低秩矩阵，也就是在原始PLM旁边增加一个旁路，做一个降维再升维的操作。并在模型训练过程中，固定PLM的参数，只训练降维矩阵A与升维矩阵B。

ptuning方法的核心是使用可微的virtual token替换了原来的discrete tokens，且仅加入到输入层，并使用prompt encoder（BiLSTM+MLP）对virtual token进行编码学习。

# **科大讯飞飞星计划-AI算法（NLP方向）面试题7道**

### **1、jieba分词的原理**

jieba分词是一种中文文本分词工具，主要是基于词频和词典匹配。核心原理包含以下几点：

* 基于前缀词典实现高效的词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图 (DAG)
* 采用了动态规划查找最大概率路径, 找出基于词频的最大切分组合
* 对于未登录词，采用了基于汉字成词能力的 HMM 模型，使用了 Viterbi 算法。

### **2、word2vec的原理，怎么训练的？**

word2vec是一种用于将词语表示为向量的模型，它的核心思想是通过预测上下文或目标词来学习词向量。word2vec模型有两种训练方法：CBOW（Continuous Bag-of-Words）和Skip-gram。

CBOW模型的训练过程是，给定一个窗口大小，在一个句子中，将目标词的上下文作为输入，预测目标词本身。模型的输入是上下文词向量的累加，通过一个浅层神经网络进行训练，最终得到词向量。

Skip-gram模型的训练过程是，给定一个目标词，在一个句子中，将目标词作为输入，预测它的上下文。模型通过一个浅层神经网络进行训练，最终得到词向量。

在训练word2vec模型时，可以使用大规模的语料库进行迭代训练。训练过程中使用梯度下降等优化算法来更新模型的参数，使得模型能够更好地预测上下文或目标词。

### **3、ChatGPT是怎么训练出来的**

ChatGPT是由OpenAI开发的大型语言模型，基于GPT-3.5架构。ChatGPT的训练过程可以分为两个主要阶段：预训练和微调。

在预训练阶段，使用了大规模的文本数据集来训练模型。这个数据集包含了从互联网上抓取的各种文本，例如网页、书籍、维基百科等等。预训练的目标是让模型学习语言的各种模式和结构，以及推断单词和句子之间的关系。为了实现这一点，预训练使用了一个自监督学习的方法。这意味着模型在训练时没有明确的监督信号，而是通过自动生成任务来进行学习。例如，模型可能被要求预测一个句子中被遮挡的单词是什么，或者预测给定上下文的下一个单词是什么。

预训练完成后，模型就可以理解和生成文本，但它还没有具体的知识或任务特定的指导。为了使ChatGPT更适合对话任务，需要进行微调。

在微调阶段，使用人工编写的对话数据集来对模型进行有监督的训练。这些对话数据集通常由人类操作员与模型进行交互生成，以创建逼真的对话场景。这些操作员会扮演用户和模型之间的角色，向模型提问并提供回答。模型通过与这些对话进行比较，并根据预期输出进行调整，逐渐改善其回答的质量和流畅度。

微调的目的是根据特定的任务和应用场景来调整模型，使其更符合实际需求。这个阶段需要大量的迭代和调试，以改进模型的性能和适应性。

总的来说，ChatGPT的训练过程包括预训练和微调两个阶段，通过大规模的文本数据预训练模型，然后使用人工编写的对话数据集进行微调，使其适应对话任务并提供准确、连贯的回答。

### **4、BERT模型简述：**

BERT（Bidirectional Encoder Representations from Transformers）是一种基于Transformer架构的预训练语言模型。与传统的语言模型只使用左侧或右侧的上下文信息不同，BERT模型采用了双向编码器，同时利用了左侧和右侧的上下文信息。

BERT模型的预训练阶段采用了两个任务：掩码语言建模（Masked Language Model，MLM）和下一句预测（Next Sentence Prediction，NSP）。MLM任务类似于ChatGPT的预训练过程中的任务，随机掩盖输入文本中的一些词，要求模型预测这些被掩盖的词。NSP任务是要求模型判断两个句子是否是连续的。

BERT模型通过预训练来学习通用的语言表示，然后可以在各种下游任务上进行微调。在微调阶段，可以将BERT模型用于文本分类、命名实体识别、问答等多种自然语言处理任务，通过微调模型的参数，使其适应特定任务的需求。

### **5、PyTorch中的train和eval模块：**

在PyTorch中，train和eval模块通常用于模型的训练和评估。

train模块主要用于模型的训练过程。在训练过程中，train模块会迭代地将训练数据输入到模型中，计算模型的损失（通常使用损失函数来度量模型的预测结果与真实标签之间的差异），然后使用反向传播算法更新模型的参数，以减小损失函数的值。训练过程还包括设置优化器、学习率调度器等，以便对模型进行参数优化。

eval模块主要用于模型的评估过程。在评估过程中，eval模块会使用验证集或测试集的数据对训练好的模型进行评估。评估过程通常包括将测试数据输入模型中进行推理，然后根据评价指标（如准确率、召回率、F1分数等）来评估模型的性能。

### **6、Python中字典的结构及实现方式：**

在Python中，字典（Dictionary）是一种可变的数据结构，用于存储键-值对（key-value pairs）。字典中的键是唯一的，而值可以重复。字典的实现方式是哈希表（Hash Table），也称为散列表。

Python的字典通过哈希表实现，具有快速的查找和插入操作。当需要访问字典中的值时，根据键经过哈希函数计算得到索引，然后通过索引在内部数组中快速定位到对应的值。

哈希表是一种根据键直接访问值的数据结构，通过将键映射到哈希函数生成的索引位置来实现高效的键值查找。哈希函数将键转换成一个固定大小的整数，该整数用作哈希表的索引。

### **7、有一组无序数组，如何取前10个最大的数：**

如果给定一组无序数组，可以采用以下步骤来取前10个最大的数：

将无序数组进行排序，可以选择快速排序、堆排序或其他适合的排序算法。排序的时间复杂度为O(nlogn)，其中n是数组的长度。

对排序后的数组，从末尾开始取前10个元素，即为前10个最大的数。

举例来说，假设有一个名为nums的无序数组，可以使用Python的内置函数sorted()进行排序，并使用切片操作获取前10个最大的数：

sorted\_nums = sorted(nums, reverse=True)

top\_10 = sorted\_nums[:10]

# **滴滴秋储风控算法面试题5道|**

### **1、样本不平衡问题怎么处理的，有什么方法？**

* 重采样方法：通过欠采样（随机删除多数类样本）或过采样（复制少数类样本或生成合成样本）来平衡样本分布。常见的方法包括随机欠采样、SMOTE（Synthetic Minority Over-sampling Technique）等。
* 类别权重调整：通过调整分类算法中不同类别的权重，使得分类器更加关注少数类样本。一些分类算法（如逻辑回归、支持向量机）提供了设置类别权重的参数。
* 集成方法：使用集成学习算法，如随机森林或梯度提升树（例如XGBoost和LightGBM），能够处理样本不平衡问题。这是因为集成算法可以从多个子模型中综合考虑不同类别的情况。

### **2、快速排序时间复杂度？稳定性怎么样？为什么是不稳定的**

快速排序（QuickSort）的平均时间复杂度为O(n log n)，最坏情况下的时间复杂度为O(n^2)。快速排序的稳定性取决于具体的实现方式。通常情况下，快速排序是不稳定的，因为在分区过程中，元素的相对顺序可能会改变。

快速排序的不稳定性是由于分区过程中的元素交换操作引起的。在分区过程中，我们选择一个基准元素（通常是第一个或最后一个元素），将小于基准的元素放在基准的左边，大于基准的元素放在基准的右边。在交换元素的过程中，相等元素的相对顺序可能发生改变，从而导致排序结果的不稳定性。

### **3、各种评估指标？**

* 准确率（Accuracy）：分类正确的样本数与总样本数之比。
* 精确率（Precision）：在被分类为正例的样本中，真正例的比例。
* 召回率（Recall）：在所有正例中，被正确分类为正例的比例。
* F1分数（F1 Score）：精确率和召回率的调和均值，综合考虑了两者。
* ROC曲线（Receiver Operating Characteristic curve）和AUC（Area Under the Curve）：绘制真正例率（TPR）和假正例率（FPR）之间的关系曲线，并计算曲线下面积。
* 混淆矩阵（Confusion Matrix）：展示分类结果的四个统计数字：真正例（True Positive, TP）、假正例（False Positive, FP）、真反例（True Negative, TN）和假反例（False Negative, FN）。

### **4、xgboost算法介绍？**

XGBoost是一种基于梯度提升树（Gradient Boosting Tree）算法的集成学习模型，它在梯度提升树的基础上进行了优化和改进。

XGBoost的主要特点和优势包括：

* 正则化：XGBoost引入了正则化项来控制模型的复杂度，防止过拟合。它通过控制叶子节点的权重以及树的复杂度来实现正则化，避免单棵树过于复杂。
* 自定义损失函数：XGBoost允许用户自定义损失函数，从而能够灵活地适应不同的任务需求。
* 特征选择：XGBoost通过计算特征的重要性得分，可以帮助识别和选择最具预测能力的特征。
* 高效并行：XGBoost在训练过程中支持并行计算，可以利用多线程和分布式计算加速模型的训练。
* 缺失值处理：XGBoost能够自动学习如何处理缺失值，无需对缺失值进行特殊处理。
* 特征交叉：XGBoost支持特征交叉，可以通过组合特征创建新的特征，提升模型的表达能力。

### **5、评分卡建模全流程？**

1. 数据准备：收集并清洗需要建模的数据，包括缺失值处理、异常值处理、变量转换等。
2. 变量选择：通过特征分析、相关性分析、变量筛选方法（如IV值、WOE值）等，选择最具预测能力的变量作为建模输入。
3. 数据切分：将数据集切分为训练集和验证集（测试集），用于模型训练和验证。
4. 建立评分卡模型：根据业务需求选择适当的分类算法，如逻辑回归、决策树等，建立评分卡模型。在建模过程中，需要进行特征工程、变量转换、变量融合等步骤。
5. 模型评估和验证：使用验证集对模型进行评估，包括计算评分卡的各项指标，如准确率、精确率、召回率、F1分数等。根据评估结果对模型进行调优和验证。

# **万得ai算法工程师一面面试题6道|含解析**

### **问题1、dataloader和dataset的区别。**

DataLoader和 Dataset 是 PyTorch 中用于处理数据的两个重要概念：

Dataset 是一个抽象类，用于表示数据集，通常需要用户自定义，包含了数据的读取、预处理等操作。DataLoader 是一个数据加载器，用于将 Dataset 中的数据按照指定的 batch\_size、shuffle 等参数加载到模型中进行训练或推理。DataLoader 可以自动实现多线程数据加载和数据批处理等功能。

### **问题2、深度学习中，常见的损失函数有哪些？**

均方误差损失函数 (Mean Squared Error, MSE)

交叉熵损失函数 (Cross Entropy Loss)

感知损失函数 (Perceptron Loss)

Hinge Loss

KL 散度损失函数 (Kullback-Leibler Divergence Loss)

Huber Loss

余弦相似度损失函数 (Cosine Similarity Loss) 等。

### **问题3、介绍下yolov8算法的模块。**

yolov8算法的模块：

Backbone：通常使用一些预训练的卷积神经网络 (CNN)，如Darknet、ResNet、EfficientNet 等，用于提取图像的特征。

Neck：用于进一步处理和整合特征，通常包括一些卷积层和池化层等。

Head：目标检测的关键部分，包括预测目标的边界框、类别以及置信度等。

### **问题4、介绍下什么是nms。**

NMS (Non-Maximum Suppression，非极大值抑制) 是目标检测领域常用的一种算法，用于去除检测到的重叠较多的边界框，保留最具代表性的边界框。其核心思想是保留置信度最高的边界框，并去除与其 IoU (Intersection over Union) 超过阈值的其他边界框。

### **问题5、CV中数据增强的方法有哪些？**

在计算机视觉中，常用的数据增强方法包括：

随机裁剪 (Random Cropping)

随机翻转 (Random Flipping)

随机旋转 (Random Rotation)

色彩变换 (Color Jittering)

尺度缩放 (Scale Augmentation)

平移 (Translation)

对比度增强 (Contrast Enhancement)

亮度调整 (Brightness Adjustment)

添加噪声 (Adding Noise) 等。

### **问题6、讲一下batchnorm的计算过程。**

1）对于每个批次的输入数据，计算其均值和方差。

2）对输入数据进行标准化，即减去均值并除以标准差。

3）使用学习参数（拉伸因子和偏移量）进行线性变换，使得数据重新具有适当的比例和偏移。

BatchNorm 可以在训练过程中通过批次数据的统计信息来进行标准化，也可以在推理过程中使用移动平均来估计整个数据集的统计信息，以实现更好的泛化能力和稳定性。

# **理想大模型实习面试题6道|含解析**

### **1、LLAMA 和 ChatGLM 的区别。**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模型 | 训练数据 | 训练数据量 | 模型参数量 | 词表大小 |
| LLaMA | 以英语为主的拉丁语系，不包含中日韩文 | 1T/1.4T tokens | 7B、13B、33B、65B | 32000 |
| ChatGLM-6B | 中英双语，中英文比例为 1:1 | 1T tokens | 6B | 130528 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模型 | 模型结构 | 位置编码 | 激活函数 | layer norm |
| LLaMA | Casual decoder | RoPE | SwiGLU | Pre RMS Norm |
| ChatGLM-6B | Prefix decoder | RoPE | GeGLU | Post Deep Norm |

### **2、BatchNorm 和 LayerNorm 什么区别。**

layernorm和batchnorm的区别：LN中同层神经元输入拥有相同的均值和方差，不同的输入样本有不同的均值和方差；

BN中则针对不同神经元输入计算均值和方差，同一个batch中的输入拥有相同的均值和方差。

LN不依赖于batch的大小和输入sequence的深度，因此可以用于batchsize为1和RNN中对变长的输入sequence的normalize操作。

由于NLP中的文本输入一般为变长，所以使用layernorm更好。

### **3、Bert 的参数量是怎么决定的。**

Bert（Bidirectional Encoder Representations from Transformers）的参数量由其模型结构以及隐藏层的大小、层数等超参数所决定。具体来说，Bert 模型由多个 Transformer Encoder 层组成，每个 Encoder 层包含多个注意力头以及前馈神经网络层。因此，Bert 的参数量主要由这些层的数量、每层的隐藏单元数、注意力头的数量等因素决定。

### **4、p tuning v2 和 prompt tuning 的区别。**

Prompt tuning是之前其他论文提出的一种方法，通过冻结语言模型仅去调整连续的prompts，在参数量超过10B的模型上，效果追上了fine-tune，但是在normal-sized模型上表现不好，并且无法解决序列标注任务。针对这两个问题，作者提出了P-tuning v2。

P-Tuning V2在P-Tuning V1的基础上进行了下述改进：

* 在每一层都加入了Prompts tokens作为输入，而不是仅仅加在输入层，这与Prefix Tuning的做法相同。这样得到了更多可学习的参数，且更深层结构中的Prompt能给模型预测带来更直接的影响。
* 去掉了重参数化的编码器。在 P-tuning v2 中，作者发现重参数化的改进很小，尤其是对于较小的模型，同时还会影响模型的表现。
* 针对不同任务采用不同的提示长度。提示长度在提示优化方法的超参数搜索中起着核心作用。在实验中，我们发现不同的理解任务通常用不同的提示长度来实现其最佳性能，这与Prefix-Tuning中的发现一致，不同的文本生成任务可能有不同的最佳提示长度。
* 可选的多任务学习。先在多任务的Prompt上进行预训练，然后再适配下游任务。一方面，连续提示的随机惯性给优化带来了困难，这可以通过更多的训练数据或与任务相关的无监督预训练来缓解；另一方面，连续提示是跨任务和数据集的特定任务知识的完美载体。

### **5、多头注意力机制和单个注意力机制时间复杂度会变吗？**

多头注意力机制和单个注意力机制的时间复杂度都是

[O(n^2d)](https://zhuanlan.zhihu.com/p/690537109/edit)，其中 n 是序列长度，d 是每个词向量的维度。因为注意力机制涉及计算注意力分数的所有词对，因此时间复杂度与序列长度的平方成正比。无论是多头还是单个注意力机制，时间复杂度都是相同的。

6、大模型微调过程中如何避免灾难性遗忘？

在微调大模型的过程中，确实可能会遇到灾难性遗忘的问题，即模型在优化某一特定任务时，可能会忘记之前学到的其他重要信息或能力。为了缓解这种情况，可以采用以下几种策略：

（1）重新训练：通过使用所有已知数据重新训练模型，可以使其适应数据分布的变化，从而避免遗忘。

（2）增量学习：增量学习是一种在微调过程中逐步添加新数据的方法。通过增量学习，大模型可以在不忘记旧知识的情况下学习新数据。

（3）知识蒸馏：知识蒸馏是一种将老模型的知识传递给新模型的方法。通过训练一个教师模型来生成数据标注或权重，然后将标注或权重传递给新模型进行训练，可以避免灾难性遗忘。

（4）正则化技术：限制模型参数的变化范围，从而减少遗忘，使得大模型在微调过程中保持稳定性。

（5）使用任务相关性数据：如果可能的话，尽量使用与原始任务相关或相似的数据进行微调。这样，模型在优化新任务时，更容易与先前学到的知识建立联系。

# **小米暑期实习NLP算法工程师面试题8道|含解析**

### **1、介绍一下Bert模型。**

Bert（Bidirectional Encoder Representations from Transformers）是一种基于Transformer架构的预训练语言表示模型。与传统的基于循环神经网络的语言模型不同，Bert利用了Transformer的自注意力机制，能够在处理语言任务时更好地捕捉上下文信息。Bert的预训练过程包括两个阶段：Masked Language Model（MLM）和Next Sentence Prediction（NSP）。通过这两个任务，Bert能够学习到单词之间的语义关系和句子之间的连贯性，从而在各种自然语言处理任务中取得了很好的效果。

### **2、GPT和Bert的区别。**

GPT（Generative Pre-trained Transformer）和Bert在架构上有所不同。GPT是一个解码器架构，只使用了Transformer的解码器部分，而Bert则是编码器架构，只使用了Transformer的编码器部分。

在预训练任务上，GPT使用了单向的语言模型任务，即给定前文生成下一个单词，而Bert则利用了双向的语言模型任务，包括了Masked Language Model（MLM）和Next Sentence Prediction（NSP）。

在应用上，由于架构和预训练任务的不同，GPT通常用于生成式任务，如文本生成、对话生成等，而Bert则更适用于各种下游任务的特征提取和Fine-tuning，如文本分类、命名实体识别等。

### **3、介绍一下Roberta，它的改进点在哪。**

* 使用了更大的训练数据集，更长的训练时间和更大的批次大小，从而提高了模型的性能。
* 改变了训练过程中的输入表示方式，将句子对之间的间隔用额外的特殊符号表示，从而消除了Next Sentence Prediction任务对模型的限制。
* 使用了动态的词表大小，即在不同训练阶段动态调整词表大小，使得模型能够更好地利用数据集中的低频词。
* 对训练过程中的超参数进行了优化，包括学习率、批次大小等。

### **4、Transformer和LSTM的结构与原理**

Transformer是一种基于自注意力机制的架构，由编码器和解码器组成，适用于各种序列到序列的任务。其核心是自注意力机制，可以同时考虑输入序列中所有位置的信息，解决了长距离依赖问题。

LSTM（Long Short-Term Memory）是一种循环神经网络的变体，具有门控机制，可以在处理长序列数据时有效地捕捉长期依赖关系。LSTM包含输入门、遗忘门和输出门，通过控制信息的流动来实现长期记忆和遗忘。

### **5、介绍RLHF技术。**

RLHF（Reinforcement Learning from Human Feedback）是一种强化学习技术，旨在通过人类反馈来加速模型训练和优化过程。在RLHF中，模型根据其生成的动作和环境的反馈来调整策略，而不是根据预先定义的奖励函数。这种方法使得模型能够从人类专家的经验中快速学习，适用于各种任务，如游戏、机器人控制等。

### **6、介绍下注意力机制。**

注意力机制是一种机制，用于在处理序列数据时动态地给予不同位置的输入不同的权重。在Transformer模型中，注意力机制通过计算查询、键和值之间的关联性来实现，从而在编码器和解码器之间传递信息并捕捉输入序列中的关键信息。自注意力机制允许模型在不同位置上关注不同程度的信息，有助于解决长距离依赖问题。

### **7、寻找将一个单词变为另一个单词所需的最少操作步数，动态规划题。**

这是一个经典的动态规划问题，可以采用动态规划算法来解决。首先定义状态dp[i][j]表示将单词A的前i个字符变为单词B的前j个字符所需的最少操作步骤。然后根据以下情况进行状态转移：

如果A的第i个字符和B的第j个字符相同，则不需要操作，即dp[i][j] = dp[i-1][j-1]。

如果A的第i个字符和B的第j个字符不同，则可以进行替换、插入或删除操作，取三者中的最小值并加1，即dp[i][j] = min(dp[i-1][j-1], dp[i][j-1], dp[i-1][j]) + 1。

def min\_edit\_distance(word1, word2):

m, n = len(word1), len(word2)

dp = [[0] \* (n + 1) for \_ in range(m + 1)]

for i in range(m + 1):

dp[i][0] = i

for j in range(n + 1):

dp[0][j] = j

for i in range(1, m + 1):

for j in range(1, n + 1):

if word1[i - 1] == word2[j - 1]:

dp[i][j] = dp[i - 1][j - 1]

else:

dp[i][j] = min(dp[i - 1][j - 1], dp[i][j - 1], dp[i - 1][j]) + 1

return dp[m][n]

word1 = "intention"

word2 = "execution"

print(min\_edit\_distance(word1, word2)) # Output: 5

### **8、合并链表。**

合并两个已排序的链表可以采用递归或迭代的方式来实现。递归的方式相对简单直观，迭代方式则需要借助额外的指针来操作。

class ListNode:

def \_\_init\_\_(self, val=0, next=None):

self.val = val

self.next = next

def merge\_two\_lists(l1, l2):

dummy = ListNode()

current = dummy

while l1 and l2:

if l1.val < l2.val:

current.next = l1

l1 = l1.next

else:

current.next = l2

l2 = l2.next

current = current.next

current.next = l1 if l1 else l2

return dummy.next

# Example usage:

l1 = ListNode(1, ListNode(2, ListNode(4)))

l2 = ListNode(1, ListNode(3, ListNode(4)))

merged = merge\_two\_lists(l1, l2)