一、基础篇

1、目前主流的开源模型体系有哪些?

Transformer 体系：由 Google 提出的 Transformer 模型及其变体，如 BERT、GPT 等。

PyTorch Lightning：一个基于 PyTorch 的轻量级深度学习框架，用于快速原型设计和实验。

TensorFlow Model Garden：TensorFlow 官方提供的一系列预训练模型和模型架构。

Hugging Face Transformers：一个流行的开源库，提供了大量预训练模型和工具，用于 NLP 任务。

2、prefix LM 和 causal LM 区别是什么?

prefix LM (前缀语言模型)：在输入序列的开头添加一个可学习的任务相关的前缀，然后使用这个前缀

和输入序列一起生成输出。这种方法可以引导模型生成适应特定任务的输出。

causal LM (因果语言模型)：也称为自回归语言模型，它根据之前生成的 token 预测下一个 token。在

生成文本时，模型只能根据已经生成的部分生成后续部分，不能访问未来的信息。

3、涌现能力是啥原因?

涌现能力 (Emergent Ability) 是指模型在训练过程中突然表现出的新的、之前未曾预料到的能力。这种现象通常发生在大型模型中，原因是大型模型具有更高的表示能力和更多的参数，可以更好地捕捉数据中的模式和关联。

随着模型规模的增加，它们能够自动学习到更复杂、更抽象的概念和规律，从而展现出涌现能力。

4、大模型 LLM 的架构介绍?

大模型 LLM(Large Language Models) 通常采用基于 Transformer 的架构。Transformer 模型由多个编码器或解码器层组成，每个层包含多头自注意力机制和前馈神经网络。这些层可以并行处理输入序列中的所有位置，捕获长距离依赖关系。大模型通常具有数十亿甚至数千亿个参数，可以处理大量的文本数据，并在各种 NLP 任务中表现出色。

前馈神经网络 (Feedforward Neural Network) 是一种最基础的神经网络类型，它的信息流动是单向的，从输入层经过一个或多个隐藏层，最终到达输出层。在前馈神经网络中，神经元之间的连接不会形成闭环，这意味着信号在前向传播过程中不会回溯。前馈神经网络的基本组成单元是神经元，每个神经元都会对输入信号进行加权求和，然后通过一个激活函数产生输出。激活函数通常是非线性的，它决定了神经元的输出是否应该被激活，从而允许网络学习复杂和非线性的函数。

前馈神经网络在模式识别、函数逼近、分类、回归等多个领域都有应用。例如，在图像识别任务中，网络的输入层节点可能对应于图像的像素值，而输出层节点可能代表不同类别的概率分布。

训练前馈神经网络通常涉及反向传播 (Backpropagation) 算法，这是一种有效的学习算法，通过计算输出层的误差，并将这些误差信号沿网络反向传播，以调整连接权重。通过多次迭代这个过程，网络可以逐渐学习如何减少输出误差，从而实现对输入数据的正确分类或回归。

在设计和训练前馈神经网络时，需要考虑多个因素，包括网络的层数、每层的神经元数目、激活函数的选择、学习速率、正则化策略等，这些都对网络的性能有重要影响。

5、目前比较受欢迎的开源大模型有哪些?

GPT 系列：由 OpenAI 开发的生成式预训练模型，如 GPT-3。

BERT 系列：由 Google 开发的转换式预训练模型，如 BERT、RoBERTa 等。

T5 系列：由 Google 开发的基于 Transformer 的编码器-解码器模型，如 T5、mT5 等。

6、目前大模型模型结构都有哪些?

Transformer：基于自注意力机制的模型，包括编码器、解码器和编码器-解码器结构。

GPT 系列：基于自注意力机制的生成式预训练模型，采用解码器结构。

BERT 系列：基于自注意力机制的转换式预训练模型，采用编码器结构。

T5 系列：基于 Transformer 的编码器-解码器模型。

7、prefix LM 和 causal LM、encoder-decoder 区别及各自有什么优缺点?

prefix LM：通过在输入序列前添加可学习的任务相关前缀，引导模型生成适应特定任务的输 出。优点是可以减少对预训练模型参数的修改，降低过拟合风险；缺点是可能受到前缀表示长度的限制，无法充分捕捉任务相关的信息。

causal LM：根据之前生成的 token 预测下一个 token，可以生成连贯的文本。优点是可以生成灵 活的文本，适应各种生成任务；缺点是无法访问未来的信息，可能生成不一致或有误的内容。

encoder-decoder：由编码器和解码器组成，编码器将输入序列编码为固定长度的向量，解码器 根据编码器的输出生成输出序列。优点是可以处理输入和输出序列不同长度的任务，如机器翻译；缺点是模型结构较为复杂，训练和推理计算量较大。

8、模型幻觉是什么?业内解决方案是什么?

模型幻觉是指模型在生成文本时产生的不准确、无关或虚构的信息。这通常发生在模型在缺乏足够信息的情况下进行推理或生成时。业内的解决方案包括：

使用更多的数据和更高质量的训练数据来提高模型的泛化和准确性。

引入外部知识源，如知识库或事实检查工具，以提供额外的信息和支持。

强化模型的推理能力和逻辑推理，使其能够更好地处理复杂问题和避免幻觉。

9、大模型的 Tokenizer 的实现方法及原理?

大模型的 Tokenizer 通常使用字节对编码 (Byte-Pair Encoding,BPE) 算法。BPE 算法通过迭代地将最频繁出现的字节对合并成新的符号，来构建一个词汇表。在训练过程中，模型会学习这些符号的嵌入表示。Tokenizer 将输入文本分割成符号序列，然后将其转换为模型可以处理的数字表示。

这种方法可以有效地处理大量文本数据，并减少词汇表的规模。

10、ChatGLM3 的词表实现方法?

ChatGLM3 使用了一种改进的词表实现方法。它首先使用字节对编码 (BPE) 算法构建一个基本的词表，然后在训练过程中通过不断更新词表来引入新的词汇。具体来说，ChatGLM3 在训练 过程中会根据输入数据动态地合并出

现频率较高的字节对，从而形成新的词汇。这样可以有效地处理大量文本数据，并减少词汇表的规模。

同时，ChatGLM3 还使用了一种特殊的词表分割方法，将词表分为多个片段，并在训练过程中逐步更新这些片段，以提高模型的泛化能力和适应性。

11、GPT3、LLAMA、ChatGLM 的 Layer Normalization 的区别是什么?各自的优缺点是什么?

GPT3:采用了 Post-Layer Normalization (后标准化)的结构，即先进行自注意力或前馈神经网络的计算，然后进行 Layer Normalization。这种结构有助于稳定训练过程，提高模型性能。

LLAMA：采用了 Pre-Layer Normalization (前标准化)的结构，即先进行 Layer Normalization,然后进行自注意力或前馈神经网络的计算。这种结构有助于提高模型的泛化能力和鲁棒性。

ChatGLM：采用了 Post-Layer Normalization 的结构，类似于 GPT3。这种结构可以提高模型的性能和稳定性。

12、大模型常用的激活函数有哪些?

ReLU（Rectified Linear Unit）：一种简单的激活函数，可以解决梯度消失问题，加快训练速度。

GeLU（Gaussian Error Linear Unit）：一种改进的 ReLU 函数，可以提供更好的性能和泛化能力。

Swish：一种自门控激活函数，可以提供非线性变换，并具有平滑和非单调的特性。

13、多查询注意力与群查询注意力是否了解?区别是什么?

Multi-query Attention 和 Grouped-query Attention 是两种不同的注意力机制变种，用于改进和扩展传统的自注意力机制。Multi-query Attention：在 Multi-query Attention 中，每个查询可以与多个键值对进行交互，从而 捕捉更多的上下文信息。这种机制可以提高模型的表达能力和性能，特别是在处理长序列或复杂关系时。

Grouped-query Attention：在 Grouped-query Attention 中，查询被分成多个组，每个组内的查询与对应的键值对进行交互。这种机制可以减少计算复杂度，提高效率，同时仍然保持较好的性能。

14、多模态大模型是否有接触?落地案例?

多模态大模型是指可以处理和理解多种模态数据(如文本、图像、声音等）的模型。落地案例，例如：

OpenAI 的 DALL-E 和 GPT-3：DALL-E 是一个可以生成图像的模型，而 GPT-3 可以处理和理解文本。两者结合可以实现基于文本描述生成图像的功能。

Google 的 Multimodal Transformer：这是一个可以同时处理文本和图像的模型，用于各种多模态任务，如图像字幕生成、视觉问答等。

二、进阶篇

1、llama 输入句子长度理论上可以无限长吗?

LLaMA（Large Language Model Adaptation）模型的输入句子长度受到硬件资源和模型设计的限制。

理论上，如果硬件资源足够，模型可以处理非常长的输入句子。然而，实际上，由于内存和处理能力的限制，输入句子长度通常是有限制的。在实际应用中，开发者会根据具体需求和硬件配置来确定合适的输入句子长度。

2、什么是 LLMs 复读机问题?

LLMs 复读机问题是指在某些情况下，大型语言模型在生成文本时会重复之前已经生成的内容，导致生成的文本缺乏多样性和创造性。

3、为什么会出现 LLMs 复读机问题?

LLMs 复读机问题可能由多种因素引起，包括模型训练数据中的重复模式、模型在处理长序列时的注意力机制失效、或者模型在生成文本时对过去信息的过度依赖等。

4、如何缓解 LLMs 复读机问题?

数据增强：通过增加训练数据的多样性和复杂性，减少重复模式的出现。

模型改进：改进模型的结构和注意力机制，使其更好地处理长序列和避免过度依赖过去信息。

生成策略：在生成文本时采用多样化的策略，如抽样生成或引入随机性，以增加生成文本的多样性。

5、什么情况用 Bert 模型，什么情况用 LLaMA、ChatGLM 类大模型?

BERT 模型通常用于需要理解文本深层语义的任务，如文本分类、命名实体识别等。

LLaMA 和 ChatGLM 类大模型则适用于需要生成文本或进行更复杂语言理解的任务，如对话系统、文本生成等。选择哪种模型取决于任务的需求和可用资源。

6、各个专业领域是否需要各自的大模型来服务?

不同的专业领域需要特定的大模型来更好地服务。专业领域的大模型可以针对特定领域的语言 和知识进行优化，提供更准确和相关的回答和生成文本。

7、如何让大模型处理更长的文本?

使用模型架构，如 Transformer，它可以有效地处理长序列。

使用内存机制，如外部记忆或缓存，来存储和检索长文本中的信息。

使用分块方法，将长文本分割成更小的部分，然后分别处理这些部分。

大模型参数微调、训练、推理

8、如果想要在某个模型基础上做全参数微调，究竟需要多少显存?

全参数微调 (Full Fine-Tuning) 通常需要大量的显存，因为这种方法涉及到更新模型的所有参数。

显存的需求取决于模型的规模、批量大小、以及使用的硬件。例如，对于大型模型如 GPT- 3,可能需要多个 GPU 甚至 TPU 来分配显存，每个 GPU 或 TPU 可能需要几十 GB 的显存。在实际操作中，需要进行试错法来确定合适的批量大小和硬件配置。

9、为什么 SFT 之后感觉 LLM 傻了?

SFT（Supervised Fine-Tuning）之后感觉 LLM（Large Language Model）“傻了”，可能是因为微调过程中出现了以下问题：

过拟合：模型可能过度适应训练数据，导致在新数据上的泛化能力下降。

据质量：如果训练数据质量不高，模型可能学到了错误的模式或偏见。

微调强度：微调的强度可能不够，导致模型没有充分适应新的任务。在这种情况下，模型可能没有学习到足够的特定领域的知识，因此在执行相关任务时表现不佳。

10、SFT 指令微调数据如何构建?

收集或生成与特定任务相关的指令和数据对，其中指令是描述任务或要求的文本，数据是对应的输入输出示例。

清洗和预处理数据，以确保数据的质量和一致性。

根据任务需求，对数据进行增强，如使用数据增强技术生成更多的训练样本。

将数据格式化为模型训练所需的格式，例如，对于语言模型，通常需要将文本转化为模型可以理解的数字编码。

11、领域模型 Continue PreTrain 数据选取?

领域模型继续预训练（Continue Pre-Training）的数据选取应该基于领域内的文本特点和应用需求。通常，需要选取大量、高质量、多样化的领域文本数据。数据可以来自专业文献、行业报告、在线论坛、新闻文章等。数据选取时应该注意避免偏见和不平衡，确保数据能够全面地代表领域内的知识和语言使用。

12、领域数据训练后，通用能力往往会有所下降，如何缓解模型遗忘通用能力?

多任务学习:在训练过程中同时包含领域内和通用的任务，使模型能够同时学习领域特定的和通用的知识。

控制微调强度：通过调整微调的学习率或训练轮数来控制模型对领域数据的适应程度。

定期回炉：在领域数据训练后，定期使用通用数据进行回炉训练，以保持模型的通用能力。

知识蒸馏：使用一个预训练的通用模型来指导领域模型，帮助模型保持通用知识。

13、领域模型 Continue PreTrain，如何让模型在预训练过程中就学习到更多的知识?

数据增强：使用数据增强技术如回译、掩码语言模型等来生成更多的训练样本。

知识注入：将领域特定的知识以文本、结构化数据或知识图谱的形式注入到预训练过程中。

多模态学习：如果适用，可以使用多模态数据(如文本和图像)进行预训练，以丰富模型的知识表示。

14、进行 SFT 操作的时候，基座模型选用 Chat 还是 Base?

在进行指令微调 (SFT) 操作时，选择基座模型 (Chat 或 Base) 取决于具体任务的需求和模型的性能。通常，如果任务需要生成对话或交互式响应，可以选择对话优化的模型 (Chat)。如果任务更注重理解和生成文本的能力，可以选择基础模型 (Base)。

在实际应用中，可能需要根据实验结果和模型性能来选择最合适的基座模型。

15、领域模型微调指令&数据输入格式要求?

领域模型微调的指令和数据输入格式要求取决于所使用的模型和框架。一般来说，指令应该是清晰、具体的，能够指导模型完成特定的任务。数据输入格式通常需要与模型的输入接口相匹配，例如，对于文本模型，数据通常需要是字符串格式，并且可能需要经过特定的预处理，如分词、编码等。

16、领域模型微调领域评测集构建?

构建领域模型微调的领域评测集时，应该确保评测集能够全面、准确地反映领域内的任务需求和性能指标。通常，

需要从领域内的真实数据中收集或生成评测样本，并确保样本的多样性和代表性。此外，可以根据任务需求设计定制的评价指标，以评估模型在领域内的性能。

17、领域模型词表扩增是不是有必要的?

领域模型词表扩增通常是有必要的，尤其是当领域内有大量的专业术语或特定词汇时。词表扩 增可以帮助模型更好地理解和生成领域内的文本，提高模型的领域适应性。然而，词表扩增也需要谨慎进行，以避免引入过多的噪音或不相关的词汇。

18、如何训练自己的大模型?

1.选择合适的预训练目标和任务：确定模型将学习哪些通用的语言知识，以及针对哪些特定任务进行优化。

2.收集和准备数据：收集大量、多样化的数据，包括通用数据和特定领域的数据，进行清洗和预处理。

3.选择模型架构：选择一个适合的模型架构，如 Transformer，并确定模型的规模和层数。

4.定义训练流程：设置训练参数，如学习率、批量大小、训练轮数等，并选择合适的优化器和损失函数。

5.训练模型：使用准备好的数据和训练流程开始训练模型，监控训练过程中的性能和资源使用。

6.评估和调优：在训练过程中定期评估模型的性能，并根据需要调整训练参数和模型架构。

7.微调和优化：在模型达到一定的性能后，进行微调以适应特定的应用场景和任务需求。

19、训练中文大模型有啥经验?

使用大量高质量的中文数据，包括文本、对话、新闻、社交媒体帖子等。

考虑语言的特点，如词序、语法结构、多义性等，并设计相应的预训练任务。

使用适合中文的语言模型架构，如 BERT 或 GPT，并进行适当的调整以优化性能。

考虑中文的特殊字符和标点，确保模型能够正确处理这些字符。

进行多任务学习，同时训练多个相关任务，以提高模型的泛化能力。

20、指令微调的好处?

提高模型在特定任务上的性能，使其能够更好地理解和执行指令。

通过指令和示例数据的结合，使模型能够学习到更具体、更实用的知识。

减少了模型对大规模标注数据的依赖，通过少量的指令和示例数据就能进行有效的微调。

可以通过不同的指令和示例数据组合，快速适应不同的任务和应用场景。

21、预训练和微调哪个阶段注入知识的?

在预训练阶段，模型通过大量的无监督数据学习通用的语言知识和模式。在微调阶段，模型通过与特定任务相关的监督数据学习特定领域的知识和任务特定的模式。因此，知识注入主要发生在微调阶段。

22、想让模型学习某领域或行业知识，是应该预训练还是应该微调?

为了让模型学习某个领域或行业的知识，通常建议先进行预训练，以学习通用的语言知识和模式。预训练可以帮助模型建立强大的语言表示，并提高模型的泛化能力。

然后，可以通过微调来注入特定领域或行业的知识，使模型能够更好地适应特定的任务和应用场景。

23、多轮对话任务如何微调模型?

收集多轮对话数据，包括用户查询、系统回复、以及可能的中间交互。对数据进行预处理，如分词、编码等，使其适合模型输入格式。

设计多轮对话的微调目标，如序列到序列学习、生成式对话等。

微调模型，使其能够生成连贯、自然的对话回复，并考虑到对话上下文和用户意图。

24、微调后的模型出现能力劣化，灾难性遗忘是怎么回事?

微调后的模型出现能力劣化，灾难性遗忘可能是因为模型在微调过程中学习到了过多的特定任 务的知识，而忽略了通用的语言知识。这可能导致模型在训练数据上表现良好，但在未见过的数据上表现不佳。

为了解决这个问题，可以采取一些措施，如多任务学习、控制微调强度、定期使用通用数据进行回炉训练等。

25、微调模型需要多大显存?

微调模型需要的显存取决于模型的规模、任务复杂度、数据量等因素。一般来说，微调模型需 要的显存通常比预训练模型少，因为微调涉及到更新的参数较少。然而，具体需要的显存仍然需要根据实际情况进行评估和调整。

26、大模型 LLM 进行 SFT 操作的时候在学习什么?

特定领域的语言模式和知识，包括专业术语、行业特定用语等。

针对特定任务的生成策略和响应模式。

对话上下文中的连贯性和逻辑性，对于多轮对话任务尤其重要。

指令理解和执行能力，使模型能够更准确地理解和执行用户的指令。

27、预训练和 SFT 操作有什么不同?

预训练和 SFT 操作的主要区别在于目标和数据集。预训练通常是在大规模的无标签数据集上进行的，目的是让模型学习到通用的语言表示和模式。这个过程不需要人工标注数据，而是通过模型自己从数据中学习。

SFT 则是在有标签的数据集上进行的，目的是让模型适应特定的任务或领域。这个过程需要人工标注数据，以确保模型能够学习到正确的任务特定的模式和知识。

28、样本量规模增大，训练出现 OOM 报错，怎么解决?

当样本量规模增大时，训练出现 OOM (Out of Memory) 错误可能是由于显存不足导致的。为了解决这个问题，可以尝试以下方法：

增加训练设备的显存，如使用更高性能的 GPU 或增加 GPU 数量。

调整批量大小，减少每次训练时处理的样本数量。

使用模型并行或数据并行技术，将模型或数据分片到多个设备上进行训练。

使用动态批处理，根据可用显存动态调整批量大小。

29、大模型 LLM 进行 SFT 如何对样本进行优化?

数据增强：通过对原始数据进行转换，如文本回译、添加噪声等，生成更多的训练样本。样本选择：选择与特定任务最相关的样本进行训练，以提高训练效率和性能。

样本权重：根据样本的难易程度或重要性为样本分配不同的权重，以优化训练过程。

平衡采样：在训练过程中，确保每个类别或子任务都有足够的样本被训练到。

30、模型参数迭代实验步骤?

模型参数迭代实验是指在训练过程中，对模型的参数进行迭代调整和优化，以提高模型的性能。这通常涉及以下步骤：

选择一组初始参数。

在训练过程中，定期评估模型的性能。

根据评估结果，调整模型的参数，如学习率、批量大小、正则化参数等。

重复评估和调整参数，直到模型的性能达到预期的目标。

31、为什么需要进行参选微调?参数微调的原因有哪些?

参数微调是指只对模型的一部分参数进行更新，以适应特定的任务或领域。进行参数微调的原因包括：

提高计算效率：参数微调通常比全量微调需要更少的计算资源，因为只有部分参数需要更新。

减少过拟合风险：只更新与特定任务相关的参数，可以减少模型对训练数据的过度依赖，降低过拟合的风险。

提高泛化能力：参数微调可以使模型在保持通用语言能力的同时，适应特定的任务需求。

32、模型参数微调的方式有那些?你最常用哪些方法?

权重共享：在模型中，将部分参数设置为共享，这些参数同时用于多个任务或领域。

参数掩码：在模型中，将部分参数设置为不可训练，这些参数保持预训练时的值不变。

参数分解：将大型的参数矩阵分解为多个小型矩阵，只更新其中的部分矩阵。

参数共享微调：在模型中，将部分参数设置为共享，这些参数用于多个相关任务。

33、prompt tuning 和 prefix tuning 在微调上的区别是什么?

Prompt Tuning 和 Prefix Tuning 都是参数高效的微调方法，它们通过在模型输入中添加特定的提示或前缀来引导模型生成适应特定任务的输出。区别在于：

Prompt Tuning：在输入序列的末尾添加可学习的提示，提示可以是几个单词或短语，用于指导模型生成特定的输出。

Prefix Tuning：在输入序列的开头添加可学习的连续前缀表示，前缀表示包含了任务特定的信息，用于引导模型生成适应特定任务的输出。

34、LLaMA-adapter 如何实现稳定训练?

LLaMA-adapter 是一种参数高效的微调方法，它通过在预训练模型的每个 Transformer 层中添加小型适配器模块来实现特定任务的适应。为了实现稳定训练，可以采取以下措施：适配器初始化：使用预训练模型的参数作为适配器模块的初始化，以保持模型的稳定性。

适配器正则化：使用正则化技术，如权重衰减或 dropout，来减少适配器模块的过拟合风险。

逐步学习：逐步调整适配器模块的参数，避免参数更新的幅度过大。

适配器优化：选择合适的优化器和训练策略，如使用较小的学习率、较长的训练周期等，以实现稳定的训练过程。

35、LoRA 原理与使用技巧有那些?

LoRA（Low-Rank Adaptation）是一种参数高效的微调方法，它通过引入低秩分解来减少需要更 新的参数数量。LoRA 的工作原理是将预训练模型的注意力矩阵或前馈网络矩阵分解为两个低秩 矩阵的乘积，其中这两个低秩矩阵被视为可学习的任务特定参数。

使用 LoRA 的技巧包括：

适配器初始化：使用预训练模型的参数作为 LoRA 适配器模块的初始化，以保持模型的稳定性。

低秩分解：选择合适的低秩分解方法，如奇异值分解 (SVD) 或随机矩阵分解，以实现低秩分解。

逐步学习：逐步调整 LoRA 适配器模块的参数，避免参数更新的幅度过大。

适配器正则化：使用正则化技术，如权重衰减或 dropout，来减少 LoRA 适配器模块的过拟合风险。

35、LoRA 微调优点是什么?

参数高效：LoRA 只更新少量的低秩矩阵，相比全量微调，可以显著减少需要更新的参数数量。

计算效率：由于只更新少量的低秩矩阵，LoRA 可以减少计算资源的需求，提高训练和推理的效率。

模型稳定性：LoRA 适配器模块可以保持预训练模型的稳定性，减少过拟合风险。

性能提升：LoRA 微调可以在不牺牲太多性能的情况下实现参数高效的微调。

36、AdaLoRA 的思路是怎么样的?

AdaLoRA 是一种自适应的 LoRA 方法，它可以根据任务的需求和模型的性能动态调整 LoRA 适配器模块的参数。AdaLoRA 的思路是：

初始化 LoRA 适配器模块的参数，使用预训练模型的参数作为初始化。

在训练过程中，根据模型的性能和任务需求，动态调整 LoRA 适配器模块的参数。

通过调整 LoRA 适配器模块的参数，使模型能够更好地适应特定的任务需求。

37、LoRA 权重合入 chatglm 模型的方法?

在 chatGLM 模型的每个 Transformer 层中添加 LoRA 适配器模块。

使用预训练模型的参数作为 LoRA 适配器模块的初始化。

在训练过程中，更新 LoRA 适配器模块的参数，以适应特定的任务需求。

保持预训练模型的参数不变，避免对预训练模型产生负面影响。

38、P-tuning 讲一下?与 P-tuning v2 区别在哪里?优点与缺点?

P-tuning 是一种参数高效的微调方法，它通过在模型输入中添加可学习的连续前缀来引导模型生 成适应特定任务的输出。P-tuning v2 是 P-tuning 的改进版本，它使用了更多的连续前缀表示来引导模型生成适应特定任务的输出。

P-tuning 与 P-tuning v2 的区别在于：

P-tuning：在输入序列的开头添加一个可学习的连续前缀，前缀的长度较短。

P-tuning v2：在输入序列的开头添加多个可学习的连续前缀，前缀的长度较长。

P-tuning 的优点是参数高效，计算资源需求较低，可以快速实现模型微调。P-tuning 的缺点是可 能受到前缀表示长度的限制，无法充分捕捉任务相关的信息。P-tuning v2 通过使用更多的连续前缀，可以更充分地捕捉任务相关的信息，但可能需要更多的计算资源来更新多个前缀的参数。

38、预训练和 SFT 操作有什么不同?

预训练和 SFT 操作的主要区别在于目标和数据集。预训练通常是在大规模的无标签数据集上进行的，目的是让模型

学习到通用的语言表示和模式。这个过程不需要人工标注数据，而是通过模型自己从数据中学习。

SFT 则是在有标签的数据集上进行的，目的是让模型适应特定的任务或领域。这个过程需要人工标注数据，以确保模型能够学习到正确的任务特定的模式和知识。

39、训练一个通用大模型的流程有那些?

数据收集：收集大量的、多样化的、无标签的文本数据。

数据预处理：对收集的数据进行清洗、分词、编码等预处理步骤。

模型设计：选择合适的模型架构，如 Transformer,并确定模型的规模和层数。预训练目标：设计预训练任务，如语言建模、掩码语言模型、句子对齐等。

训练模型:使用预训练数据集和预训练目标开始训练模型。

评估性能：在预训练过程中定期评估模型的性能，并根据需要调整训练参数。

微调和优化：在预训练完成后，使用有标签的数据集进行微调，以适应特定的任务或领域。

40、DDO 与 DPO 的区别是什么?

DDO（Dual Data Objectives）和 DPO（Dual Prompt Objectives）是两种不同的训练策略，用于提高大型语言模型的性能。

DDO: 在训练过程中，同时优化两个数据集的目标， 一个是通用数据集，另一个是特定领域数据集。这样可以让模型同时学习通用知识和特定领域的知识，提高模型的泛化能力和领域适应性。

DPO: 在训练过程中，同时使用两个提示 (prompt)，一个是通用提示，另一个是特定领域提示。这样可以让模型在执行任务时，同时利用通用知识和特定领域的知识，提高模型在特定任务上的性能。

41、是否接触过 embeding 模型的微调方法?

嵌入模型微调通常涉及调整模型中的嵌入层，以适应特定的任务或领域。这可能包括：初始化：使用特定领域的数据来初始化嵌入层，以便更好地捕捉领域特定的信息。

调整：通过训练或优化嵌入层的参数，使其能够适应特定任务或领域的需求。

知识注入：将领域特定的知识以向量的形式注入到嵌入层中，以增强模型对领域知识的理解和应用。

42、有哪些省内存的大语言模型训练/微调/推理方法?

模型剪枝：通过移除模型中的冗余结构和参数，减少模型的内存占用。

知识蒸馏：使用一个大型教师模型来指导一个小型学生模型，使学生模型能够学习到教师模型的知识，同时减少内存占用。

量化：将模型的权重和激活从浮点数转换为低精度整数，减少模型的内存占用和计算需求。

模型并行:将大型模型分割到多个设备上进行训练和推理，减少单个设备的内存需求。

数据并行：将训练数据分割到多个设备上，每个设备训练模型的一个副本，减少单个设备的内存需求。

动态批处理：根据可用内存动态调整批量大小，以适应内存限制。

43、大模型 (LLMs) 评测有那些方法?如何衡量大模型的效果?

大模型 (LLMs) 的评测方法通常包括:

准确性: 评估模型在特定任务上的预测准确性。

泛化能力: 评估模型在未见过的数据上的表现。

计算效率: 评估模型训练和推理的速度和资源需求。

安全性: 评估模型在对抗性输入下的稳定性和鲁棒性。

多样性和创造性: 评估模型生成文本的多样性和创造性。

人类评估: 通过人工评估来衡量模型的性能，特别是在对话和生成任务中。

衡量大模型效果的方法包括:

自动评估指标: 使用如 BLEU 、ROUGE 、METEOR 等自动评估指标来衡量模型的语言生成和理解能力。

任务特定的指标: 使用任务特定的指标来衡量模型在特定任务上的性能，如准确率、F1 分数等。

用户反馈: 收集用户对模型生成内容的反馈，以评估模型的实际应用效果。

44、如何解决三个阶段的训练（SFT->RM->PPO）过程较长，更新迭代较慢问题?

减少训练数据量：如果训练数据量过大，可以考虑减少数据量，以加快训练速度。

优化训练流程：优化训练流程，如使用更高效的训练算法、调整训练参数等，以加快训练速度。

并行训练：使用多 GPU 或多服务器并行训练模型，以加快训练速度。

提前停止：在训练过程中，如果模型性能不再提高，可以提前停止训练，以节省时间。

知识蒸馏：使用一个大型教师模型来指导一个小型学生模型，使学生模型能够快速学习到教师模型的知识。

45、模型训练的数据集问题：一般数据集哪里找?

公开数据集：许多研究机构和组织会发布公开数据集，如 IMDb 、Wikipedia 、Common Crawl 等。

特定领域数据集：针对特定领域的数据集，如医疗、金融、法律等，通常需要从相关的专业文献、报告、论坛等渠道获取。

合成数据：通过自动化或半自动化方法生成数据，如文本合成、数据增强等。

用户生成数据：通过众包、调查、游戏等方式收集用户生成的数据。

商业数据：从商业公司或服务中获取数据，通常需要遵守相关的数据使用协议和隐私政策。

46、为什么需要进行模型量化及原理?

模型量化是将模型中的权重和激活从高精度浮点数转换为低精度整数(如 INT8、INT4、FP16 等)的过程，目的是减少模型的大小、提高计算效率并降低内存需求。

模型量化的原理在于，低精度数值格式可以提供足够的精度来保持模型性能，同时显著减少数值的位数，从而减少存储和计算资源的使用。

47、大模型词表扩充的方法及工具?

大模型词表扩充的方法包括：

新增词汇：手动添加领域特定的术语和词汇到词表中。

数据驱动：通过分析大量文本数据自动识别和添加高频出现的词汇。

词汇映射：将特定领域的词汇映射到现有的词表中，或者创建新的词汇条目。

工具方面， 一些流行的词表管理工具和库包括：

Hugging Face Transformers：提供了一个预训练模型和词表管理的接口。

SentencePiece：一个用于构建词汇表的工具，支持 BPE 和其他子词分割方法。

Moses：一个开源的自然语言处理工具，包括用于词表构建和分词的工具。

48、大模型应用框架及其功能?

大模型应用框架提供了一组工具和库，用于构建、训练和部署大型语言模型。这些框架通常包括以下功能：

模型加载和保存：支持加载预训练模型和保存微调后的模型。

数据处理：提供数据预处理、分词、编码等工具。

模型训练：支持模型训练、评估和调试。

模型部署：支持将模型部署到不同的环境和平台，如服务器、移动设备等。

API 接口：提供模型预测的 API 接口，方便集成到其他应用中。

一些流行的大模型应用框架包括：

Hugging Face Transformers：一个流行的 NLP 研究工具，提供了大量预训练模型和工具。

PyTorch：一个开源的深度学习框架，支持大型语言模型的训练和部署。

TensorFlow：另一个流行的深度学习框架，也支持大型语言模型的训练和部署。

49、搭建大模型应用遇到过那些问题？如何解决的？

搭建大模型应用时可能会遇到以下问题：

资源限制：计算资源不足，如显存不足、计算时间受限等。

模型稳定性：模型在训练或部署过程中出现不稳定的行为。

数据质量：训练数据质量不高，导致模型性能不佳。

模型部署：将模型部署到生产环境中的技术挑战。

解决这些问题的方法可能包括：

资源优化：使用更高效的训练算法、调整训练参数、使用模型并行或数据并行技术。模型调试：使用

调试工具和技术来分析模型行为，找出问题的根源。

数据处理：进行数据清洗、增强和预处理，以提高数据质量。

部署策略：选择合适的部署策略，如使用模型压缩技术、优化模型结构等。

50、如何提升大模型的检索效果?

优化索引：使用更高效的索引结构，如倒排索引、BM25等。

特征工程：提取和利用有效的特征，如文本向量、词频等。

模型选择：选择合适的检索模型，如基于向量的相似度计算、基于排序的模型等。

训练策略：使用训练策略，如多任务学习、知识蒸馏等，来提高模型的性能。

评估指标：使用更准确的评估指标，如 MAP、NDCG 等，来衡量检索效果。

51、是否了解上下文压缩方法?

上下文压缩是一种减少模型参数数量和计算复杂度的技术，同时尽量保持模型的性能。这种方法通常涉及：

模型剪枝：移除模型中的冗余结构和参数。

知识蒸馏：使用一个大型教师模型来指导一个小型学生模型，使学生模型能够学习到教师模型的知识。

权重共享：在模型中，将部分参数设置为共享，这些参数同时用于多个任务或领域。

低秩分解：将大型参数矩阵分解为多个小型矩阵，只更新其中的部分矩阵。

52、如何实现窗口上下文检索?

窗口上下文检索是一种在给定文本片段的上下文中检索相关信息的方法。实现窗口上下文检索通常涉及以下步骤：

文本分块：将长文本分割成多个较小的文本块，这些文本块被称为窗口。

索引构建：为每个文本块构建索引，以便快速检索相关信息。

查询处理：将查询文本与索引中的文本块进行匹配，找到与查询最相关的文本块。

上下文检索：在找到的相关文本块中，检索与查询相关的信息。这可能涉及到计算文本块与查询的相似度，并根据相似度排序文本块。结果生成：根据检索结果生成答案或摘要。

53、开源的 RAG 框架有哪些，你比较了解?

RAG(Retrieval-Augmented Generation) 是一种结合了检索和生成的框架，用于提高大型语言模型生成文本的质量和相关性。开源的 RAG 框架包括：

Hugging Face's RAG:一个结合了检索增强生成的开源框架，支持多种任务，如文本生成、摘要等。

Google's Retrieval-Augmented Generator(RAG)TensorFlow 实现：一个基于 TensorFlow 的 RAG 实现，用于支持大规模的文本生成任务。

Microsoft's RAG: 一个结合了检索和生成的框架，用于支持多轮对话和知识密集型任务。

54、大模型应用框架 LangChain 和 LlamaIndex 各自的优势有那些?

LangChain 和 LlamaIndex 是大模型应用框架，它们提供了构建、训练和部署大型语言模型的工具和库。这些框架的优势包括：

易用性：提供了一组易于使用的工具和库，简化了大模型应用的开发和部署过程。

灵活性：支持多种模型架构和任务，能够适应不同的应用场景和需求。

高效性：提供了高效的训练和推理算法，减少了计算资源的需求。

集成性：与其他工具和框架具有良好的集成，如数据处理、模型评估等。

社区支持：拥有活跃的社区，提供了大量的教程、文档和讨论，帮助用户解决问题和提高技能。

55、向量库有那些?各自优点与区别?

TensorFlow: 一个开源的深度学习框架，提供了向量操作和计算的支持。

PyTorch: 另一个流行的深度学习框架，也提供了向量操作和计算的支持。

NumPy: 一个用于数值计算的 Python 库，提供了向量操作和矩阵运算的支持。

SciPy: 基于 NumPy 的 Python 库，提供了用于科学计算的向量操作和函数。

这些向量库的优点包括:

高效性: 提供了高效的向量操作和矩阵运算，能够快速处理大规模数据。

灵活性: 支持多种数据类型和操作，能够适应不同的应用场景和需求。

社区支持: 拥有活跃的社区，提供了大量的教程、文档和讨论，帮助用户解决问题和提高技能。

区别在于它们的设计哲学、API 接口和使用场景。例如，TensorFlow 和 PyTorch 都是深度学习框架，提

供了全面的神经网络构建和训练功能，而 NumPy 和 SciPy 更专注于数值计算和科学计算。

56、向量数据库有那些?各自优点与区别?

向量数据库是一种数据库，专门设计用于存储和查询向量数据，常用于机器学习和数据科学领域。向量数据库可以高效地处理高维空间数据的相似性搜索,这在图像识别、文本搜索、推荐系统等应用中非常重要。以下是一些流行的向量数据库及其优缺点:

Milvus

优点: Milvus 是一个开源的向量数据库，支持多种类型的向量索引，如 IVF、HNSW、Flat 等。它提供了可扩展的架构，可以处理大量数据，并支持云原生部署。

缺点: 由于是较新的项目，社区和文档可能不如一些老牌数据库成熟。

Faiss

优点: Faiss 是 由 FacebookAI 团队开发的高效相似性搜索和密集向量聚类库。它提供了多种向量索引算法，性能极高。

缺点：作为一个库而不是完整的数据库系统，Faiss 不提供完整的数据管理功能，需要用户自己集成到应用中。

Vespa

优点：Vespa 是由 Yahoo 开发的一个高性能分布式数据存储和查询系统，支持向量相似性搜索和实时数据摄入。

缺点：Vespa 的配置和使用相对复杂，可能需要较深的系统知识。

Pinecone

优点：Pinecone 是一个托管的向量数据库服务，易于设置和使用，提供了强大的相似性搜索功能。

缺点：作为一个商业服务，Pinecone 的成本可能比开源解决方案要高。

Weaviate

优点：Weaviate 是一个开源的向量搜索引擎，支持多种数据类型，包括文本、图像和向量，并提供了易于使用的 RESTAPI。

缺点：相对于其他一些解决方案，Weaviate 可能还不够成熟，社区较小。

57、使用外部知识数据库时需要对文档进行分块，如何科学的设置文档块的大小?

查询需求：根据查询的需求和上下文长度来确定文档块的大小。

检索效率：较小的文档块可以提高检索效率，但过小的块可能导致信息的碎片化。

存储和计算资源：考虑存储和计算资源的需求，确定文档块的大小以平衡效率和资源使用。

用户体验:确保文档块的大小适合用户的阅读和理解需求。

一种科学的方法是进行实验和评估，通过比较不同文档块大小对检索效果、效率和用户体验的影响，来确定最佳的分块大小。

58、LLMs 受到上下文长度的限制，如果检索到的文档带有太多噪声，该如何解决这样的问题?

上下文修剪：使用摘要或摘要生成技术来提取文档的关键部分，减少噪声。

知识蒸馏：使用一个大型教师模型来指导一个小型学生模型，使学生模型能够学习到教师模型的知识，从而提高模型的鲁棒性。

过滤和去噪：使用文本过滤和去噪技术，如文本清洗、去重、去除无关信息等，来减少噪声。

强化学习：通过强化学习训练模型，使其能够自动识别和忽略噪声信息，专注于相关和有用的信息。

数据增强：通过对原始数据进行转换，如文本回译(将文本翻译成另一种语言再翻译回来)、添加噪声等，生成更多的训练样本，从而提高模型对噪声的鲁棒性。

知识蒸馏是一种模型压缩技术，其中一个大型的、表现良好的模型(教师模型)被用来训练一 个小型的模型(学生模型)。这个过程涉及到将教师模型的知识转移到学生模型中，通常通过模仿教师模型的输出或中间层的表示。学生模型因此能够学习到如何处理噪声，同时保持较小的模型大小，这有助于在有限的上下文长度内工作。

59、RAG(检索增强生成)对于大模型来说，有什么好处?

提高生成质量：通过结合检索到的相关信息， RAG 可以帮助大型语言模型生成更准确、更相关和更高质量的文本。

增强上下文关联性：检索到的信息可以为模型提供更多的上下文信息，使生成的文本更加符合上下文语境。

提高模型鲁棒性：通过结合检索到的信息，模型可以更好地处理不完整或噪声的输入，提高模型的鲁棒性。

减少训练数据需求：RAG 可以通过检索相关信息来增强模型的知识，从而减少对大规模标注数据的依赖。

提高模型泛化能力：RAG 可以帮助模型学习到更广泛的知识，提高模型的泛化能力，使其能够更好地适应不同的任务和领域。

60、Self-attention 的公式及参数量？为什么用多头？为什么要除以根号 d?

Self-attention 模型在对当前位置的信息进行编码时，会过度的将注意力集中于自身的位置，因 此作者提出了通过多头注意力机制来解决这一问题。同时，使用多头注意力机制还能够给予注意力层的输出包含有不同子空间中的编码表示信息，从而增强模型的表达能力。

这是因为点积的数量级增长很大，因此将 softmax 函数推向了梯度极小的区域。

Self-attention (自注意力)机制是Transformer模型的核心组成部分，它允许模型在处理序列数据 时，为序列中的每个元素(如词或标记)分配不同的注意力权重，从而捕捉序列内的依赖关系。

Self-attention 的基本公式如下：

计算 Query（Q）、Key（K）和 Value（V）：

这些矩阵是通过将输入序列的嵌入（或隐藏状态）与三个不同的权重矩阵（Wq、Wk、Wv）相乘得到的。这三个权重矩阵是模型需要学习的参数。

Q=X*WqK*=*X*Wk

V=X*Wv

其中，X 是输入序列的嵌入矩阵，维度为，N 是序列长度，D 是嵌入维度。

计算注意力得分：

使用 Query 和 Key 计算注意力得分，这反映了序列中每个元素对其他元素的重要性。

得分=Q*K^T

应用 softmax 函数：

将得分通过 sofrmax 函数转换为概率分布，确保所有注意力权重的总和为 1。

概率分布=softmax(得分/ √D)

计算加权的 Value:

将 Value 与 softmax 得到的概率分布相乘，得到加权后的 Valuc,这是考虑了序列中其他元素的上下文信息的新表示。

加权 Value=概率分布*V

输出：

将加权 Value 相加，得到最终的输出，这是序列中每个元素的上下文表示。

输出=加权 Value 之和

参数量的计算：

每个权重矩阵 (Wq、Wk、Wv) 的参数量为，因此总共有 3 个权重矩阵，参数量为。

——

为什么用多头 (Multi-Head) 注意力：

多头注意力允许模型在不同的表示子空间中学习信息，这样可以让模型同时关注不同的信息维度。每个头学习到的信息可以独立地编码输入序列的不同方面，然后将这些信息综合起来，得到更丰富的表示。

为什么要除以根号 D:

将得分除以根号 D（得分归一化）可以防止内积过大导致 softmax 函数梯度变得非常小，这有助于数值稳定性，使得学习过程更加稳定。此外，它还可以看作是一种缩放因子，帮助模型在不同维度上保持一致的性能。

三、大模型 (LLMs)LangChain 什么是 LangChain?

LangChain 是一个用于构建和运行大型语言模型应用的开源框架。它提供了一套工具和组件，帮助开发者将大型语言模型（如 GPT-3）与其他工具和 API 结合，以完成更复杂的任务。

1、LangChain 包含哪些核心概念?

Components:可重用的模块，例如 API 调用、数据库查询等。

Chains:将多个 Components 链接在一起以完成特定任务的流程。

Prompt Templates: 用于指导语言模型生成输出的文本模板。

Output Parsers:解析语言模型输出的工具。

Indexes and Retrievers: 用于存储和检索信息的索引和数据检索器。

Agents and Toolkits:提供特定领域功能的代理和工具集。

2、什么是 LangChain Agent?

LangChain Agent 是一种可以执行一系列操作以完成复杂任务的程序。它可以根据给定的输入和上下文，选择合适的工具和策略来生成响应或执行操作。

3、如何使用 LangChain?

定义 Components: 创建或集成各种 API 和工具。

构建 Chains: 将 Components 组合成完成特定任务的流程。

设置 Prompt Templates：定义用于指导语言模型的文本模板。

配置 Output Parsers：解析和提取语言模型的输出。

部署和运行：将构建的应用部署到服务器或云平台，并进行测试和优化。

4、LangChain 支持哪些功能?

集成和调用外部 API。

查询和操作数据库。

文本生成和编辑。

信息检索和问答。

多步骤任务执行和决策。

5、什么是 LangChain model?

LangChain model 指的是在 LangChain 框架中使用的大型语言模型，如 GPT-3 或类似的模型。这些模型

通常用于生成文本、回答问题或执行特定的语言任务。

6、LangChain 包含哪些特点?

开源和可扩展：易于集成和扩展新功能。

模块化和可重用：Components 和 Chains 可以重用和组合。灵活和可定制：可以自定义 Prompt Templates 和 Output Parsers。

支持多种语言模型：可以集成和使用不同的语言模型。

7、LangChain 如何使用?

定义 Components: 创建或集成各种 API 和工具。

构建 Chains: 将 Components 组合成完成特定任务的流程。

设置 Prompt Templates:定义用于指导语言模型的文本模板。

配置 Output Parsers: 解析和提取语言模型的输出。

部署和运行：将构建的应用部署到服务器或云平台，并进行测试和优化。

8、LangChain 存在哪些问题及方法方案?

低效的令牌使用问题：可以通过优化 Prompt Templates 和减少不必要的 API 调用来解决。

文档的问题：可以通过改进文档和提供更多的示例来帮助开发者理解和使用 LangChain。

太多概念容易混淆：可以通过提供更清晰的解释和更直观的 API 设计来解决。

行为不一致并且隐藏细节问题：可以通过提供更一致和透明的 API 和行为来解决。

缺乏标准的可互操作数据类型问题：可以通过定义和使用标准的数据格式和协议来解决。

低效的令牌使用问题：

在语言模型应用中，令牌是模型处理文本的单位，通常与成本挂钩。如果 Prompt Templates 设计 不当或 API 调用频繁，可能会导致令牌的浪费，增加成本。

解决方案：优化 Prompt Templates，确保它们尽可能高效地传达信息，减少冗余。同时，减少不必要的

API 调用，例如通过批量处理数据或合并多个请求。

文档的问题：

如果 LangChain 的文档不清晰或不完整，开发者可能难以理解如何使用框架，或者可能无法充分利用其功能。

解决方案：改进文档的质量，提供详细的 API 参考、教程和最佳实践指南。增加更多的示例代码和应用场景，帮助开发者更快地上手。

太多概念容易混淆：

LangChain 可能引入了许多新的概念和抽象，对于新用户来说，这可能难以理解和区分。

解决方案：提供清晰的解释和定义，使用户能够理解每个概念的目的和作用。设计更直观的 API，使其易于理解和使用。

行为不一致并且隐藏细节问题：

如果 API 的行为不一致，开发者可能难以预测其结果，这会导致错误和混淆。隐藏细节可能会让开发者难以调试和优化他们的应用。

解决方案：确保 API 的行为一致，并提供清晰的错误消息和文档。避免隐藏太多细节，而是提供适当的抽象级别，同时允许高级用户访问底层实现。

缺乏标准的可互操作数据类型问题：

如果 LangChain 没有定义和使用标准的数据格式和协议，那么在不同的系统和服务之间进行数据交换可能会很困难。

解决方案：定义和使用标准的数据格式(如 JSON、CSV)和协议(如 REST、gRPC)，以确保不同组件和服务之间的互操作性。

9、LangChain 替代方案?

LangChain 的替代方案包括其他用于构建和运行大型语言模型应用的开源框架，例如 HuggingFace 的 Transformers 库、OpenAI 的 GPT-3AP1 等。

10、LangChain 中 Components and Chains 是什么?

Components 是可重用的模块，例如 API 调用、数据库查询等。Chains 是将多个 Components 链接在一起以完成特定任务的流程。

11、LangChain 中 Prompt Templates and Values 是什么?

Prompt Templates 是用于指导语言模型生成输出的文本模板。Values 是填充 Prompt Templates 中的变量的实际值。

12、LangChain 中 Example Selectors 是什么?

Example Selectors 是从一组示例中选择一个或多个示例的工具。它们可以用于提供上下文或示例，以帮助语言模型生成更准确的输出。

上下文关联：当模型需要根据特定的上下文或场景生成回答时，Example Selectors 可以帮助选择与当前上下文最相关的示例。

数据过滤：在处理大量数据时， Example Selectors 可以根据特定的标准和条件过滤数据，以便模型仅处理最相关的信息。

个性化回答：Example Selectors 可以根据用户的需求和偏好选择示例，从而生成更加个性化的回答。

13、LangChain 中 Output Parsers 是什么?

Output Parsers 是解析和提取语言模型输出的工具。它们可以将语言模型的输出转换为更结构化和有用的形式。

14、LangChain 中 Indexes and Retrievers 是什么?

Indexes and Retrievers 是用于存储和检索信息的索引和数据检索器。它们可以用于提供上下文或从大量数据中检索相关信息。

15、LangChain 中 Chat Message History 是什么?

Chat Message History 是存储和跟踪聊天消息历史的工具。它可以用于维护对话的上下文，以便在多轮对话中提供连贯的向应。

16、LangChain 中 Agents and Toolkits 是什么?

Agents and Toolkits 是提供特定领域功能的代理和工具集。Agents 是一系列可以执行的操作，而 Toolkits 则是为这些操作提供接口和实现的工具集合。

17、LangChain 如何调用 LLMs 生成回复?

LangChain 通过定义好的 Prompt Templates 向 LLMs 发送指令，LLMs 根据这些指令生成文本回复 。

LangChain 还可以使用 Output Parsers 来解析和格式化 LLMs 的输出。

18、LangChain 如何修改提示模板?

在 LangChain 中，可以通过修改 Prompt Templates 的文本内容或变量来定制提示。

19、LangChain 如何链接多个组件处理一个特定的下游任务?

LangChain 通过构建 Chains 来链接多个 Components。每个 Component 执行一个特定的任务，然后将输出传递给链中的下一个 Component，直到完成整个任务。

20、LangChain 如何 Embedding&vector store?

LangChain 可以使用嵌入函数将文本数据转换为向量，并将这些向量存储在向量存储库中。这样做的目的是为了能够高效地检索和查询文本数据。

四、大模型分布式训练

1、大模型进行训练，用的是什么框架?

TensorFlow 是一个由 Google 开发的开源机器学习框架，它提供了强大的分布式训练功能。

TensorFlow 支持数据并行、模型并行和分布式策略等多种分布式训练方法。PyTorch 是一个由 Facebook 的 AI 研究团队开发的流行的开源机器学习库。它提供了分布式包（torch.distributed），支持分布式训练，并且可以通过使用 torch.nn.parallel.DistributedDataParallel（DDP）或 torch.nn.DataParallel 来实现数据并行。

Horovod 是由 Uber 开源的分布式训练框架，它基于 MPI（Message Passing Interface）并提供了一 种

简单的方法来并行化 TensorFlow 、Keras 、PyTorch 和 Apache MXNet 等框架的训练。Horovod 特别适合于大规模的深度学习模型训练。

Ray 是一个开源的分布式框架，用于构建和运行分布式应用程序。Ray 提供了 Ray Tune（用于超参数调优）和 Ray Serve（用于模型服务），并且可以与 TensorFlow、PyTorch 和 MXNet 等深度学习库集成。

Hugging Face 的 Accelerate 库是为了简化 PyTorch 模型的分布式训练而设计的。它提供了一个简单的 API 来启动分布式训练，并支持使用单个或多个 GPU 以及 TPU。

DeepSpeed 是微软开发的一个开源库，用于加速 PyTorch 模型的训练。它提供了各种优化技术，如 ZeRO（Zero

Redundancy Optimizer）和模型并行性，以支持大规模模型的训练。

2、业内常用的分布式 AI 框架?

Horovod：由 Uber 开发，基于 MPI 的分布式训练框架。

Ray：用于构建和运行分布式应用程序的开放源代码框架。

DeepSpeed：由微软开发，用于加速深度学习训练的库，它提供了数据并行、张量并行和模型并行等多种并行策略。

FairScale：由 Facebook 开发，提供了类似于 DeepSpeed 的功能。

3、数据并行、张量并行、流水线并行的原理及区别?

数据并行：在数据并行中，模型的不同副本在不同的设备上运行，每个设备处理输入数据的不 同部分。每个设备独立地进行前向传播和反向传播，但参数更新是同步的。数据并行的主要优点是简单且易于实现。

张量并行：在张量并行中，模型的单个层或参数被切分成多个部分，每个部分在不同的设备上运行。

张量并行通常用于训练非常大型的模型，因为它可以减少每个设备的内存需求。

流水线并行：在流水线并行中，模型的不同层被放置在不同的设备上，每个设备负责模型的一部分。

输入数据在设备之间按顺序流动，每个设备完成自己的计算后将数据传递给下一个设备。流水线并行可以减少每个设备的内存需求，并提高训练速度。

4、推理优化技术 Flash Attention 的作用是什么?

Flash Attention 是一种用于加速自然语言处理模型中自注意力机制的推理过程的优化技术。它通过减少计算量和内存需求，使得在有限的资源下能够处理更长的序列。Flash Attention 使用了一种有效的矩阵乘法算法，可以在不牺牲准确性的情况下提高推理速度。

5、推理优化技术 Paged Attention 的作用是什么?

Paged Attention 是一种用于处理长序列的优化技术。它将注意力矩阵分页，使得只有当前页的注意力分数被计算和存储，从而大大减少了内存需求。这种方法可以在不增加计算成本的情况下处理比内存容量更大的序列。

Flash Attention 是一种高效的注意力机制实现，旨在提高大规模模型训练的速度和内存效率。它通过减少 GPU 内存使用和增加计算吞吐量来实现这一点。

Flash Attention 利用 GPU 上的特定优化，如共享张量核心和高效的内存使用，以减少内存占用 并提高计算速度。这种方法特别适用于具有长序列和大型模型参数的场景，例如自然语言处理和推荐系统。

Paged Attention 是一种用于处理超长序列的注意力机制。在标准的注意力机制中，序列的长度受到 GPU 内存的限制。

Paged Attention 通过将序列分割成多个较小的部分(页面)来克服这个问题，只将当前需要计 算的部分加载到内存中。这种方法允许模型处理比单个 GPU 内存更大的序列，同时保持较高的计算效率。

Paged Attention 对于需要处理极长序列的应用场景(例如长文档处理、音频处理等)非常有用。

6、CPU-offload，ZeRO-offload 了解?

CPU-offload：在深度学习训练中，将一些计算或数据从 GPU 转移到 CPU 上，以减轻 GPU 的负担。这通常用于减少 GPU 内存使用，提高 GPU 利用率。

ZeRO-offload：是 DeepSpeed 中的一种优化技术，它将模型的参数、梯度和优化器状态分散存储 在 CPU 内存或 NVMe 存储中，从而减少 GPU 内存的使用。

ZeRO-offload 是 ZeRO（零冗余优化器）策略的一部分，旨在提高训练大规模模型的能力。

7、ZeRO，零冗余优化器的三个阶段?

ZeRO-Stage 1：将优化器状态分割到不同设备上，减少内存占用。

ZeRO-Stage 2：除了优化器状态，还将模型参数分割到不同设备上。

ZeRO-Stage 3：将梯度和优化器状态也分割到不同设备上，实现最大的内存节省。

8、混合精度训练的优点是什么?可能带来什么问题?

优点：混合精度训练使用不同精度（例如， FP16 和 FP32）的数字来执行计算，可以提高训练速度，减少内存使用，并可能减少能源消耗。它利用了现代 GPU 对 FP16 运算的支持，同时使用 FP32 进行关键的计算，以保持准确性。

可能的问题：混合精度训练可能会导致数值不稳定，特别是在模型梯度非常小或非常大时。此外，它可能需要额外的校准步骤来确保 FP16 计算的准确性。

9、Megatron-DeepSpeed 方法?

Megatron-DeepSpeed 是结合了 Megatron-LM 和 DeepSpeed 的技术，用于训练超大型语言模型。它利用了 Megatron-LM 的模型并行技术和 DeepSpeed 的数据并行和优化器技术，以实现高效的训练。

10、Megatron-LM 方法?

Megatron-LM 是一种由 NVIDIA 开发的用于训练大规模语言模型的模型并行技术。它通过将模型 的不同部分分布在多个 GPU 上，以及使用张量并行和流水线并行等技术，来减少每个 GPU 的内存需求，并提高训练速度。Megatron-LM 已经成功训练了数十亿参数的语言模型。

11、DeepSpeed 方法?

DeepSpeed 是一个开源的库，由微软开发，用于加速大规模模型训练。DeepSpeed 通过多种技术实现了这一点，包括：

数据并行：通过在不同的 GPU 上分配不同的数据批次，来并行处理数据，从而加速训练过程。

模型并行：通过在不同的 GPU 上分配模型的各个部分，来并行处理模型，从而可以训练更大的模型。

管道并行：通过将模型的不同层分配到不同的 GPU 上，并在这些 GPU 之间创建数据流管道，来进一步加速训练过程。

优化器并行：通过将模型的参数分为多个部分，并在不同的 GPU 上并行计算每个部分的梯度更新，来加速优化器步骤。

零冗余优化器（ZeRO）：通过将模型的参数、梯度和优化器状态分割存储在多个 GPU 上，并消除冗余存储，来减少内存使用并提高训练效率。

五、大模型（LLMs）推理

1、为什么大模型推理时显存涨的那么多还一直占着?

模型大小：大模型本身具有更多的参数和计算需求，这直接导致了显存的增加。推理过程中的激活和梯度：在推理时，模型的前向传播会产生激活，这些激活需要存储在显存中，尤其是在执行动态计算或需要中间结果的情况下。

优化器状态：即使是在推理模式下，某些框架可能会默认加载优化器状态，这也会占用显存空间。

内存泄漏:有时代码中的内存泄漏会导致显存一直被占用，而不是在推理完成后释放。

要解决显存占用问题，可以采用的技术包括使用内存分析工具来检测泄漏，优化模型结构，或者使用如 TensorFlow 的内存管理功能来显式释放不再需要的内存。

2、大模型在 GPU 和 CPU 上推理速度如何?

大模型在 GPU 上的推理速度通常远快于 CPU，因为 GPU 专门为并行计算设计，具有更多的计算 核心和更高的浮点运算能力。例如，NVIDIA 的 GPU 使用 CUDA 核心，可以同时处理多个任务，这使得它们在执行深度学习推理时非常高效。

CPU 虽然也可以执行深度学习推理任务，但由于其核心数量和浮点运算能力通常不及 GPU，因 此速度会慢得多。然而， CPU 在处理单线程任务时可能更高效，且在某些特定场景下，如边缘计算设备上，CPU 可能是唯一可用的计算资源。

3、推理速度上， int8 和 fp16 比起来怎么样?

INT8（8 位整数）和 FP16（16 位浮点数）都是低精度格式，用于减少模型的大小和提高推理速度。

INT8 提供更高的压缩比，可以显著减少模型的内存占用和带宽需求，但由于量化过程中的 信息损失，可能会对模型的准确性产生一定影响。FP16 提供比 INT8 更高的精度，通常对模型的准确性影响较小，但相比 INT16 或 FP32，它的速度和内存效率仍然有所提高。

在实际应用中， INT8 和 FP16 的推理速度取决于具体的模型和硬件。一般来说，INT8 可能会提供 更高的吞吐量，但 FP16 可能会提供更好的延迟和准确性。例如，NVIDIA 的 Tensor Cores 支持 FP16 和 INT8 运算，可以显著提高这两种格式的推理性能。

4、大模型有推理能力吗?

大模型 (LLMs) 具有推理能力。推理能力不仅限于回答事实性问题，还包括理解复杂语境、生 成连贯文本、执行文本分类、翻译等任务。例如，GPT-3 是一个大模型，它能够生成文章、故事、诗歌，甚至编写代码。

5、大模型生成时的参数怎么设置?

大模型生成时的参数设置取决于具体的任务和模型。一些常见的参数包括：温度 (Temperature)：控制生成的文本的随机性。较低的温度值将导致生成更保守的文本，而较高的温度值将导致更多样化的文本。

Top-k 采样：仅从概率最高的 k 个词中采样，以减少生成文本的随机性。

Top-p 采样：从累积概率超过 p 的词中进行采样，这有助于生成更相关的文本。

最大生成长度：指定生成文本的最大长度。

例如，使用 GPT-3 生成文本时，可以设置温度为 0.7,top-k 为 50,最大生成长度为 100 个词。

6、有哪些省内存的大语言模型训练/微调/推理方法?

模型并行：将模型的不同部分分布在多个设备上。

张量切片：将模型的权重和激活分割成较小的块。

混合精度训练：使用 FP16 和 INT8 精度进行训练和推理。

优化器状态分割：如 ZeRO 技术，将优化器状态分割到不同设备上。

梯度累积：通过累积多个批次的梯度来减少每个批次的内存需求。

在机器学习中，优化器状态是指在训练模型时优化器所维护的关于模型参数更新的额外信息。这些信息对于执行梯度下降算法的变体（如 Adam、RMSprop、SGD 等）至关重要，因为它们帮助优化器更有效地调整模型参数。

优化器状态通常包括以下几个关键组件：

梯度：在反向传播过程中计算的权重参数的梯度，指示了损失函数相对于每个参数的斜率。

动量：某些优化器（如 SGD with Momentum、Adam 等）会使用动量来平滑参数更新，这可以帮 助优化器在相关方向上加速学习，并减少震荡。

平方梯度：某些优化器（如 RMSprop、Adam）会保存每个参数梯度的平方的移动平均，这有助于调整学习率并稳定训练过程。

学习率：优化器可能会根据训练的进度或某些其他信号调整每个参数的学习率。

其他统计量：某些优化器可能会使用其他统计量，如 Adam 优化器会维护梯度的一阶和二阶矩的估计。

优化器状态对于实现高效的参数更新至关重要。在训练过程中，优化器会根据这些状态信息来 计算每个迭代步骤中参数的更新量。在分布式训练设置中，如 DeepSpeed 中的 ZeRO 优化器，优化器状态的

管理变得尤为重要，因为它们需要跨多个 GPU 或节点高效地分配和同步。

7、如何让大模型输出合规化?

过滤不当内容：使用内容过滤器来识别和过滤掉不当的语言或敏感内容。

指导性提示：提供明确的提示，指导模型生成符合特定标准和偏好的输出。

后处理：对模型的输出进行后处理，例如使用语法检查器和修正工具来提高文本的质量。

强化学习：使用强化学习来训练模型，使其偏好生成符合特定标准的输出。

应用模式变更：应用模式变更是指在部署模型时，根据实际应用的需求和环境，对模型的配置、部署策略或使 用方式进行调整。例如， 一个在云端运行的模型可能需要调整其资源分配以适应不同的负载，或者在边缘设备上运行的模型可能需要减少其内存和计算需求以适应有限的资源。

应用模式变更可能包括：

资源调整：根据需求增加或减少用于运行模型的计算资源。

模型压缩：使用模型压缩技术如剪枝、量化来减少模型大小。

动态部署：根据负载动态地扩展或缩小模型服务的实例数量。

缓存策略：实施缓存机制来存储常用查询的响应，减少重复计算的次数。

性能优化：对模型进行性能分析，并优化其运行效率，例如通过批处理输入数据来提高吞吐量。

举例来说，如果一个大型语言模型在云平台上运行，当用户查询量增加时，可以通过增加服务 器的数量或使用更高效的硬件来扩展其能力。相反，如果模型需要在嵌入式设备上运行，可能需要将模型压缩到更小的尺寸，并优化其运行时的内存使用，以确保模型可以在资源有限的设备上顺利运行。

在实际操作中，应用模式变更通常需要综合考虑模型的性能、成本、可扩展性和业务需求，以找到最佳的平衡点。