1

AI 大模型基础面

Q1：目前主流的大模型体系有哪些？

A1：目前主流的开源大模型体系包括以下几个：

- GPT（Generative Pre-trained Transformer）系列：由 OpenAI 发布的一系列基于 Transformer 架构的语言模型，包括 GPT-1、GPT-2、GPT-3、ChatGPT 等。GPT 模型通过在大规模无标签文本上进行预训练，然后在特定任务上进行微调，具有很强的生成能力和语言理解能力。
- BERT（Bidirectional Encoder Representations from Transformers）：由 Google 发布的一种基于 Transformer 架构的双向预训练语言模型。BERT 模型通过在大规模无标签文本上进行预训练，然后在下游任务上进行微调，具有强大的语言理解能力和表征能力。
- XLNet：由 CMU 和 Google Brain 发布的一种基于 Transformer 架构的自回归预训练语言模型。XLNet 模型通过自回归方式预训练，可以建模全局依赖关系，具有更好的语言建模能力和生成能力。

- RoBERTa：由 Meta 发布的一种基于 Transformer 架构的预训练语言模型。RoBERTa 模型在 BERT 的基础上进行了改进，通过更大规模的数据和更长的训练时间，取得了更好的性能。
- T5（Text-to-Text Transfer Transformer）：由 Google 发布的一种基于 Transformer 架构的多任务预训练语言模型。T5 模型通过在大规模数据集上进行预训练，可以用于多种自然语言处理任务，如文本分类、机器翻译、问答等。

这些大模型在自然语言处理领域取得了显著的成果，并被广泛应用于各种任务和应用中。

Q2：涌现能力是啥原因？

A2：大模型的涌现能力主要是由以下几个原因：

- 数据量的增加：随着互联网的发展和数字化信息的爆炸增长，可用于训练模型的数据量大大增加。更多的数据可以提供更丰富、更广泛的语言知识和语境，使得模型能够更好地理解和生成文本。
- 计算能力的提升：随着计算硬件的发展，特别是图形处理器（GPU）和专用的 AI 芯片（比如：TPU）的出现，计算能力大幅提升。这使得训练更大、更

复杂的模型成为可能，从而提高了模型的性能和涌现能力。

- 模型架构的改进：近年来，一些新的模型架构被引入，比如：Transformer，它在处理序列数据上表现出色。这些新的架构通过引入自注意力机制等技术，使得模型能够更好地捕捉长距离的依赖关系和语言结构，提高了模型的表达能力和生成能力。
- 预训练和微调的方法：预训练和微调是一种有效的训练策略，可以在大规模无标签数据上进行预训练，然后在特定任务上进行微调。这种方法可以使模型从大规模数据中学习到更丰富的语言知识和语义理解，从而提高模型的涌现能力。

综上所述，大模型的涌现能力是由数据量的增加、计算能力的提升、模型架构的改进以及预训练和微调等因素共同作用的结果。这些因素的进步使得大模型能够更好地理解和生成文本，为自然语言处理领域带来了显著的进展。

2

AI 大模型进阶面

Q3：大模型如何选型？如何基于场景选用 ChatGLM、LlaMa、Bert 类大模型？

A3：选择使用哪种大模型，取决于具体的应用场景和需求。下面是一些指导原则。

- ChatGLM 大模型：ChatGLM 是一个面向对话生成的大语言模型，适用于构建聊天机器人、智能客服等对话系统。如果你的应用场景需要模型能够生成连贯、流畅的对话回复，并且需要处理对话上下文、生成多轮对话等，ChatGLM 模型可能是一个较好的选择。ChatGLM 的架构为 Prefix Decoder，训练语料为中英双语，中英文比例为 1:1。所以适合于中文和英文文本生成的任务。
- LlaMA 大模型：LLaMA（Large Language Model Meta AI）包含从 7B 到 65B 的参数范围，训练使用多达 14,000 亿 tokens 语料，具有常识推理、问答、数学推理、代码生成、语言理解等能力。它由一个 Transformer 解码器组成。训练预料主要为以英语为主的拉丁语系，不包含中日韩文。所以适合于英文文本生成的任务。
- Bert 大模型：Bert 是一种预训练的大语言模型，适用于各种自然语言处理任务，如文本分类、命名实体识别、语义相似度计算等。如果你的任务是通

用的文本处理任务，而不依赖于特定领域的知识或语言风格，Bert 模型通常是一个不错的选择。Bert 由一个 Transformer 编码器组成，更适合于 NLU 相关的任务。

在选择模型时，还需要考虑以下因素：

- 数据可用性：不同模型可能需要不同类型和规模的数据进行训练。确保你有足够的数据来训练和微调所选择的模型。
- 计算资源：大模型通常需要更多的计算资源和存储空间。确保你有足够的硬件资源来支持所选择的模型的训练和推理。
- 预训练和微调：大模型通常需要进行预训练和微调才能适应特定任务和领域。了解所选择模型的预训练和微调过程，并确保你有相应的数据和时间来完成这些步骤。

最佳选择取决于具体的应用需求和限制条件。在做出决策之前，建议先进行一些实验和评估，以确定哪种模型最适合你的应用场景。

Q4：各个专业领域是否需要专用的大模型来服务？

A4：各个专业领域通常需要各自的专用大模型来服务，原因如下：

- 领域特定知识：不同领域拥有各自特定的知识和术语，需要针对该领域进行训练的大模型才能更好地理解和处理相关文本。比如：在医学领域，需要训练具有医学知识的大模型，以更准确地理解和生成医学文本。
- 语言风格和惯用语：各个领域通常有自己独特的语言风格和惯用语，这些特点对于模型的训练和生成都很重要。专门针对某个领域进行训练的大模型可以更好地掌握该领域的语言特点，生成更符合该领域要求的文本。
- 领域需求的差异：不同领域对于文本处理的需求也有所差异。比如：金融领域可能更关注数字和统计数据的处理，而法律领域可能更关注法律条款和案例的解析。因此，为了更好地满足不同领域的需求，需要专门针对各个领域进行训练的大模型。
- 数据稀缺性：某些领域的数据可能相对较少，无法充分训练通用的大模型。针对特定领域进行训练的大模型可以更好地利用该领域的数据，提高模型的性能和效果。

尽管需要各自的大模型来服务不同领域，但也可以共享一些通用的模型和技术。比如：通用的大模型可以用于处理通用的文本任务，而领域特定的模型可以在通用模型的基础上进行微调和定制，以适应特定领域的需求。这样可以在满足领域需求的同时，减少模型的重复训练和资源消耗。

3

AI 大模型 LangChain 开发框架面

Q5：LangChain Agent 是如何工作和使用？

A5：LangChain Agent 是 LangChain 框架中的一个组件，用于创建和管理对话代理。最新发布的首个稳定版本 v0.1.0 支持了 LangGraph 组件库，把 Agent 创建为图的组件库，提供创建更加定制化的循环行为。代理是根据当前对话状态确定下一步操作的组件。LangChain 提供了多种创建代理的方法，包括 OpenAI Function Calling、Plan-and-execute Agent、Baby AGI 和 Auto GPT 等。这些方法提供了不同级别的自定义和功能，用于构建代理。代理可以使用工具包执行特定的任务或操作。工具包是代理使用的一组工具，用于执行特定的功能，如语言处理、数据操作和外部 API 集成。工具可以是自

定义构建的，也可以是预定义的，涵盖了广泛的功能。通过结合代理和工具包，开发人员可以创建强大的对话代理，能够理解用户输入，生成适当的回复，并根据给定的上下文执行各种任务。以下是使用 LangChain 创建代理的示例代码：

```
from langchain.chat_models import ChatOpenAI
from langchain.agents import tool

# 加载语言模型
llm = ChatOpenAI(temperature=0)

# 定义自定义工具
@tool
def get_word_length(word: str) -> int:
    """返回单词的长度。"""
    return len(word)

# 创建代理
agent = {
    "input": lambda x: x["input"],
    "agent_scratchpad": lambda x: format_to_openai_fu
nctions(x['intermediate_steps'])
```

```
} | prompt | llm_with_tools | OpenAIFunctionsAgentOut
putParser()

# 调用代理
output = agent.invoke({
    "input": "单词 educa 中有多少个字母? ",
    "intermediate_steps": []
})

# 打印结果
print(output.return_values["output"])
```

这只是一个基本示例，LangChain 中还有更多功能和功能可用于构建和自定义代理和工具包。您可以参考 LangChain 文档以获取更多详细信息和示例。

4

AI 大模型向量数据库面

Q6：基于大模型 + 向量数据库如何更好地实现企业级知识库平台？

A6：主要进行以下 6 方面的优化工作：

- 数据准备：准备大量高质量的训练数据，包括 Query、Context 和对应的高质量 Response。确保数据的多样性和覆盖性，以提供更好的训练样本。
    - 模型架构：选择合适的模型架构，比如：Transformer 等，以便提取 Query 和 Context 中的重要信息，并生成相应的高质量 Response。确保大模型具有足够的容量和复杂性，以适应各种复杂的查询和上下文。
    -
- 微调和优化：使用预训练的模型作为起点，通过在特定任务上进行微调和优化，使模型能够更好地理解 Query 和 Context，并生成更准确、连贯的 Response。可以使用基于强化学习的方法，比如：强化对抗学习，来进一步提高模型的表现。
    - 评估和反馈：定期评估模型的性能，使用一些评估指标，比如：BLEU、ROUGE 等，来衡量生成的 Response 的质量。根据评估结果，及时调整和改进模型的训练策略和参数设置。同时，收集用户反馈和意见，以便进一步改进模型的性能。

- 多模态信息利用：如果有可用的多模态信息，如图像、视频等，可以将其整合到大模型中，以提供更丰富、准确的 Response。利用多模态信息可以增强模型的理解能力和表达能力，从而生成更高质量的 Response。