LMCC-例卷-青少年组

单选题（20 * 3 分 = 60 分）

1. 在掩码语言模型预训练中，核心任务“掩码预测”的主要目的是什么？

A. 学习语言的深层双向上下文表征

B. 评估模型最终的分类准确率

C. 专门优化模型的文本生成流畅度

D. 减少模型训练所需的计算资源

2. 小明和小华在学习大语言模型时，尝试让它写太空探险的故事。

小明的提示词是：“写一个关于太空探险的故事。”

小华的提示词是：“请你扮演一位经验丰富的科幻小说家，为中学生写一个大约 500 字的短篇故事。故事的主角是一位名叫‘星尘’的年轻宇航员，他在探索一颗未知行星时，意外发现了一个古老外星文明的遗迹。故事风格要紧张刺激，结局要留下悬念。”

结果小华生成的故事比小明更好。从提示词工程的角度分析，为什么小华能获得远比小明更好的结果？

A. 因为小华提问时，大语言模型的知识库恰好更新了与“外星文明”相关的数据，所以能写出更具体的内容。

B. 因为小华的提示词更长、更复杂，根据提示工程原理，投入的文字量与生成质量成正比，只要提示词足够长，效果就会更好。

C. 因为小华的提示词为模型设定了明确的角色、目标受众和具体约束，有效地引导了模型的思维链，使其在庞大的可能性空间中聚焦于一个高质量的输出方向。

D. 因为大语言模型具有真正的创造力和情感理解能力，小华的提示词中“紧张刺激”和“悬念”等词语激发了它的创作灵感。

3. Transformer 架构中，使大语言模型能够有效处理长距离依赖关系的关键组件是：

A. 前馈神经网络

B. 层归一化

C. 自注意力机制

D. 残差连接

4. 指令微调的主要目的是什么?

A. 让模型从零开始学习海量无标注数据中的通用知识。

B. 增强模型理解和遵循人类指令或意图的能力，并改善其输出风格。

C. 大幅降低模型的基础参数量，以提高推理速度。

D. 专门用于修复模型在预训练阶段产生的事实性错误。

5. 大语言模型中存在的社会偏见，其主要根源最可能是:

A. 模型训练时使用的 GPU 硬件存在设计缺陷。

B. 模型 Transformer 架构中的注意力机制算法存在固有偏差。

C. 模型从预训练数据中学习了现实世界中存在的社会偏见。

D. 模型在指令微调阶段，工程师有意注入了有偏见的指令。

6. 目前主流的生成式大语言模型（如 GPT 系列）的核心架构通常基于:

A. 编码器模型

B. 编码器-解码器模型

C. 解码器模型

D. 卷积神经网络模型

7. 用于指令微调的数据通常具有什么典型特征?

A. 大规模、无标注的原始网页文本。

B. 由（指令/问题， 期望输出）配对组成的监督数据集。

C. 纯粹的代码仓库和编程语言片段。

D. 未经处理的原始音频和图像数据。

8. 关于“贪心搜索”和“束搜索”这两种解码策略，以下哪种说法是最准确的?

A. 贪心搜索因为每次都选最好的，所以总能生成最完美、最富有创造力的句子。

B. 束搜索需要同时考虑多条路径，计算起来更复杂，但它更有可能找到一个整体上更

通顺、更合理的句子。

C. 束搜索的速度通常比贪心搜索更快，因为它是并行的。

D. 这两种策略没有本质区别，无论用哪一种，AI 生成的句子质量都完全一样。

9. 大模型“人类对齐”的核心目标是使得模型的行为：

A. 在所有的数学计算上达到零误差。

B. 无限接近地在所有任务上超越人类专家水平。

C. 符合人类的价值观、意图，并做到安全、有帮助。

D. 其内部神经网络的计算过程对人类完全透明可解释。

10. 大语言模型在解决一道复杂的数学应用题时，生成了一段非常长的思考过程，其中包含了将问题分解为多个子步骤、对每个步骤进行详细解释、并逐步推导出中间结果。这种推理模式的主要优势是什么？

A. 能够确保最终答案的正确性，避免计算错误。

B. 通过展示详尽的思维过程，提高了解题逻辑的可解释性和可靠性。

C. 显著减少了模型处理问题所需的总时间和计算资源。

D. 主要目的是为了生成更多的文本内容，使回答看起来更丰富。

11. 关于 AI 智能体的角色配置文件设置，下列哪个说法是最准确的？

A. 角色配置文件的主要作用是限制智能体的知识量，防止它回答超出范围的问题。

B. 角色配置文件设置一旦完成就无法改变，智能体将严格按照初始设定运行。

C. 角色配置文件是定义智能体核心身份、行为准则和对话风格的关键，通常通过初始系统指令实现。

D. 角色配置文件的功能与用户每次对话的提问内容作用相同，都是为智能体提供临时信息。

12. 在评估一个代码生成智能体时，我们常使用 `Pass@K` 指标。假设我们有一个包含 `n=100` 个编程问题的测试集。对于每个问题，让智能体独立生成 `k=5` 个不同的代码解决方案。如果对于某个问题，生成的 5 个方案中有任意一个能够通过单元测试，该问题就被视为“已解决”。最终，在 100 个问题中，有 60 个问题被成功解决。

根据上述描述，以下关于 `Pass@K` 的计算和解释，哪一项是正确的？

A. 这里 K=5, Pass@5 = (60 / 100) * 100% = 60%。这表示随机抽取一个生成方案，其通过测试的概率是 60%。
B. 这里 K=5, Pass@5 = (60 / 100) * 100% = 60%。这表示通过生成 5 个方案，智能体成功解决其中 60%问题的能力。
C. 这里 K=100, Pass@100 = (60 / 100) * 100% = 60%。这表示智能体解决了测试集中 60%的问题。
D. 这里 K=5, 但正确的计算方式应为 `Pass@5 = (60 / (100 * 5)) * 100% = 12%`。这表示所有生成方案的总通过率。

13. 下列哪一项通常不属于对大语言模型进行系统评测的核心内容？

A. 模型训练所用 GPU 集群的实时功耗与散热效率

B. 模型在问答、摘要、代码生成等任务上的准确率与流畅度

C. 模型输出是否存在社会偏见、歧视性内容等安全性问题

D. 模型在数学推理、常识推理等复杂任务上的泛化能力

14. 在 Transformer 的自注意力机制中，关于 Query、Key 和 Value 的来源，以下哪一项描述是最准确的？

A. Query、Key 和 Value 是由三个完全不同的、随机初始化的参数矩阵生成的，与输入序列无关。

B. Key 和 Value 是输入序列的嵌入向量本身，而 Query 是目标序列的嵌入向量本身。

C. Query 来自编码器的最终隐藏状态，而 Key 和 Value 来自解码器的输入序列。

D. Query、Key 和 Value 是由同一个输入向量通过三个不同的线性变换层（权重矩阵）投影得到的。

15. 下列哪一项不属于测试时间拓展（Test Time Scaling，TTS）技术？

A. 对单个测试样本进行多种变换（如翻转、裁剪），并将所有变换版本的预测结果进行集成。

B. 在模型推理时，仍然随机丢弃部分神经元，进行多次前向传播并将结果平均，以模拟集成效果提升健壮性。

C. 为每个测试样本定义一个辅助任务（如旋转图片预测），利用该样本对模型进行一步梯度下降微调，再完成主任务预测。

D. 在一个大型源数据集上预训练模型，然后在一个较小的目标数据集上对模型的所有权重进行微调。集上对模型的所有权重进行微调。

16. 为了保障社会群体的“公平安全”，避免模型输出产生歧视性内容，最关键的缓解方法是在哪个阶段介入？

A. 仅在模型推理部署后，通过实时过滤敏感词来屏蔽有害输出。

B. 主要依靠扩大模型规模，期望其自动学习并消除偏见。

C. 在模型预训练和微调阶段，系统性地清洗数据和引入公平性约束。

D. 完全依赖第三方评测机构在发布前进行一次性的偏见评估。

17. 当前意义上的“大语言模型”最显著的特征不包括以下哪一项？

A. 模型参数量巨大，但仅在特定领域的小规模数据上训练。

B. 展现出强大的涌现能力，如推理、编程和创造性写作。

C. 基于“下一个词预测”等自监督目标在海量互联网文本上训练。

D. 具有强大的上下文学习能力，无需梯度更新即可根据提示完成新任务。

18. 基于强化学习的推理能力优化方法面临的主要挑战是什么？

A. 模型会因此完全丧失其原有的语言生成能力。

B. 推理路径的搜索空间巨大，导致训练不稳定和效率低下。

C. 强化学习会使模型的词汇量急剧减少。

D. 该方法只能应用于数学推理，无法泛化到其他领域。

19. 关于语言模型的困惑度，以下描述正确的是：

A. 困惑度越低，说明模型对测试数据的预测越不确定，性能越差。

B. 困惑度是模型在训练集上的准确率直接转换而来的指标。

C. 困惑度反映模型对一组文本的平均预测不确定性，值越低表示模型性能越好。

D. 困惑度与模型的训练步数无关，只与模型架构有关。

20. 关于监督学习和无监督学习的区别，以下说法正确的是：

A. 监督学习不需要训练数据，无监督学习需要大量标注数据

B. 监督学习使用标注数据学习，无监督学习从无标签数据中发现模式

C. 无监督学习只能处理数值型数据，监督学习可以处理任何类型数据

D. 无监督学习的模型性能总是优于监督学习

//

选择题详细解答

1. 在掩码语言模型预训练中，核心任务“掩码预测”的主要目的是什么？

答案 ：A

解析 ：掩码语言模型（如 BERT）在预训练时，会随机掩盖输入文本中的部分词（如用“[MASK]”替换），让模型根据上下文预测被掩盖的词。这一过程的核心目的是迫使模型学习双向上下文信息（即同时利用左右词语的关系），从而理解语言的深层语义和语法结构。

例如，在句子“The quick brown [MASK] jumps...”中，模型需结合“quick brown”和“jumps”来预测“fox”。选项 B、C、D 偏离了掩码预测的设计本质：它不直接评估准确率（B）、不专门优化生成流畅度（C，生成任务通常用自回归模型），也无法减少计算资源（D，反而增加训练成本）。

选项分析技巧 ：

- 注意目标关键词：题目问“主要目的”，选项 A 中的“学习深层双向上下文表征”直接对应掩码预测的核心功能（如搜索结果中强调的“双向上下文”）。
- 排除法：B、C、D 的描述过于具体或功利（如“评估准确率”“减少资源”），而预训练任务通常是基础性、通用性的（如“学习表征”）。

---

2. 小明和小华的故事生成提示词对比分析

答案 ：C

解析 ：小华的提示词通过设定角色（科幻小说家）、受众（中学生）、具体约束（500 字、主角名字、情节要素）和风格要求，为模型提供了清晰的引导。这能帮助模型缩小生成范围，聚焦于高质量、符合预期的输出（参考智能客服系统设计中对用户意图和上下文的管理）。而小明的提示词过于宽泛，模型容易产生泛泛而谈的内容。选项 A 错误（模型知识库更新非用户可控）；B 错误（提示词长度与质量非正比，关键在有效性）；D 错误（模型无真实创造力，仅基于统计模式）。

选项分析技巧 ：

- 比较选项的合理性：C 选项提到“角色、约束、引导思维链”，这些是提示词工程的常见原则（如搜索结果中的“多轮对话管理”），而 A、B、D 依赖偶然因素

（如“恰好更新数据”）或夸大模型能力（如“真正创造力”）。

---

3. Transformer 架构中处理长距离依赖的关键组件

答案 ：C

解析 ：自注意力机制（Self-Attention）是 Transformer 的核心，它允许模型在处理序列时动态计算每个位置与其他所有位置的关联权重，从而直接捕捉长距离依赖。

例如，在句子“The cat which ate the fish sitting by the river was full.”中，模型能直接关联“cat”和“was”，不受中间词语距离影响。其他选项虽重要但非关键：前馈神经网络（A）负责局部变换；层归一化（B）稳定训练；残差连接（D）缓解梯度消失，但不直接解决长距离依赖。

选项分析技巧 ：

- 关联技术名称：搜索结果显示，“自注意力机制”常与“长距离依赖”并列出现（如“Transformer 模型的核心机制是自注意力”

），而其他选项（如“残差连接”）更通用。

---

4. 指令微调的主要目的

答案 ：B

解析 ：指令微调（Instruction Tuning）使用（指令/问题，期望输出）配对数据，让模型学习遵循人类指令的意图和风格，使其输出更符合需求（如更简洁、安全或具体）。

例如，微调后模型能更好区分“写首诗”和“用莎士比亚风格写首诗”。选项 A 是预训练的目的；C 错误（微调不减少参数量）；D 错误（修正事实错误需额外技术如检索增强）。

选项分析技巧 ：

- 抓取关键词：指令微调关注“意图”“风格”（选项 B），而 A 中的“从零学习”和 D 中的“修复错误”与微调的定义不符。

---

5. 大语言模型中社会偏见的主要根源

答案 ：C

解析 ：大模型通过预训练从互联网文本中学习统计模式，而这些数据本身蕴含现实社会的偏见（如性别、种族刻板印象），导致模型复制偏见。

例如，训练数据中“护士”常关联女性，模型可能生成有偏见的回答。选项 A、B、D 错

误：硬件缺陷（A）和注意力算法（B）不直接导致偏见；指令微调通常旨在减少偏见，而非注入（D）。

选项分析技巧　　：

- 逻辑推断：偏见是“学习”而非“硬件”或“算法固有”问题（排除 A、B）；而微调阶段通常纠正偏见而非引入（排除 D）。

---

6. 主流生成式大语言模型的核心架构

答案　　：C

解析　　：GPT 系列、LLaMA 等生成模型采用解码器（Decoder-only）架构，通过自回归方式逐个生成词（如根据上文预测下一个词）。

编码器模型（A）如 BERT 适用于理解任务；编码器-解码器模型（B）如 T5 用于翻译等；卷积网络（D）处理图像。

选项分析技巧　　：

- 关联常见模型：GPT、LLaMA 等知名生成模型常被描述为“解码器模型”（搜索结果中对比 MLM 和 CLM 时提及）。

---

7. 指令微调数据的典型特征

答案　　：B

解析　　：指令微调数据由人工标注的（指令/问题，期望输出）配对组成，例如“指令：总结这段文本；输出：摘要内容”。

选项 A 是预训练数据；C、D 与其他模态任务相关。

选项分析技巧　　：

- 对比选项：B 明确提到“配对组成的数据集”，与“指令微调”的名称直接对应，而 A、C、D 描述的数据类型不匹配。

---

8. 贪心搜索与束搜索的比较

答案　　：B

解析　　：贪心搜索每一步选概率最高的词，可能陷入局部最优；束搜索（Beam Search）保留多条候选路径（如 Top-K），最终选择整体概率最高的序列，生成结果更通顺

（参考缓存替换策略中的多路径优化）。A 错误（贪心搜索可能生成重复内容）；C 错误（束搜索更耗时）；D 错误（质量差异显著）。

选项分析技巧 ：

- 关注权衡关系：B 提到“更复杂但更可能找到合理句子”，体现计算成本与质量的平衡，其他选项有绝对化描述（如“总能”“完全一样”）。

---

9. 大模型“人类对齐”的核心目标

答案 ：C

解析 ：人类对齐（Human Alignment）旨在使模型行为符合人类价值观（如安全、公平、帮助性），通常通过 RLHF 等技术实现。

A、B、D 是具体技术目标，而非对齐的核心。

选项分析技巧 ：

- 关键词匹配：C 中的“价值观”“安全”与搜索结果中的“伦理”“公平性”一致

，而 A、B、D 局限于性能或透明性。

---

10. 模型生成详细推理过程的优势

答案 ：B

解析 ：展示逐步推理（如思维链）能提高逻辑的可解释性，让用户理解模型如何得出结论，增强信任

（参考缓存机制中“可解释性”的重要性）。A 错误（不保证正确性）；C 错误（增加时间成本）；D 片面（本质是提升可靠性）。

选项分析技巧 ：

- 区分主次：B 强调“可解释性和可靠性”，是思维链的核心价值，而 A、C、D 关注次要方面（如正确性、效率）。

---

11. AI 智能体角色配置文件的作用

答案 ：C

解析 ：角色配置文件通过系统指令定义智能体的身份、行为准则和风格（如“你是一位医生”），确保输出一致性

（参考多租户缓存隔离中的身份划分）。A 错误（不限制知识量）；B 错误（可动态更新）；D 错误（配置文件是持久设置，而非临时信息）。

选项分析技巧 ：

- 抓住核心功能：C 中的“定义身份、准则”是配置文件的本质，其他选项描述过

于片面或错误。

---

12. Pass@K 指标的正确理解

答案 ：B

解析 ：Pass@K 衡量模型生成 K 个方案时解决问题的比例。本例中，K=5，解决 60/100 问题，Pass@5=60%，表示通过生成 5 个方案，模型能解决 60%的问题

（类似缓存命中率的计算）。A 错误（Pass@K 不针对单个方案）；C 错误（K 是生成数，非问题数）；D 错误（分母是问题数，非总方案数）。

选项分析技巧 ：

- 理解指标定义：Pass@K 关注“解决问题比例”，而非单个方案通过率（排除 A、D）；K 通常指生成方案数（排除 C）。

---

13. 不属于大模型系统评测的内容

答案 ：A

解析 ：模型评测关注性能（如准确率、B）、安全性（C）、泛化能力（D），而硬件功耗（A）是工程优化问题，非核心评测内容。

选项分析技巧 ：

- 区分本质与附属：A 中的“GPU 功耗”与模型能力无关，其他选项均涉及模型输出质量。

---

14. 自注意力机制中 Query、Key、Value 的来源

答案 ：D

解析 ：Query、Key、Value 由输入向量通过三个独立的线性变换层投影得到，使模型灵活学习不同表示。

A 错误（与输入相关）；B、C 混淆了编码器-解码器结构（适用于翻译任务）。

选项分析技巧 ：

- 技术细节一致性：D 的描述“同一输入通过不同变换”是自注意力的标准定义，其他选项含绝对化错误（如“完全无关”）。

---

15. 不属于测试时间拓展（TTS）的技术

答案 ：D

解析 ：TTS 技术在推理时优化模型（如数据增强、Dropout 集成），而 D 的“预训练+微调”是训练阶段方法，非测试时技术

（参考缓存冷启动与分层策略）。A、B、C 均为 TTS 常见手段。

选项分析技巧 ：

- 时间阶段判断：TTS 针对“测试时”，D 明确提到“预训练和微调”，属于训练阶段，直接排除。

---

16. 保障公平安全的关键方法

答案 ：C

解析 ：偏见缓解需在训练阶段系统化处理，如清洗数据、添加公平性约束。

A、B、D 是事后补救，效果有限。

选项分析技巧 ：

- 根本原因分析：偏见源于数据（题目 5），故需从训练阶段解决（C），其他选项偏表面。

---

17. 大语言模型的显著特征

答案 ：A

解析 ：大模型需同时满足参数量大、海量数据训练、涌现能力等特征。

A 错误（大模型需海量数据）。B、C、D 均为典型特征。

选项分析技巧 ：

- 特征完整性：A 中的“小规模数据”与“大模型”定义矛盾（搜索结果显示需“海量数据”），其他选项符合常见描述。

---

18. 强化学习推理优化的主要挑战

答案 ：B

解析 ：强化学习搜索空间大（如数学推理需尝试多种路径），导致训练不稳定和低效

（参考推理缓存中的动态负载问题）。A、C、D 夸大或错误。

选项分析技巧 ：

- 问题定位：B 中的“搜索空间大”是 RL 常见挑战，其他选项（如“丧失生成能力”）缺乏依据。

---

19. 关于困惑度的描述

答案 ：C

解析 ：困惑度（Perplexity）衡量模型对文本的预测不确定性，值越低表示模型越准确。

A 错误（低困惑度代表高确定性）；B 错误（基于测试数据）；D 错误（与训练相关）。

选项分析技巧 ：

- 指标方向：困惑度“越低越好”是常识，C 符合这一逻辑，A 反向描述。

---

20. 监督学习与无监督学习的区别

答案 ：B

解析 ：监督学习使用标注数据（如分类任务），无监督学习从无标签数据发现模式（如聚类）。

A、C、D 均描述错误。

选项分析技巧 ：

- 核心定义：B 直接对应两种学习范式的本质区别，其他选项含明显错误（如“无监督学习不需要数据”）。

//

程序题（10 * 4 分 = 40 分）

一、检索增强生成

这是一个基于检索增强生成（Retrieval-Augmented Generation, RAG）的问答系统实现。该系统包含两个主要组件：

1. 检索器（Retriever）：使用嵌入模型将文档转换为向量，根据查询语义检索最相关的文档
2. 生成器（Generator）：基于检索到的相关文档和原始问题，生成准确的答案

请根据代码逻辑和注释提示，补全代码中的空位。

```Python
import torch
import torch.nn.functional as F
from transformers import AutoTokenizer, AutoModel,
AutoModelForCausalLM

class Retriever:
    def __init__(self, embedder, tokenizer, corpus):
        self.embedder = embedder
        self.tokenizer = tokenizer
        self.corpus = corpus
        self.embeddings = self.build_index(corpus)

    def _text_to_vector(self, text_batch):
        enc = ____[21]____
        enc = {k: v.to(self.embedder.device) for k, v in
enc.items()}

        with torch.no_grad():
            outputs = ____[22]____
            last_hidden = outputs.last_hidden_state
            attn_mask = enc["attention_mask"]
            sent_vecs = ____[23]____

        return sent_vecs

    def build_index(self, corpus):
        vecs_tensor = self._text_to_vector(corpus)
        vecs_normalized = F.normalize(vecs_tensor, p=2, dim=1)
        return vecs_normalized.cpu()

    def search(self, query_vec, topk=3):
        if query_vec.dim() == 1:
            query_vec = query_vec.unsqueeze(0)
        query_normalized = F.normalize(query_vec, p=2, dim=1)
        similarities = torch.matmul(query_normalized,
self.embeddings.T)
        similarities = similarities.squeeze(0)

        topk_scores, topk_indices = torch.topk(similarities,
k=min(topk, len(self.corpus)))
```

```
        results = [
            {"text": self.corpus[idx], "score":
float(topk_scores[i])}
            for i, idx in enumerate(topk_indices)
        ]
        return results

def rag_answer(query, retriever, generator, gen_tokenizer, topk=3,
max_new_tokens=128):
    query_vec_tensor = retriever._text_to_vector([query])
    query_vec = query_vec_tensor.squeeze(0).cpu()

    docs = ____[24]____

    context = "\n".join([d["text"] for d in docs]) if docs else ""
    prompt = f"Use the context to
answer.\nContext:\n{context}\n\nQ: {query}\nA:"

    inputs = gen_tokenizer(prompt, return_tensors="pt")
    inputs = {k: v.to(generator.device) for k, v in
inputs.items()}
    pad_id = gen_tokenizer.pad_token_id if
gen_tokenizer.pad_token_id is not None else
gen_tokenizer.eos_token_id

    with torch.no_grad():
        output_ids = generator.generate(
            **inputs,
            do_sample=True, temperature=0.7, top_p=0.9,
            max_new_tokens=max_new_tokens,
            pad_token_id=pad_id,
eos_token_id=gen_tokenizer.eos_token_id,
        )

    new_tokens = ____[25]____

    answer = gen_tokenizer.decode(new_tokens,
skip_special_tokens=True).strip()
    return answer

if __name__ == "__main__":
    embedder_name = "../Qwen/Qwen3-Embedding-0.6B"
    retriever_tokenizer =
AutoTokenizer.from_pretrained(embedder_name)
```

```
    embedder = AutoModel.from_pretrained(embedder_name)
    generator_name = "../Qwen/Qwen3-0.6B"
    gen_tokenizer = AutoTokenizer.from_pretrained(generator_name)
    generator =
AutoModelForCausalLM.from_pretrained(generator_name)
    corpus = [
        "Paris is the capital of France.",
        "Tokyo is the capital of Japan.",
        "Beijing is the capital of China.",
    ]
    retriever = Retriever(embedder, retriever_tokenizer, corpus)
    query = "What is the capital of France?"
    ans = rag_answer(query, retriever, generator, gen_tokenizer,
topk=2)
    print("Q:", query)
    print("A:", ans)
```

21. 在方法 `_text_to_vector` 中第一步是将输入的文本批次 text_batch 转换为模型可以理解的格式，以下代码正确的是：

A. self.embedder(text_batch, return_tensors="pt", truncation=True, max_length=512)

B. self.tokenizer(text_batch, return_tensors="pt", truncation=True, max_length=512)

C. self.tokenizer.encode(text_batch, return_tensors="pt", truncation=True, max_length=512)

D. self.embedder(text_batch return_tensors="pt", truncation=True, max_length=512)

22. 此处需要进行前向计算获取隐藏向量，以下代码正确的是：

A. self.embedder(**enc)

B. self.embedder.forward(enc["input_ids"])

C. self.embedder(enc["input_ids"])

D. self.embedder.forward(enc)

23. 此处需要池化得到句向量，以下代码正确的是：

A. last_hidden.mean(1)

B. (last_hidden * attn_mask.unsqueeze(-1)).sum(1) / attn_mask.sum(1, keepdim=True)

C. last_hidden[:, 0, :]

D. (last_hidden * attn_mask).sum(1) / attn_mask.sum(1)

24. 此处需要进行检索得到对应文档，以下代码正确的是：

A. retriever.index.search(query_vec, topk)

B. retriever.search(query_vec, topk=topk)

C. retriever.build_index(query_vec)

D. retriever.generate(query_vec, topk=topk)

25. 此处需要对模型输出进行处理，以下代码正确的是：

A. output_ids[0][-max_new_tokens:]

B. output_ids[0][inputs["input_ids"].shape[1]:]

C. output_ids[:, -max_new_tokens:]

D. output_ids[:, inputs["input_ids"].shape[1]:]

二、参数高效微调

低秩适配方法 LoRA（Low-Rank Adaptation）是一种参数高效微调方法，通过只训练部分低秩矩阵插入至原始模型的关键层，实现大语言模型的高效定制。与传统微调相比，LoRA 显著减少了训练参数量和显存需求，适用于资源受限场景下的大模型下游训练。一个典型训练场景如下：

- 模型：**Qwen-2.5-7B-Instruct**；LoRA 仅插入注意力机制中关键的两个投影层。
- 资源：**单卡 16GB 显存**；输入序列长度上限 **2048**。
- 训练策略：需将全局 batch size 控制在 8 左右，在显存足够的情况下**尽量快**的进行训练。
- 精度：启用半精度训练以降低显存占用。
- 产出：训练完成后，仅保存 LoRA adapter 权重，供下游推理使用。

请阅读以下训练代码，并根据需求完成空缺部分。

```python
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM,
TrainingArguments, Trainer
from peft import LoraConfig, get_peft_model
```

```
model_name = "Qwen/Qwen2.5-7B-Instruct"

# 1) 加载分词器
tokenizer = ____[26]____
model = AutoModelForCausalLM.from_pretrained(
    model_name, device_map="auto", torch_dtype=torch.float16,
trust_remote_code=True
)

# 2) 配置 LoRA
lora_config = LoraConfig(
    ____[27]____
    target_modules=____[28]____,
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM"
)

model = get_peft_model(model, lora_config)

# 3) 定义 Trainer（假设 train_dataset 已经准备好）
training_args = TrainingArguments(
    output_dir="./lora-qwen2.5-7b",
    ____[29]____
    num_train_epochs=3,
    learning_rate=2e-4,
    ____[30]____
    logging_steps=10,
    save_strategy="epoch"
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    tokenizer=tokenizer,
)

trainer.train()

# 4) 保存 LoRA 权重
model.save_pretrained("./lora-qwen2.5-7b-lora")
```

26. Qwen2.5 可能包含自定义分词器/处理逻辑，要求与模型端一致的 remote code。请选择一种能正确加载该模型分词器的方式：

A. AutoTokenizer.from_pretrained(model_name, return_tensors="pt")

B. AutoTokenizer.load(model_name)

C. AutoTokenizer(model_name)

D. AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)

27. 请参考题目描述的计算资源，在尽量保持训练稳定和速度的前提下，选择常见且相对节省显存的 LoRA 配置：

A. r=64, lora_alpha=32,

B. r=8, lora_alpha=64,

C. r=8, lora_alpha=16,

D. r=32, lora_alpha=8,

28. 为了配置 LoRA 权重的注入位置，以下代码更合理的是：

A. ["mlp", "norm"]

B. "all_linear_layers"

C. ["k_proj", "o_proj"]

D. ["q_proj", "v_proj"]

29. 显存限制和内存溢出（Out Of Memory，OOM）风险下，以下配置最稳妥的是：

A. per_device_train_batch_size=2，gradient_accumulation_steps=4,

B. global_train_batch_size=8，gradient_accumulation_steps=4,

C. global_train_batch_size=8，per_device_train_batch_size=1,

D. per_device_train_batch_size=1，gradient_accumulation_steps=8,

30. 为了配置训练的数据类型，以下代码更合理的是：

A. fp32=True,

B. fp16=True,

C. training_type="fp32",

D. training_type="bf16",

//

代码选择题答案解析：

一、检索增强生成（RAG）部分

21. 文本转换为模型输入格式

答案　：B

解析　：在 _text_to_vector 方法中，第一步是将文本批次 text_batch 转换为模型可处理的格式。这里需要调用分词器（tokenizer），因为分词器的作用就是将原始文本转换为模型所需的输入 ID、注意力掩码等张量。

- 选项 B self.tokenizer(text_batch, return_tensors="pt", truncation=True, max_length=512)正确使用了分词器，并设置参数返回 PyTorch 张量、截断和最大长度。
- 选项 A 和 D 错误地使用了 self.embedder（嵌入模型），但模型本身不直接处理原始文本。
- 选项 C 的 self.tokenizer.encode 方法通常返回列表而非张量，而代码中需要字典格式的输入（如后续的 enc 被处理为字典）。

如何通过上下文分析　：

- 查看代码：enc 变量后续被用作 enc = {k: v.to(...) for k, v in enc.items()}，这表明 enc 应该是一个字典（包含 input_ids 等键）。分词器的调用通常返回这种字典结构。
- 对比选项：B 选项直接使用分词器并返回张量字典，而 C 选项的 encode 方法可能返回列表，不匹配上下文。

---

22. 前向计算获取隐藏向量

答案　：A

解析　：此处需要进行模型的前向传播，以获取隐藏状态。嵌入模型（self.embedder）需要接受分词后的输入 enc（一个字典），其中包含 input_ids 和 attention_mask。

- 选项 A self.embedder(**enc)使用字典解包（**enc）将字典的键值对作为关键字参数传入模型，这是 Transformer 模型的标准调用方式。
- 选项 B 和 C 只传入 input_ids，忽略了注意力掩码，会导致错误。
- 选项 D 的 self.embedder.forward(enc)直接传入字典，但 forward 方法通常需要

明确的关键字参数。

如何通过上下文分析 ：

- 查看代码：enc 是一个字典（从上一题可知），而模型前向传播需要同时使用 input_ids 和 attention_mask。
- 对比选项：A 选项的 **enc 能完整传递所有参数，其他选项均缺失部分信息。

---

23. 池化得到句向量

答案 ：B

解析 ：池化的目标是将序列的隐藏状态汇总为一个句向量，并考虑注意力掩码以忽略填充标记。选项 B (last_hidden * attn_mask.unsqueeze(-1)).sum(1) / attn_mask.sum(1, keepdim=True)实现了带掩码的平均池化：

- attn_mask.unsqueeze(-1)将掩码从 [batch_size, seq_len]扩展为 [batch_size, seq_len, 1]，以便与隐藏状态相乘。
- 先对加权隐藏状态求和，再除以有效长度（掩码和），得到正确池化结果。
- 选项 A 是简单平均池化，忽略掩码；选项 C 只取第一个标记（如 CLS），但这里未指定模型类型；选项 D 的掩码未扩展，形状不匹配。

如何通过上下文分析 ：

- 查看代码：attn_mask 被明确定义，说明池化需使用掩码。
- 对比选项：B 是唯一涉及掩码计算的选项，且数学形式完整（扩展掩码并归一化）。

---

24. 检索相关文档

答案 ：B

解析 ：在 rag_answer 函数中，已计算出查询向量 query_vec，需要调用检索器的 search 方法获取相关文档。

- 选项 B retriever.search(query_vec, topk=topk)正确调用检索器的搜索方法，并传入参数。
- 选项 A 的 retriever.index.search 错误（检索器无 index 属性）；选项 C 的 build_index 用于构建索引，非搜索；选项 D 的 generate 不属于检索器方法。

如何通过上下文分析 ：

- 查看代码：Retriever 类中明确定义了 search 方法（def search(self, query_vec, topk=3):），且参数与调用匹配。

- 对比选项：B 直接使用检索器的现有方法，其他选项的方法名或属性不存在于类中。

---

25. 提取新生成令牌

答案 ：B

解析 ：在 rag_answer 函数中，已计算出查询向量 query_vec，需调用检索器的搜索功能获取相关文档。Retriever 类中明确定义了 search 方法，其参数为 query_vec 和 topk，与空缺处的需求完全匹配。选项 B 直接调用该方法，并传入参数。

A 的 retriever.index.search 错误（类中无 index 属性）；C 的 build_index 用于构建索引，非搜索；D 的 generate 不属于检索器方法。

如何通过上下文分析 ：

1. 查看类的方法列表：Retriever 类中只有 search 方法具备检索功能，其他选项的方法名在类中未定义。
2. 匹配参数：search 方法的定义是 def search(self, query_vec, topk=3)，与调用时的 query_vec 和 topk 参数一致。

二、参数高效微调（LoRA）部分

26. 加载分词器

答案 ：D

解析 ：Qwen2.5 模型可能包含自定义代码，因此加载分词器需与模型一致，设置 trust_remote_code=True。

- 选项 D AutoTokenizer.from_pretrained(model_name, trust_remote_code=True) 正确启用远程代码信任。
- 选项 A 的 return_tensors="pt"是编码文本时的参数，非加载时分词器所需；选项 B 和 C 语法错误。

如何通过上下文分析 ：

- 查看代码：模型加载时使用了 trust_remote_code=True，因此分词器也需相同设置以保持一致性。
- 对比选项：D 是唯一包含 trust_remote_code 的选项。

---

27. 选择 LoRA 配置

答案 ：C

解析 ：LoRA 配置需在训练稳定性和显存效率间平衡。资源受限（16GB 显存）下，较小的秩（r）更节省显存。

- 选项 C r=8, lora_alpha=16 是常见配置（r=8 时 alpha 常设为 16 或 32），平衡效果与显存。
- 选项 A 的 r=64 过大，显存占用高；选项 B 的 alpha=64（r 的 8 倍）可能不稳定；选项 D 的 r=32 仍较大。

如何通过上下文分析 ：

- 查看题目描述：强调“资源受限”和“节省显存”，暗示应选较小 r。
- 对比选项：C 的 r 值最小，且 alpha/r=2，符合常见实践。

---

28. 配置 LoRA 注入位置

答案 ：D

解析 ：题目指出 LoRA 仅插入注意力机制中的“关键投影层”。Transformer 注意力模块包含查询（q_proj）、键（k_proj）、值（v_proj）、输出（o_proj）投影层，通常选择 q_proj 和 v_proj。

- 选项 D ["q_proj", "v_proj"]是标准注入目标。
- 选项 A 和 B 针对非注意力模块；选项 C 的 k_proj 和 o_proj 较少见。

如何通过上下文分析 ：

- 查看题目描述：“注意力机制中关键的两个投影层”对应查询和值投影。
- 对比选项：D 直接匹配注意力机制的关键层。

---

29. 配置批量大小与梯度累积

答案 ：D

解析 ：在显存限制下，需通过梯度累积实现全局 batch_size=8，同时避免 OOM。

- 选项 D per_device_train_batch_size=1, gradient_accumulation_steps=8 使全局 batch=1×8=8，且单步 batch 较小，最稳妥。
- 选项 A 的 per_device=2 可能显存不足；选项 B 和 C 使用了不存在的参数（如 global_train_batch_size）。

如何通过上下文分析 ：

- 查看题目：要求全局 batch=8，且“显存限制下最稳妥”。
- 对比选项：D 的 per_device=1 显存占用最低，而其他选项参数错误或风险更高。

30. 配置训练数据类型

答案 ：B

解析 ：题目要求启用半精度训练以降低显存。TrainingArguments 中，fp16=True 启用 FP16 训练。

- 选项 B fp16=True 正确。
- 选项 A 的 fp32=True 会禁用半精度；选项 C 和 D 的参数名不存在。

如何通过上下文分析 ：

- 查看代码：模型加载时已设置 torch_dtype=torch.float16，训练需一致。
- 对比选项：B 是标准参数，其他选项语法错误或矛盾。

//

答案

单选题:

- 1-5 ACCBC
- 6-10 CBBCB
- 11-15 CBADD
- 16-20 CABCB

程序题:

- 21-25 BABBB
- 26-30 DCDDB