基于 DeepSeek-Coder-1.3B 和 QLoRA 构建的 SQuAD 2.0 问答系统

项目背景和目标

本项目旨在构建一个基于 **SQuAD 2.0** 数据集的问答系统，通过实践来学习问答系统的关键技术和实现方法。问答系统领域涵盖**单文档问答**（针对单一文本片段进行阅读理解）、**多文档问答**（需要跨多个文档检索信息来回答问题）以及**机器阅读理解**等核心技术方向。SQuAD 2.0 数据集提供了一个典型的机器阅读理解任务场景，要求模型从给定段落中找出问题答案，或在答案不存在时输出“无答案”。通过本项目，我将深入研究这些技术，包括答案抽取、不确定性判断和上下文检索等，并将所学成果应用于实际系统构建。

项目的最终目标是训练出一个高性能的问答模型，并通过 **API 接口**部署这一模型服务。这样，用户或其他应用程序可以通过 API 提交问题及相关上下文，获取模型返回的答案。该系统将作为个人求职展示的技术项目，证明我对问答系统构建流程的理解和掌握，展现解决实际 NLP 问题的能力。简而言之，本项目既有学习研究的意义，也会产出一个可演示的问答服务，为求职或技术分享提供支撑。

数据收集与处理

本项目采用了 **Stanford SQuAD 2.0** 英文数据集作为训练语料。SQuAD 2.0 是斯坦福推出的阅读理解问答数据集，包含由人工标注的问题-答案对。与早期版本 SQuAD 1.1 只包含有答案的问题不同，SQuAD 2.0 在其中引入了**无法回答的问题**。具体来说，SQuAD 2.0 将原有约 10 万条可回答的问题与**超过 5 万条不可回答的问题**合并在一起。这些不可回答问题是由众包工作人员刻意设计的、基于段落内容无法找到答案的问题，其形式看起来与可回答问题相似。模型不仅需要在段落中找到答案，还必须学会判断**段落中不包含答案的情况并选择不作答**。这一特性使数据集更具挑战性，迫使问答系统“知晓自己不知道”的情况。

在数据处理方面，我进行了如下步骤：

1. **数据筛选与格式转换**：由于专注于英文问答，本项目**仅使用英文内容**的 SQuAD 2.0 数据集。首先将官方提供的 JSON 格式数据转换为训练所需的文本格式。每条训练样本包含“上下文段落”、“问题”和“答案”三部分。其中“答案”要么是段落中的一个文本片段，要么在无答案情况下需要输出特殊标记。
2. **引入 <NO_ANSWER> 标记**：针对 SQuAD 2.0 中不可回答的问题，我在数据集中引入了一个特殊答案标记 <NO_ANSWER> 来表示“无答案”。具体做法是在构造训练样本时，如果某个问题在段落中没有答案，则将该问题对应的答案标注为 <NO_ANSWER>。这一特殊标记被添加到模型的词表中（如果模型原始词表

中不存在），以确保模型可以生成该标记。通过明确地提供 <NO_ANSWER>，模型在训练时能够学习何时应当输出这一标记来表示“无法在文本中找到答案”。

3. **数据均衡采样**：由于 SQuAD 2.0 训练集中**可回答问题和不可回答问题的比例不平衡**（训练集中可回答问题数量约为不可回答问题的两倍，我采用了均衡采样策略。在每一训练批次（batch）中，我**随机采样近似相等数量的可回答和不可回答问题**用于训练。通过这种过采样/欠采样的方法，保证模型在训练过程中既看到足够多的有答案示例，也看到足够多的无答案示例，从而减小数据偏差的影响。均衡采样有助于模型更好地学习何时应该输出 <NO_ANSWER>，避免因训练集中无答案样本相对较少而导致模型倾向于**总是尝试给出答案**。经过上述处理，我得到了格式统一且平衡的训练数据集，并保留了一部分开发集数据用于模型评估。

模型选择与训练方案

模型选择：DeepSeek-Coder-1.3B

我选择 **DeepSeek-Coder-1.3B** 作为基础语言模型。DeepSeek-Coder 是一系列开源的代码语言模型，涵盖从 1B 到 33B 不等的多种规模。其中，1.3B 参数模型是该系列中规模较小的一款。DeepSeek-Coder 模型在海量混合语料上预训练了 2 万亿个 token，其中包含约 87%的代码和 13%的自然语言（英文和中文）内容。虽然以代码数据为主，但它也掌握了一定程度的英文自然语言理解能力。另外，DeepSeek-Coder 提供了**长达 16K tokens** 的上下文窗口，这使它能够处理较长的文档，这对于问答任务中可能需要处理长段落甚至多文档的情况非常有利。

之所以选择 1.3B 模型，一方面是考虑到**模型能力与资源的平衡**：1.3B 参数的模型在保证基本语言理解和推理能力的同时，计算和显存开销相对可控，适合在有限的硬件资源上进行训练和部署。另一方面，DeepSeek-Coder 系列采用了开放许可，允许研究和商用，这使我能够自由地微调和应用该模型而无需担心授权问题。此外，代码预训练背景可能赋予模型较强的逻辑推理能力，这对于解答需要推理的问题也有潜在益处。综合考虑，这款模型满足我在性能、资源和许可方面的项目要求。

微调方案：QLoRA 高效微调

在模型微调过程中，我采用了 **QLoRA**（Quantized Low-Rank Adapter）技术对 DeepSeek-Coder-1.3B 进行高效微调。QLoRA 是一种**参数高效微调**方法：首先将预训练模型权重量化为 4-bit 低精度表示，然后在模型中引入少量可训练的低秩适配器权重（LoRA 层），仅微调这些新增参数。这种方法大幅降低了显存占用，使我能够在单张 GPU 上对大模型进行微调，同时仍能保持与全精度微调相近的效果。事实上，QLoRA 技术曾成功实现在单 GPU 上微调 65B 参数模型而性能几乎不下降，足见其对内存优化的效率。本项目虽然使用的模型规模较小（1.3B），但采用 QLoRA 依然有诸多益

处：一是可以进一步降低训练资源需求（例如更小的 batch 也能跑，更快的迭代），二是为将来微调更大模型掌握方法。使用 QLoRA，我将 DeepSeek-Coder-1.3B 以 4-bit 精度加载，并冻结其原有所有权重，仅针对 LoRA 适配器的权重进行梯度更新。

训练流程与参数控制：在微调过程中，预训练模型的原始参数保持冻结状态，我**仅训练 LoRA 适配器层的参数**。这样有效控制了需要更新的参数总量，微调过程中训练的参数仅占模型总参数的极小一部分（例如不到 1%）。这不仅降低了过拟合风险，也缩短了训练收敛时间。我设定了适当的超参数来保证训练稳定性，例如选用了**适中的学习率**（避免量化下训练不稳定）、采用梯度累积来增大等效批大小等策略，并使用了 AdamW 优化器的 8-bit 优化实现（来自 bitsandbytes 库）以配合 4-bit 量化存储，进一步减少内存开销。整个微调训练在单张 GPU 上完成，我针对 SQuAD 2.0 训练集进行了若干个 epoch 的训练，在每个 epoch 后使用开发集计算指标监控模型性能。训练数据格式上，我将每个样本构造成一种指令式的 QA 输入形式：**将上下文段落和问题拼接作为模型输入，期望模型生成对应的答案**。例如输入形式可以是：“<Context>...[段落文本]...<Question>...[问题]...<Answer>...”，模型需要根据提供的段落回答问题或者输出<NO_ANSWER>。这种格式使模型明确定义了任务：给定一段文章和其中的问题，输出正确答案文本。如果问题无解，则输出预先约定的<NO_ANSWER>标记。通过这种监督微调，模型逐步学会在阅读提供的文本后生成正确的答案或判断无答案。

训练中我还运用了**提前停止（early stopping）**等策略：当开发集上的指标不再提升时，我停止训练并选取表现最佳的 checkpoint 作为最终模型。这确保了模型不过度拟合训练集，同时在验证集上保持良好的泛化性能。综合以上方案，我成功地在相对有限的资源下完成了对 DeepSeek-Coder-1.3B 模型在 SQuAD 2.0 数据上的高效微调，为下游问答系统部署打下基础。

评估指标与结果

在模型评估阶段，我主要采用了问答任务常用的几项指标来衡量模型表现，包括：**精确匹配率（EM）**、**F1 值**以及 **NoAns_exact** 等。其中，EM（Exact Match）表示模型给出的答案与标准答案**完全匹配**的比例；F1 值是根据预测答案与标准答案的重叠词语计算出的调和平均分，综合考虑了预测的**准确率和召回率**，即即便模型的答案不与标准答案一字不差，也能通过正确重叠的部分获得一定分数；**NoAns_exact** 则专门衡量模型在无答案问题上的判断准确率，即对于那些正确答案应为空的问题，模型预测“无答案”与标准答案一致的比例。

我分别在模型微调前后对上述指标进行了评估对比。初始未微调模型（或未经过优化处理的模型）在 SQuAD 2.0 开发集上的成绩表现较弱，**F1 值仅约为 22.4%**，而对于无答案类问题几乎**从未给出正确的空答案**（NoAns_exact ≈ 0%）。这意味着原始模型面对

SQuAD 2.0 任务时，大部分问题都无法正确提取答案，尤其对不可回答的问题总是试图给出某个答案，完全无法判断何时应该不作答。

经过上述微调训练和数据增强策略后，模型的问答性能有了显著提升：**F1 值提高到了约 49.8%**，整整提升了两倍有余；同时 **NoAns_exact 提升至 44.4%**。F1 从 22.4%跃升至 49.8%，表明微调极大增强了模型从文本中找出正确答案的能力，模型现在能够在一半左右的问题上找到与标准答案高度重合的回答。而 NoAns_exact 从 0 增加到 44.4%，意味着在将近一半的无答案测试样本中，模型学会了正确地输出 <NO_ANSWER>来表示无解——先前模型总是妄图作答，现在经过训练能有选择地**拒绝回答无法回答的问题**了。这一点对于问答系统的可靠性具有重大意义：模型具备了一定的“不知道就不乱答”的判断能力，大幅降低了提供错误答案的几率。尽管我提升后的 F1 约 50%在绝对值上仍低于当前该数据集最佳模型（学界顶尖模型 F1 可达约 66%，人类表现 F1 约 89%），但我的模型已接近于一个强基线水平（例如**始终不作答**的模型在测试集也有约 48.9%的 F1）。综合来看，这次微调使模型无论在**回答准确性**还是**拒答无效问题**两个方面都有了长足进步，为实际应用打下了更坚实的基础。

系统部署方式

完成模型训练和评估后，我将微调后的问答模型通过 **API 接口**的形式进行了部署。具体来说，我搭建了一个后端服务，将加载了微调模型的推理引擎封装为可远程调用的 RESTful API。客户端（例如网页前端、移动应用或其他系统）可以向该 API 发送包含**问题和相关上下文**的请求，服务器端的模型会处理该请求并返回答案结果。在遇到无答案的情况时，模型会返回一个预定义的表示（如字符串“<NO_ANSWER>”或空答案），客户端据此可以做出相应处理。

这种基于 API 的部署方式使模型易于集成到各种应用场景中。例如：

- **阅读理解服务**：应用于教育或内容平台，提供文章和问题即可通过 API 获得答案，实现自动化的阅读理解测验讲解功能。
- **客服问答系统**：结合企业自己的知识库，将用户提问发送给模型，并由模型从知识库内容中找出答案或判断无解。这可以用来构建智能客服或知识问答助理，为客户提供自动答疑和推荐相关问题解答，提高客服效率。
- **通用问答系统**：作为基础的问答引擎，供聊天机器人、智能音箱等使用。当用户提出事实类问题时，系统可以检索相关文本（如 Wiki 片段）提供给模型，通过 API 获得准确答案并反馈给用户。

在部署过程中，我对模型推理进行了优化，例如加载了经过量化后的权重以减少内存占用，并使用异步请求和缓存机制来提高并发处理能力。整个系统具有良好的可扩展性，可以根据需求扩容模型实例来支撑更多请求。通过以 API 形式提供服务，本项目

的问答模型得以在真实应用环境中被调用演示，这非常适合作为求职展示的项目成果——面试者可以方便地通过 API 体验模型的问答能力，验证我的技术实现。

持续优化路径

尽管目前系统已经基本成型并达到预期目标，我也规划了未来可以进一步提升系统性能和效果的几条优化路径。这些优化尚未实施，但具有潜在的高价值：

- **基于人类反馈的强化学习（RLHF）**：引入人类反馈来优化模型回答质量。具体做法是收集模型回答的人工评价数据，然后通过强化学习调整模型，使其更符合人类偏好。例如，奖励准确、详尽的回答，惩罚含糊或不恰当的回答。RLHF 有望提高模型回答的可靠性和有用性，并让模型学会更好地拒绝不合理的问题或在不确定时表明无答案。

- **知识增强**：结合**检索增强生成（RAG）**等技术，将外部知识库或搜索引擎接入模型。在模型回答问题之前，先根据问题检索相关资料片段供模型参考。通过将最新的知识提供给模型，模型可以回答超出训练语料范围的问题并提高准确率。这对于实时问答或专业领域问答非常重要，避免模型因训练语料有限而产生幻觉答案。

- **推理加速**：优化模型的推理效率，降低响应延迟。例如，采用更高级的模型量化（如 8-bit/4-bit 推理）和稀疏化技术，或使用专门的推理引擎和硬件加速（如 TensorRT、ONNX Runtime 等）来提高吞吐量。推理加速能够让问答系统在实际使用中更流畅地响应用户提问，并降低部署成本（例如节省 GPU 计算资源）。对于需要大规模并发访问的应用，推理速度的提升是关键优化方向。

- **模型蒸馏**：利用模型蒸馏技术，将当前微调后的大型模型的知识迁移到一个较小的学生模型中。通过让小模型学习复现大模型的回答行为，我可以得到一个参数量更小、推理速度更快的模型，而其问答性能与大模型相近。蒸馏后的模型更适合部署在资源受限的环境中（比如移动端或嵌入式设备），同时在服务器端也能大幅降低内存和计算消耗。这将提高系统的**可部署性**和**成本效益**。

- **提示优化**：进一步改进模型的提示词设计（Prompt Engineering）。我可以探索最佳的提示模板或上下文格式，引导模型产出更准确的答案。例如，在问题前后加入明确的指令性描述，或者在输入中加入 few-shot 示例来示范如何回答。此外，也可以考虑对提示进行自动化优化（Prompt Tuning 或 PEFT 方法），直接微调出一组优化的连续向量作为提示，从而在不改动模型权重的情况下提升回答质量。良好的提示能够让模型更充分地理解问题意图，避免歧义，并控制回答风格，使系统回答更符合用户预期。

除了上述方向，后续还可以考虑引入多语言支持、利用更大规模数据继续预训练等手段。但总体而言，这些持续优化路径将围绕**提升模型能力、增强系统实用性和提高运行效率**三个方面展开。随着这些优化的逐步实施，本问答系统有望在准确率、更广知识覆盖以及交互体验上更上一层楼，为用户提供更出色的问答服务。在未来的项目迭代中，我将根据实际需求和资源，选择其中一项或多项方案进行深入研究和实现，不断完善本系统，巩固其作为求职技术展示项目的亮点。