ChatGPT

基于Agent机制的大语言模型解题能力评估系统设计

系统总体架构概览

本系统旨在评估多个大语言模型（如GPT-4、Anthropic Claude、Google Gemini等）在英语完形填空和阅读理解选择题上的解题能力。系统采用多Agent架构，将整个流程划分为题目生成、模型解题、答案评分和统计分析等模块，由中心调度流程串联各模块工作 [1] [2]。各模块各司其职，协作完成从原始PDF内容到评估报告的自动化流程：

- **题目生成Agent**从用户上传的英文PDF材料中自动抽取内容并生成不同难度的完形填空题和阅读理解选择题（包含标准答案）。
- **模型解题Agent**将生成的题目分别发送给多个大语言模型求解，并要求模型给出思维链条（Chain-of-Thought）推理过程和最终答案。
- **答案评分Agent**（评估模型）读取标准答案与各模型的作答及其推理过程，对答案正确性、语言逻辑性和推理合理性进行评分。
- **统计分析模块**汇总各模型在各类题目上的表现指标，包括正确率、解题耗时、思维链条准确性以及不同难度下的适应性表现，并生成评估报告。

这种模块化多Agent设计遵循单一职责原则，每个Agent专注完成特定任务，既提高协同效率又便于系统扩展 [1] [2]。下面将详细阐述各模块功能及实现方案。

模块设计与功能

1. 题目生成模块

功能描述： 题目生成模块负责从上传的PDF英文原文中自动生成练习题，包括完形填空和阅读理解选择题，并附带参考答案。该模块需要能够根据内容领域和语言难度，生成多样且不同难度等级的问题。

工作流程： 首先，通过PDF解析工具读取文档内容，将长文按段落或章节拆分处理（例如使用LangChain提供的PDF Loader和文本分割器，将PDF按长度或语义分块 [3]）。然后由**题目生成Agent**对每个内容块生成题目：

- **完形填空题生成：** Agent基于段落内容选取关键句子或单词，生成填空题干和多个选项（包含正确答案和若干干扰项）。可以采用大语言模型自身的语言理解来选取易考查的单词或短语作为空格，并让模型提供三个干扰选项和一个正确选项。研究表明利用LLM（如GPT-3.5）自动生成完形填空题是可行且有效的；例如，有方法通过预处理词汇列表、用GPT生成含该词的句子和候选项，再筛选选项来生成填空题 [4]。LLM还能自动生成高质量的干扰项，并支持根据学习者水平调整题目难度 [5] [6]。为了控制难度，Agent在提示词中可以明确要求“生成简单/中等/困难的完形填空题”，或调节所挖空词汇的复杂度和干扰项的相似性。生成结果包含题干（句子或段落，带空格）、选项列表以及正确答案标注。

- **阅读理解题生成：** Agent从PDF内容中选取段落生成阅读理解题。可先让LLM对段落主题生成若干问题，再对每个问题生成四个选项（其中一个正确，其余为干扰项），同时提供正确答案及解析。为了区分难度，Agent可针对段落内容提出不同层次的问题：**容易**的题目侧重段落中的直接事实细节，**中等**难度的题目涉及段落主旨或简单推理，**困难**题目则需要综合段落信息进行深层推断或理解隐含含义。LLM强大的问答生成能力可用于此步骤——例如使用LangChain的QAGenerationChain可以自动为文档生成

若干问答对 [7] [8]。在提示设计上，可以直接要求模型“阅读以下段落并出一道四选一阅读理解题，提供题干、选项和正确答案”。模型会输出问题、选项和答案，将其保存下来备用。

输出格式： 题目生成模块将输出结构化的题目集合。例如每道题包括题目类型（完形填空/阅读理解）、难度标签、题干文本、选项列表及参考答案。为方便后续处理，可采用JSON或Python数据结构存储题目信息。例如：

```
{
 "type": "cloze",
 "difficulty": "easy",
 "question": "The cat ___ on the mat.",
 "options": ["sits", "sat", "sit", "sitting"],
 "answer": "sat"
}
```

类似地，阅读理解题包含段落或问题描述、4个选项和正确答案标记。通过结构化输出，可以确保后续解题Agent准确提取所需信息。

2. 模型解题模块

功能描述： 模型解题模块将题目逐一道给各大语言模型，让它们作答，并记录作答内容，包括最终答案和中间推理过程（思维链条）。该模块旨在比较不同模型对同一题目的解题能力和解题方式。

工作流程： 系统针对每道题，依次调用多个候选的大语言模型API，包括但不限于OpenAI GPT-4、Anthropic Claude、Google Gemini等。可以采用并行或异步调用提高效率，但需确保记录各模型的响应时间用于耗时统计。为获取模型的**Chain-of-Thought**推理过程，提示设计上应引导模型进行逐步推理并给出最终答案 [9] [10]。例如，对每个模型，可以构造如下通用提示（Prompt）：

```
请仔细阅读以下题目并给出解答。在给出最终答案前，请展示你的推理过程。
题目：<题干内容>
选项：<选项列表>（如有）
请先进行分析，然后给出答案。
```

通过明确要求模型“展示推理过程”，大多数LLM会先输出若干推理步骤，然后再给出答案。这种**思维链条提示**已被证明可以显著提升模型在复杂任务上的表现 [11]。例如Google的研究指出，引导模型逐步推理能够大幅提高解决推理题的准确性 [11]。同时，思维链条使模型的决策过程对用户可见，提高了解释性 [10]。

对于完形填空题，模型可能按顺序分析每个选项与空格的匹配程度；对于阅读理解题，模型会先总结段落关键信息，再逐一评估各选项与段落内容的吻合度。模型最后输出其认为的正确答案（若模型直接给出完整解答，包括推理和答案，需要从文本中解析提取最终选择的选项）。

思维过程记录： 系统对每个模型的输出都做保存，包括完整的推理文本和最终答案。可以将模型回答解析为一个含有 `reasoning` 和 `answer` 字段的结构。例如：

```
{
 "model": "GPT-4",
 "question_id": 1,
```

```
  "reasoning": "这是一道完形填空题...（此处省略模型详细推理过程）...因此我选择B。",
  "answer": "B",
  "time_used": 2.31
}
```

这里`time_used`表示该模型回答此题所用时间（秒）。通过记录时间戳计算差异，可以获取**解题耗时**指标。

多模型接口调用： 不同模型可能有不同的API端点，但可以通过统一接口封装。例如使用OpenAI的SDK调用GPT-4，通过Anthropic API调用Claude，或通过Google PaLM2接口调用Gemini等。LangChain等框架可以封装不同模型，使调用格式统一，从而方便切换模型后端。也可以直接使用OpenAI接口管理，由调用时指定`model=GPT-4`或`Claude`等（若Claude、Gemini提供兼容OpenAI API的接口）。无论实现方式如何，**模型解题Agent**在架构上可以被视为多个子Agent的集合：每个子Agent对应一个待评估模型，负责接受题目并返回该模型的解答。中心调度模块遍历题目集合，对每个题逐次激活所有模型Agent并收集它们的回答与推理。

3. 答案评分模块

功能描述： 答案评分模块利用另一个强大的评估模型（如GPT-4或专门微调的评分模型）作为**评分Agent**，自动比对各大模型的答案和推理过程，给出评分和评价。评分主要依据标准答案，同时考虑回答的语言逻辑性和思维过程的合理性。

评分策略： 为了实现自动、客观的评价，我们采用**LLM-as-a-judge（以LLM为裁判）**的方法 [12] [13]。具体来说，就是使用一个被公认为性能最强、较为中立的模型（如GPT-4）来充当评审官。通过精心设计提示词，引导该评审Agent根据预定标准对候选模型的作答进行打分 [13]。研究表明，利用强大的LLM作为评审，可以大规模地对模型输出进行可靠的自动评估，但前提是要明确评分标准并精心撰写提示 [13]。

评分标准： 评分Agent的提示需要包括明确的评价维度和评分尺度。例如，可以制定如下几个维度： - **正确性（Accuracy）：**模型作答与标准答案的匹配程度。完全正确得满分，部分正确或有知识性错误则减分，错误答案为零分 [14]。 - **语言逻辑性（Coherence）：**回答是否语言表述清晰、有逻辑，是否准确围绕问题作答，是否有自相矛盾或语义混乱之处。 - **思维合理性（Reasoning Soundness）：**模型提供的思维链条是否合理、有据可依。考察其推理过程是否正确引用了题目信息、推理步骤是否严谨有效，是否存在不合理的跳步或谬误。

评分Agent可以针对每道题每个模型的回答给出上述每项的评价分数或等第，并给出一段简短的评语解释。例如提示可以这样设计：

```
你是一名严格的英语试题阅卷官。现在将对比标准答案和考生答案，并根据以下标准评分：
1. 正确性：答案是否正确或符合标准答案。
2. 语言逻辑性：答案的表述是否清楚、逻辑严谨、紧扣问题。
3. 思维合理性：考生的解题思路是否合乎情理，推理过程是否正确。
请对下面提供的内容进行评估并给出每项的评分（满分10），同时给出简短评价。

题目：<题干/段落及问题>
标准答案：<参考答案>
考生答案：<模型给出的最终答案>
考生思维过程：<模型的推理过程>
```

通过上述提示，评分Agent会阅读题目、标准答案和模型回答，然后从正确性、逻辑性、思维合理性三方面进行分析打分。例如，它可能输出：

```
正确性：10/10 – 模型答案与标准答案完全一致。 [14]

语言逻辑性：9/10 – 答案用词准确，表述清晰有条理。

思维合理性：8/10 – 推理过程大体合理，但有一步对细节的解释略显冗长。

综合评价：该作答准确回答了问题，语言表达清晰且推理过程基本正确。
```

上述输出中引用了模型推理的情况并给出评语（以上为示例格式）。实际实现中，评分Agent可直接输出机器可读的评分结构，例如JSON：

```
{
 "accuracy": 10,
 "coherence": 9,
 "reasoning": 8,
 "comments": "答案正确，语言表达清晰，推理过程合理但略有冗余。"
}
```

模型选择： 评分Agent应选择一个擅长评估和理解的模型。如GPT-4由于其强大的理解和比较能力，非常适合作为裁判模型 [12] [13]。OpenAI近期提出的GPT-4评估方法（如在MT-Bench等基准中使用GPT-4进行模型比较打分）也证明了这一思路的有效性。除了GPT-4，也可考虑专门微调的评估模型或者Anthropic Claude (因其擅长长文本分析)作为备选。评分Agent的角色是统一的，不管被评估的是哪个模型的答案，它都用相同标准进行打分，从而保证公平性和一致性。

Chain-of-Thought辅助评分： 为提高评分的准确和透明性，可以让评审Agent也采用Chain-of-Thought方式进行评估。例如使用类似G-Eval的方法，要求评审模型在给出最终分数前也“逐条分析理由” [15]。这样不仅能让我们看到评分模型的思路，也有助于它更严谨地对照标准答案和考生思路来评判。这种链式评估方法已被用于细粒度地测试模型输出在特定标准上的表现 [15]。当然，在最终报告中可以选择不展示评分Agent的内部思路，仅保留分数和评语给用户查看。

4. 统计分析模块

功能描述： 统计分析模块汇总整理所有题目在各模型上的作答结果和评分结果，计算整体统计指标，生成对比分析报告。此模块是评估的总结阶段，提供每个模型全面的表现概览。

数据收集： 从模型解题模块与评分模块获取以下原始数据： - 各模型对每道题的答案是否正确（可由评分模块的正确性评分或直接比较标准答案得出二元对错结果）。 - 各模型对每道题的详细评分（三项评分和评语）。 - 各模型回答每题所用时间。 - 题目元信息：类型（完形/阅读）、难度等级等。

指标统计： - **总体正确率：** 统计每个模型在全部题目中的正确率 = 正确作答数量 / 总题目数量。可以细分为完形填空部分正确率和阅读理解部分正确率，以及不同难度题目的分项正确率。例如，生成一个表格列出模型A、B、C在简单、中等、困难题目的正确率分别是多少，以衡量**难度适应性**：模型是否能在高难度题上仍保持不错的表现，抑或表现随难度显著下降。 - **平均评分：** 将评分模块给出的各维度分数取平均。例如每模型平均正确性得分、平均语言逻辑性得分、平均思维合理性得分。此指标可弥补仅看正确率的不全面之处，尤其当有的答案虽正确但表述不佳、有的答案错误但思路合理等情况，评分能体现出差异。 - **解题耗时：** 计算每个模型从接收问题到给出答案的平均耗时，以及分布情况（例如最快和最慢的题）。这反映模型推理效率和稳定性。可以用柱状图比较不同模型平均响应时间，也可给出耗时统计表。 - **思维链条质量：** 由于我们保存了模型的推理过程文本，可进一步分析**思维链条的准确性**。一种量化方式是统计各模型在推理过程中出现错误推断或不合理步

骤的频率。例如，利用评分Agent曾给出的“思维合理性”评分的平均值来衡量；或者开发简单规则，如检查模型推理中是否有自相矛盾或与事实不符的陈述。我们也可以人工抽样检查一些思维链条以佐证自动评分。最终可以报告例如“模型A思维链条平均得分8.5/10，模型B为7.2/10”，并总结常见的思维问题（如模型B经常过度泛化或忽略段落细节）。 - **难度适应性分析：** 对比各模型在不同难度题目的表现差异。例如绘制折线图，横轴为题目难度（易、中、难），纵轴为正确率或平均评分，各模型各一条线。从中可以看出模型随题目难度变化的性能曲线。如果某模型曲线陡降，表示其在高难度时失分严重；如某模型曲线相对平稳，则适应性更好。这一指标对教育场景很重要，可揭示模型在简单练习和复杂推理下能力的稳定性。

报告生成： 统计分析模块将上述分析结果编撰成**评估报告**。报告内容包括： - **总体表现概览：** 文字总结各模型的总体得分、排名，性能亮点和不足。 - **表格数据：** 列出每个模型的关键指标，如正确率、平均分、平均耗时等。 - **图表可视化：** 辅助展示，例如正确率对比柱状图、耗时折线图、难度-正确率曲线图等，以方便读者直观比较。 - **示例案例：** 报告中可选取一些典型题目，展示各模型的回答和评分评语，帮助读者理解评分细节。例如某难题上GPT-4答对且逻辑清晰，Claude答错并产生不合理推理，这些案例可以附上简短分析。

报告可生成为PDF或Markdown文档，便于课程作业提交或进一步发布。所有引用的数据来源模块都应在报告中标明，如某模型特别快就指出测得的时间等。通过详尽的统计分析和直观的报告呈现，读者能够清晰地了解各大模型在所生成测试集上的表现差异。

5. 批量处理支持（可选）

为了提高评估效率和适应更多场景，系统可以扩展支持**批量导入多份PDF**并自动生成综合评估报告。批量模式下，系统按以下方式工作： - 用户一次性提供多份PDF文件（例如不同章节的教材、不同领域的文章等）。可以通过接口上传多个文件或指定一个目录批处理。 - **题目生成模块**针对每份PDF分别生成题目集，并可以在生成时记录PDF来源。也可以选择将所有PDF内容合并后统一抽题，但为了保持多样性和独立性，通常分别生成各自的题目更清晰。 - **模型解题模块**对每个PDF生成的题集分别进行解答。可以并行处理不同PDF的题目，以节省整体时间。需要注意对不同来源PDF，可能题目难度分布不同，可在结果中分别统计。 - **答案评分模块**逐题评分不变，但在标识上可注明题目来源PDF，以便后续按来源统计。 - **统计分析模块**既可以对每份PDF单独形成报告，也可以汇总所有数据形成综合报告。例如对于每份PDF生成一个子报告章节，给出该部分内容上的模型对比结果，然后再有一个全局总结比较不同领域内容上模型性能差异。如果希望生成一份综合报告，可以增加对不同资料来源的比较（例如模型在科学类文章上的正确率 vs 在文学类文章上的正确率）。

批量处理对系统的扩展要求主要在调度和数据区分上。技术上需要确保内存管理和过程控制，因为多PDF批处理可能生成大量题目和调用众多模型API。可以在模块之间增加标识（如source_id）来追踪哪个PDF来源。此外，可通过多线程或多进程提升处理速度，同时在报告阶段合并结果。

这个可选功能使系统具备大规模评测能力，一次运行即可对多个数据集上的模型性能进行评估，为研究者或教育工作者提供更丰富的比较维度。

Agent机制与技术实现方案

多Agent架构优势： 本系统采用Agent机制将复杂任务分解为多个智能体协作完成，每个Agent有清晰分工，使系统更具模块化和可扩展性 [1] [2] 。通过多Agent协作，复杂任务得以分步完成，Agents之间可通过共享内存或消息传递交流信息，从而达到**团队协作**的效果 [1] 。这种架构具备良好的伸缩性：添加新的模型Agent或新的评分维度Agent不会影响其他部分，实现**松耦合**设计 [2] 。

Agent角色与配置： 本系统涉及如下几类Agent，每类有各自的配置和所用模型： 1. **题目生成Agent：** 使用擅长内容理解和生成的模型（如GPT-4）执行。配置侧重于Prompt设计，需在系统信息里定义Agent身份（如：“你是一个考试出题专家”），并提供明确指令生成指定数量、类型和难度的问题及答案。技术实现上，可借助LangChain的LLMChain或ConversationalAgent包装GPT-4调用，将PDF内容作为输入，Prompt模板包含生成要求，最终返回结构化结果。Agent需具备一定**工具使用**能力，例如如果需要根据PDF长文本逐段生

成，可以让Agent调用一个“摘要工具”先概括，再出题；但通常直接输入段落让模型出题更直接。 2. **模型解题Agent（求解Agent）：** 其实包含多个子Agent，每个子Agent绑定一个待测的大模型接口。例如我们配置一个Agent使用GPT-4（temperature适中以保证稳定输出），另一个Agent使用Claude-instant等。每个解题Agent的系统提示可简单设为“你是一个英语测验答题助手”，主要区别在于调用不同底层模型API。我们可以利用**LangChain的Agent机制**，为每个模型创建一个Pseudo-Agent，通过设置`llm=ChatOpenAI(model="gpt-4")`或`ChatAnthropic(model="claude-v1")`等方式来对应不同模型实例。调度时，轮流激活各Agent处理当前题目。由于只是回答问题，不需要复杂工具调用，所以这些Agent可使用简单的LLMChain而非React Agent。核心在于统一的提示模板，以保证不同模型收到完全相同的问题表述，从而比较输出差异。 3. **答案评分Agent（评审Agent）：** 选用评价能力最强的GPT-4作为底层模型。配置的System Prompt明确其身份：“你是一个严格公正的自动阅卷官”，再给定评分标准和格式要求。这个Agent可能需要处理较长输入（包括题干、标准答案、模型答案和模型推理），因此应选择上下文长度大的模型（GPT-4-32k或Claude-100k等）以防截断。LangChain可用ChatOpenAI实例化GPT-4，或直接通过OpenAI API调用。为确保一致性，评分Agent的temperature应设为0以减少随机性，从而保证评分可重复。
实现细节： 可以设计评分Agent为一个Tool或Chain，使其能够方便地被调度模块调用，对给定答案进行评估。LangChain也提供了一些内置评估类 [16] [17]，但为了满足本任务的复合标准，采用自定义Prompt让GPT-4打分更灵活。

1. **调度控制Agent（可选）：** 虽然核心流程可以由编排脚本完成，但也可以考虑引入一个高层Agent（如一个Orchestrator Agent）来根据任务需求动态调用上述各Agent。这类似于一个**监督Agent**，负责整个多Agent对话流程的安排 [18] [19]。例如Supervisor Agent读取用户输入（PDF文件列表和评估请求），然后指挥题目生成Agent开始出题，收集题目后通知各模型Agent答题，最后召集评分Agent打分，再将结果传给统计模块处理。微软的Semantic Kernel等框架可以用于实现这种Orchestrator功能 [20] [19]。不过在本系统中，调度逻辑也可以直接以编程方式实现，不一定需要LLM智能体来控制。

框架与技术栈： 实现上可以采用以下技术栈： - **OpenAI API/模型接口：** 用于访问GPT-4等模型。OpenAI的接口简单易用，可指定模型名称和参数直接获得回复。Anthropic Claude等也有类似HTTP API。此外，如果Google Gemini开放接口，也可能通过Google API调用。使用这些API可灵活调用不同云端大模型。 - **LangChain：** 提供丰富的链式调用和Agent封装工具 [21] [22]。我们可以利用LangChain的模块快速搭建PDF读取、问答生成、评估等Pipeline。例如LangChain内置的QAGenerationChain可用于从文档生成问答对 [7] [8]，Evaluation模块可用于简单正确性对比 [16] [17]。更重要的是，LangChain支持多LLM融合，可以很方便地切换底层模型或同时管理多个模型接口，这非常适合我们的多模型对比场景。 - **AutoGen (Microsoft)：** AutoGen是微软开源的多Agent对话框架，允许开发者用多个LLM Agent协作完成任务 [23]。AutoGen的Agent具备可定制和可对话的特性，支持LLM与工具、人类混合交互 [23]。如果我们的设计需要各Agent之间直接对话协商（例如评分Agent向解题Agent提问澄清思路等），AutoGen会非常有用。不过本系统的流程相对线性，Agent之间主要是数据传递，暂无需复杂对话。因此，AutoGen可作为扩展方案或替代LangChain来实现Agent框架，其优势在于定义复杂交互协议。如果日后我们希望引入**自我优化**（Agent观察到模型答错太多，自动调整出题策略）等高级功能，AutoGen灵活的多Agent编程接口将能满足需求。 - **其他工具与库：** PDF解析可用`PyMuPDF (fitz)` [24] 或 `pdfplumber` 等将PDF转文本。数据存储和传递使用Python的数据结构或Pandas DataFrame便于统计分析。最终报告生成可用Markdown模板或者直接用Jupyter Notebook汇总结果输出为PDF。若需可视化，使用Matplotlib或Plotly绘制图表。对于批量运行和结果管理，可用`multiprocessing`或任务队列（如Celery）来调度。

安全与稳定性考虑： 调用外部LLM需考虑接口速率限制和费用，可实现请求排队和缓存。Agent产生内容时注意加入异常处理，如当LLM输出格式不符预期时重新请求或调整Prompt。为了防止某个Agent失败导致整体终止，调度应在超时或错误时给予默认处理（如某模型无响应则跳过记录为空，并在报告注明）。另外，评估流程中涉及多模型并行调用，要确保不会发生数据混淆，严格保证每个模型答每道题的对应关系正确。

综上，借助现有的大模型接口和Agent开发框架，我们能够较快速地搭建起上述评估系统。多Agent机制赋予系统良好的模块边界和扩展能力，未来可以方便地替换或新增模型，以及引入更复杂的互动逻辑而不需推倒重来。

自动化流程与调用步骤

在本节，我们以单个PDF评估为例，串联各模块说明整个系统的自动化执行流程：

1. **用户上传PDF & 设置参数：** 用户通过界面或命令行提供一份英文PDF文档，并选择所需评估的模型列表（默认包含GPT-4, Claude, Gemini等）以及题目数量、难度比例等配置。系统开始解析PDF文本，将其存入内存供后续使用。

2. **题目生成Agent执行：** 调度模块调用题目生成Agent，传入PDF文本或其摘要，以及生成参数（每种题目数量/难度等）。Agent调用底层LLM（如GPT-4）根据指令生成所需题目。举例来说，指令可能请求“请从以下教材内容生成5道完形填空和5道阅读理解题，其中简单、中等、困难难度各占1/3”。LLM输出题目草案后，代码层面会对结果进行解析和格式化。如有不合理之处（比如缺少答案或格式错误），可以重新提示LLM修正 [5] [6]。生成完成后，系统持有一个题目列表，每题附有正确答案和难度标签。

3. **并行模型解题：** 对于生成的题目集，系统按照题目逐一道处理。每道题会并行或轮流送入不同模型Agent。为了说明，我们以并行为例：系统对当前题目构造统一的提示（含题干和选项），然后同时向各模型的API发送请求。其中Chain-of-Thought提示使模型返回推理过程+答案 [11]。系统收集所有模型返回，记录响应时间。**例如：** 对第1题，GPT-4用2.3秒答出“A选项，并给了详细推理”，Claude用3.1秒答出“可能是B并解释原因”，Gemini用2.0秒答出“A及简单理由”等。所有输出被暂存起来。然后系统进入下一题，重复调用。这个过程持续直到所有题目都让所有待评模型解答完毕。完成后，我们得到一份**原始回答数据表**，包含 (题目, 模型, 模型答案, 模型推理, 正确/错误标记, 用时) 等内容。

4. **答案评分Agent评估：** 接下来，调度模块将每个模型的回答送入评分Agent进行评分。为效率考虑，可以按题目批量评分或按模型批量评分。例如一次让评审Agent对比“第1题的标准答案和模型A/B/C的答案”分别打分；或者逐个模型逐个题请求评审。具体实现视LLM上下文长度和费用优化来决定。常用做法是**逐题逐模型**地调用评分Agent，以保持提示简洁且避免不同模型答案互相影响评价。
   评分Agent的输出会被解析提取分数和评语，添加回原始回答数据。例如在第1题记录中加入GPT-4答案的accuracy=1（正确）, coherence_score=9, reasoning_score=8 等。最终，我们得到丰富标注的结果数据。

5. **统计分析与报告生成：** 评分完成的数据进入统计模块做汇总。代码会遍历数据计算每模型总体正确率、平均分等。如需更直观，将计算结果载入Pandas DataFrame，以便后续制表和绘图。然后根据分析结果填入预先准备的报告模板。报告可以使用Markdown格式编写，通过脚本将各部分内容（如一个表格字符串或图像）插入相应占位符。
   例如，我们在报告的“正确率比较”段落插入一个Markdown表：

模型	完形填空正确率	阅读理解正确率	总体正确率
GPT-4	90%	85%	87.5%
Claude	85%	80%	82.5%
Gemini	88%	78%	83.0%

同时插入语言逻辑性和思维合理性平均分的对比表、各模型平均响应时间统计等。如果实现了图表生成，则将图表保存为图片文件并在报告中通过Markdown引用显示。(由于图表无法通过Markdown直接呈现，这里假定使用报告PDF方式或Notebook展示)。

最后，报告文件生成输出给用户。例如命令行打印报告路径，或在Web界面直接呈现结果。用户可以查看详细的评估，并将其用于课程作业或模型改进参考。

备注： 如果是**批量PDF**模式，上述步骤2-5会针对每个PDF循环执行，然后在报告生成时在不同章节分别汇报。调度Agent需要在每轮之间重置或区分上下文，但大体流程一致，只是在最终汇总时多了跨文档的比较分析。

整个流程高度自动化，从原始资料到分析结论几乎不需人工干预。这得益于各环节都由大模型Agent智能承担。通过合理的Prompt和脚本控制，我们能够可靠地生成题目并评估模型表现，同时保存了丰富的过程数据以备分析和改进使用。

关键评价指标与衡量方法

设计这样的评估系统，需提前明确**关键评价指标**以便全面衡量各模型的解题能力。根据需求，我们关注以下主要指标：

- **正确率（Accuracy）：** 模型给出正确答案的频率，是衡量解题能力的基本指标。计算方法通常为 正确题数/总题数，可细分为不同题型或难度下的正确率。正确率直接来源于标准答案比对，也可通过评分Agent的正确性评分判断（例如将评分>=满分的一定比例视为正确答对）。在报告中，会突出每个模型的总体正确率，并比较在完形填空、阅读理解两类题目的正确率差异，从而分析模型擅长的题目类型。

- **解题耗时（Latency）：** 每个模型平均每题用时，及其分布情况。这个指标体现模型的推理速度和效率。在实际应用中，响应速度很重要，因此在评估时需要报告模型的耗时表现。通过记录API调用时间戳，我们可以精确获取每题每模的耗时，并计算平均值和方差等。报告中可能会给出例如“GPT-4平均每题耗时2.5秒，Claude为3.0秒，Gemini为1.8秒”等结论，以及解释速度差异可能源于模型大小或架构优化等因素。

- **语言逻辑性（Coherence of Answer）：** 由评分Agent评估的模型回答语言表达和逻辑严谨程度。虽然选择题最终只需一个选项，但模型的完整回答（尤其在要求展示推理时）可以反映其语言组织能力和逻辑清晰度。评分Agent可对答案的语言给出评分或评价，例如句子是否通顺、论证是否有条理。我们将这些评分汇总为模型的平均语言逻辑得分，用于比较模型在表达和逻辑上的差异。若某模型正确率虽高但表述混乱，可能在此项得分偏低，提示其思维链条不够清晰。

- **思维链条的准确性与合理性（Reasoning Quality）：** 这是本系统特色关注指标，反映模型推理过程本身的可靠程度。通过要求模型输出思维链，我们可以检查其中的推理步骤是否正确理解了题意、是否合乎逻辑地推导出答案。如果模型在推理中出现事实性错误或不合理假设，即使最后答案猜对了，也应在此指标上扣分。我们借助评分Agent的“思维合理性”评分来量化该指标。统计每模型该项平均得分，或统计有多少回答的思维链存在明显错误。这个指标很重要，因为它反映模型解题是否**知其然亦知其所以然**。理想的模型不仅答对，还应答得有理有据 [10] 。

- **难度适应性（Performance by Difficulty）：** 评估模型在不同难度水平的问题上的表现变化情况。具体可通过前述不同难度子集的正确率或评分比较实现。如果模型A在简单题上95%正确，但难题只有60%，而模型B简单题90%、难题80%，则后者在难题上相对更稳定。我们可以定义一种**难度适应指数**，例如 困难题正确率/简单题正确率 比值，来定量比较。当该比值接近1时表示模型无论难题易题都发挥稳定；远小于1则说明模型在高难度场景下能力下降明显。此指标对于挑选**鲁棒性**更高的模型很有价值，也能帮助分析模型是否存在知识盲区或推理短板。

- **其他指标（可选）：** 若需要，更可引入诸如输出长度（模型解释的详尽程度）、创造性（对没见过的问题是否有创新解法）、一致性（多次回答同题的一致程度）等指标。不过这些不是本系统刚需，可以在未来扩展评估范围时考虑。

评价指标的度量实现： - 正确率由系统直接比对或评分Agent判定，最为客观明确。 - 耗时通过系统日志提取，精确到毫秒级。 - 语言逻辑性和思维合理性来自评分Agent给出的定量分数，我们需保证评分标准清晰一致，以提高这些主观指标的可靠性 [13] 。为此，可以让同一道题由评审Agent多次评估取平均（即让评审Agent自己做

Chain-of-Thought评估 [15]），或者人工校验部分评分以确保合理。 - 难度适应性可从题目元数据和正确率组合得到，无需额外模型支持。

综合评价： 系统会综合以上指标给出对模型的总体评价。例如，可以设计一个**总分**，加权组合正确率、平均评分和难度适应性等，生成模型排名。但考虑到不同应用关注点不同，我们更倾向以**雷达图**或表格形式呈现各模型多维度成绩，让读者自行权衡。例如报告结论部分可以写道："GPT-4在正确率上略胜一筹，Claude在推理合理性上表现突出，而Gemini响应最快但在高难度题上正确率下降明显"，并配上各指标对比图。这种多指标分析确保评估结果全面、客观，能够凸显每个模型的优势和短板，为后续模型改进或选型提供有力依据。

引用支持： 在编制课程设计报告时，上述关键指标与方法很多都有文献或实践支持，例如LLM作为评审的有效性 [13]、Chain-of-Thought对模型推理能力的提升 [11] 等，我们在文中已附上相应引用以示依据。

通过上述模块设计、Agent配置和指标体系，本工作流系统可以自动地生成测试题并评估多种大语言模型的解题能力。系统架构清晰、模块解耦，利用Agent机制充分发挥了LLM在文本生成和理解上的强大能力 [23]。整个方案兼具**实用性**（可应用于教学测试、模型benchmark等）和**扩展性**（易于替换模型或增加任务类型）。如果需要进一步完善，还可考虑引入人工审核环节以提高题目质量，或者让多个评审Agent投票以增强评分客观性。但即使在纯自动化配置下，本系统也能产出有价值的评估报告，帮助我们深入了解不同大型语言模型在英语理解和推理任务上的表现差异。祝课程设计顺利开展！ [10] [13]

---

[1] [2] Understanding LLM-Based Agents and their Multi-Agent Architecture | by Pallavi Sinha | Medium
https://medium.com/@pallavisinha12/understanding-llm-based-agents-and-their-multi-agent-architecture-299cf54ebae4

[3] [12] [13] [14] [15] LLM Evaluation: A Practical Guide for Developers | by Akash Mathur | Medium
https://akash-mathur.medium.com/llm-evaluation-a-practical-guide-for-developers-fb25608e920b

[4] [2403.02078] Automated Generation of Multiple-Choice Cloze Questions for Assessing English Vocabulary Using GPT-turbo 3.5
https://arxiv.org/abs/2403.02078

[5] [6] Personalized Cloze Test Generation with Large Language Models: Streamlining MCQ Development and Enhancing Adaptive Learning
https://aclanthology.org/2024.inlg-main.26.pdf

[7] [8] Document-based question generation : r/LangChain
https://www.reddit.com/r/LangChain/comments/15264ee/documentbased_question_generation/

[9] [10] [11] Chain-of-Thought Prompting: Step-by-Step Reasoning with LLMs | DataCamp
https://www.datacamp.com/tutorial/chain-of-thought-prompting

[16] [17] LangChain: Automating Large Language Model (LLM) Evaluation
https://www.analyticsvidhya.com/blog/2024/01/langchain-automating-large-language-model-llm-evaluation/

[18] [19] [20] Reference architecture - Multi-agent Reference Architecture
https://microsoft.github.io/multi-agent-reference-architecture/docs/reference-architecture/Reference-Architecture.html

[21] [22] [24] Build a Document QA generator with Langchain and Streamlit - DEV Community
https://dev.to/sathish_panthagani/build-a-document-qa-generator-with-langchain-and-streamlit-1di0

[23] AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation - Microsoft Research
https://www.microsoft.com/en-us/research/publication/autogen-enabling-next-gen-llm-applications-via-multi-agent-conversation-framework/