ARC提示词工程实验报告

一、实验背景与目标

- 简述本实验的目标：利用提示词工程让大语言模型（LLM）解决 ARC（Abstraction and Reasoning Corpus）推理任务。
- 概述 ARC 任务特点：输入输出网格变换、抽象规则、泛化与推理难度。
- 明确本实验的核心问题：如何通过提示词设计引导模型发现隐藏规律并输出正确结果。

二、实验环境与工具

- **模型及 API**：如 DeepSeek-V3.2 (chat 模式)，temperature=1.0，max_tokens=8k。
- **运行环境**：Python 3.9，使用官方 API。
- **文件说明**：
  - `val.jsonl`：验证集，用于自测。
  - `template.py`：包含 `construct_prompt()` 与 `parse_output()` 函数。
  - `report.pdf`：即本报告。

三、提示词策略设计思路

3.1 基础提示词设计（Baseline）

- 系统提示词设计逻辑
  1. 角色设定

     将模型定义为“ARC谜题专家”，强调其在“模式识别”和“逻辑推理”方面的能力。
  2. 任务目标明确化

     分析训练样本的变换模式

     识别所有样本中一致的规则

     将规则精确地应用到测试输入上
  3. 行为约束

     请系统化和精确地进行分析。
- 用户提示词设计逻辑
  1. 结构化分步设计

     明确解题过程（**观察** → **模式发现** → **规则验证** → **应用** → **输出**）。
  2. 强化观察阶段

     要求模型从**网格维度、数值分布、非零元素位置、空间结构**等角度全面观察。
  3. 指导模式发现

     明确列出可供思考的变换类型（**平移、旋转、反射、颜色变化、对称性、数学关系**等）。
  4. 规则验证

     强制模型验证规则在所有样本上的一致性，鼓励其形成“通用规则”而非单例映射。
  5. 测试应用与输出

要求模型系统化应用已验证规则，并展示推理过程。

最后规定输出格式必须为 `[[...], [...]]` 的 Python 二维列表，并加上“最终输出：”标识。

这一格式约束减少了解析错误，有助于程序自动提取结果。

- 问题与不足
  1. **无法应对复杂问题**，例如使用多个操作对矩阵进行变化。
  2. **泛化能力有限**，缺乏层级推理机制。
- 结果

```
Epoch 1 Results:
  Successfully processed: 20/20 samples
  Failed samples: 0
  Correct predictions: 15/20
  Accuracy (on completed samples): 75.00%

==================================================
=== Final Results ===
Total samples in dataset: 20
Final Accuracy: 75.00%
```

3.2 尝试一：丰富ARC操作步骤

- 引入多步骤构建与验证机制

  优先单步规则：**先尝试最简单的单步操作**（如旋转或颜色映射），验证是否匹配所有训练样本。
  迭代多步扩展：如果单步失败，逐步添加第二步、第三步操作**（最多三步）**，以**模拟复合变换。**
  限制步数：最多三步，避免过度复杂化，确保模型遵循**奥卡姆剃刀原则**（优先简单解释）。
  验证强化：在规则验证阶段，明确要求展示每步中间结果，并迭代调整规则。

- 更改部分如下

```
### 步骤2：模式发现
通过以下方面识别变换规则。**优先尝试单步规则，如果单步无法一致匹配所有样本，则逐步添加第二步、第三步操作（最多三步组合）**：
- 移动模式（平移、旋转、反射、缩放）
- 颜色/数值变换
- 元素的分组或分割
- 对称操作
- 数学关系
- 基于区域的操作（角落、边缘、中心）
- 复制或删除模式

**规则构建原则**：
- 先假设单步规则（如“所有输入元素顺时针旋转90度”），并在步骤3验证。
- 如果单步规则无法匹配所有训练样本，则尝试两步组合（如“先提取红色元素，再旋转90度”）。
- 如果两步仍不匹配，则尝试三步组合（如“先复制元素到四个角落，再翻转，再填充颜色”）。
- 最多使用三步操作；优先最简单的解释（奥卡姆剃刀原则）。
- 清晰描述每步操作的顺序和细节。
```

- 运行结果

  与Baseline相比下降10%

```
Epoch 1 Results:
  Successfully processed: 20/20 samples
  Failed samples: 0
  Correct predictions: 13/20
  Accuracy (on completed samples): 65.00%

==================================================
=== Final Results ===
Total samples in dataset: 20
Final Accuracy: 65.00%
```

- 结果分析
    1. prompt长度、复杂性增加。可能导致模型在处理时注意力分散或token超限。
    2. 在temperature=1.0的设置下，复杂prompt易产生“创意变异”。
    3. 模型推理链负担加重。推理过程消耗更多计算资源。
    4. 多步验证增加输出变异性。

3.3 尝试二：丰富ARC操作类型

- 描述引入思维链提示（如“请先描述推理步骤，再给出最终输出”）。
- 给出示例 prompt。
- 分析结果变化（如推理解释变得合理，但最终输出易混入文字）。

```
Epoch 1 Results:
  Successfully processed: 20/20 samples
  Failed samples: 0
  Correct predictions: 12/20
  Accuracy (on completed samples): 60.00%

==================================================
=== Final Results ===
Total samples in dataset: 20
Final Accuracy: 60.00%
```

3.4 优化尝试四：

- 尝试在提示词中引导模型“归纳变换规则”。
- 示例：通过描述“找出元素在矩阵中的相对位置规律”。
- 效果：泛化性增强，但仍存在部分错误输出。

3.5 最终策略说明

- 汇总最终采用的 prompt 结构（附最终模板）。
- 说明设计选择的原因（如在准确率与稳定性之间的平衡）。

四、输出解析策略（parse_output）

- 说明模型输出中如何提取二维列表。
- 提供简单的解析正则/逻辑代码。
- 举例说明一次成功解析的样例。

五、实验结果与分析

5.1 在验证集上的表现

实验编号	提示词策略	准确率（val.jsonl）	备注
1	基础版	0.75	结构简单，但是效果好。 考虑到真实数据会有更难的解题策略， 还需要进一步优化。
2	尝试一（多步骤）	0.65	下降10%。
3	面向复杂问题	0.60	下降15%。
4	自适应+示例	0.50	添加的约束过多，

5.2 定性分析

- 对比不同策略在“规则抽象”“输出格式”“稳定性”上的差异。
- 可附若干示例任务截图或表格，展示预测与真实输出对比。

六、总结与反思

- 总结最有效的提示词设计特征。
- 反思当前方法的局限（如模型对空间逻辑理解有限）。
- 提出未来改进方向（如引入视觉化 prompt 或多模态提示）。

七、附录

- 附上部分实验记录（不同 prompt 与对应输出节选）。
- 可能附 `construct_prompt` 关键代码片段（非完整代码）。
- 可列出本次使用的主要参考思路或相关论文。

✅ **评委关注重点（建议在报告中体现）：**

- 展示你**系统探索的过程**（不是只贴一个成功 prompt）。
- 每个版本的 prompt 都要有**设计理由 + 效果分析**。
- 报告清晰、有逻辑，能体现**思考深度与创新性**。
- 图表清晰、结构规范（建议 4~6 页为宜）。