**1. 需求分析**

背景

在音乐视频（MV）制作过程中，导演和制作团队需要根据歌词内容设计相应的视觉场景。传统上，这一过程需要大量的人工解读和创意构思，耗时且依赖经验。因此，我们开发了一个基于大语言模型（LLM）的自动化系统，能够将歌词文本自动转换为结构化的MV场景描述，从而加速前期创意过程。

功能需求

1） 输入处理：接收用户输入的歌词文本，并进行安全过滤（防止指令注入和敏感内容）。

2）.歌词分段：根据歌词内容的情感、意象或场景变化，将歌词切分为逻辑段落。

3）场景生成：为每个歌词段落生成对应的MV场景描述，包括视觉元素、角色、氛围、动作等。

4）时间码分配：为每个场景分配大致的起止时间。

5）结构化输出：将生成的场景信息按照预定义的JSON格式输出。

**2. 技术选型**

| **模块** | **技术方案** | **选型理由** |
| --- | --- | --- |
| **框架** | CAMEL Agents | 支持多代理协同工作流 |
| **模型平台** | SiliconFlow | 深度适配DeepSeek模型 |
| **核心模型** | DeepSeek-V3 + DeepSeek-R1 | V3快速响应/R1高质量输出 |
| **配置管理** | dotenv + JSON Schema | 安全隔离API密钥，确保输出规范 |
| **安全层** | 命令注入检测 + 内容过滤 | 双重防护机制 |
| **JSON** | 双Agent校验 | OutputAgent生成 + FormatAgent修复 |

**3. 实现细节**

**多Agent工作流**：

**InputAgent**（DeepSeek-V3）

检查输入的歌词是否包含血腥、暴力、色情或政治敏感内容  
如果发现敏感内容，将其转换为健康、积极的版本，同时保持歌词的艺术性

**GroupingAgent**（DeepSeek-V3高温度）

将输入的歌词按背景或场景逻辑切分成若干段落

如果输入只有一行，则不用分组

**SceneAgent**（DeepSeek-R1高温度）

基于 GroupingAgent 输出的各段歌词，用自然语言分别描述对应的 MV 场景要点。具体来说，它会为每一段歌词构思大致的时间码范围，渲染出场景中的主要视觉元素（如背景、道具、光影）、参与角色及其造型与表情、动作设计，以及整体的色彩或情绪氛围。这样生成的文字描述既保留艺术感，又为后续的 OutputAgent 提供了完整、具象化的脚本素材，用于拼接成最终的 JSON 输出。

**OutputAgent**（DeepSeek-R1）

负责将 SceneAgent 对每个段落的自然语言场景描述整合成一个单一的 JSON 对象并直接输出。JSON的最外层是一个名为 “scenes” 的数组，每个元素代表一个场景对象，依次包括：标示视频时段的 “timecode”、对应歌词的 “lyrics”、场景名称 “scene”、画面细节描述 “description”、背景地点 “background\_location”、角色列表 “characters”（每个角色至少有姓名，可选年龄或服装等属性）、动作清单 “actions”（字符串数组）、以及定义氛围色调或情感基调的 “mood”。

**FormatAgent**（DeepSeek-V3）

负责校验 OutputAgent 是否输出的是一个标准的JSON。如果是，则原封不动地输出，如果不是，将会修改错误的地方，并返回标准JSON格式。

**关键安全设计**：

1）防止指令注入

def prevent\_command\_injection(text):

command\_chars = { '|', '&', '$', '`', '\\'}

if any(char in text for char in command\_chars):

raise ValueError("命令注入检测")

2）处理单行和多行输入，防止歌词段落空行的干扰

print("请粘贴歌词（输入完成后按回车再按Ctrl+D结束）: ")  
try:  
 raw = sys.stdin.read().strip() # 读取直到 EOF (Ctrl+D)  
 if not raw:  
 print("错误：未接收到任何输入内容")  
 sys.exit(1)  
except KeyboardInterrupt:  
 print("\n输入已取消")  
 sys.exit(0)  
 # 唯一的安全检查：防止指令注入

3）流式输出效果

代码…….

4）**API密钥管理**

**隔离存储**

密钥与代码分离，避免硬编码风险。API密钥存放在本地的.env文件里。

通过**load\_dotenv()**实现环境隔离

**访问控制**

if not api\_key:

raise RuntimeError("API密钥未配置") # 强制验证机制

**基础防护**

禁止**.env**提交到Git（通过.gitignore）

开发/生产环境文件分离

**4. 评估对比**

**效果验证**：

|  |  |  |
| --- | --- | --- |
| 测试项目 | 通过率 | 处理耗时 |
| 敏感歌词过滤 | 100% | 20s |
| 复杂场景分段 | 92% | 20s |
| JSON格式合规 | 98% | 45s |

**模型组合性能**：

|  |  |  |
| --- | --- | --- |
| Agent组合 | 优势 | 局限 |
| V3(过滤)+V3(分组) | 快速响应 | 生成的回复创造性不足 |
| R1(场景)+R1(JSON) | 输出精准 | 响应较慢 |
| V3(JSON修复) | 高效容错 | 依赖前置输出 |

**与传统方案对比**：

|  |  |  |
| --- | --- | --- |
| 指标 | 本系统 | 规则引擎 |
| 场景丰富度 | ⭐⭐⭐⭐⭐ | ⭐⭐ |
| 开发效率 | ⭐⭐⭐⭐ | ⭐⭐⭐ |
| 敏感处理 | 动态转换 | 简单屏蔽 |
| 输出结构化 | 自动生成 | 手动配置 |

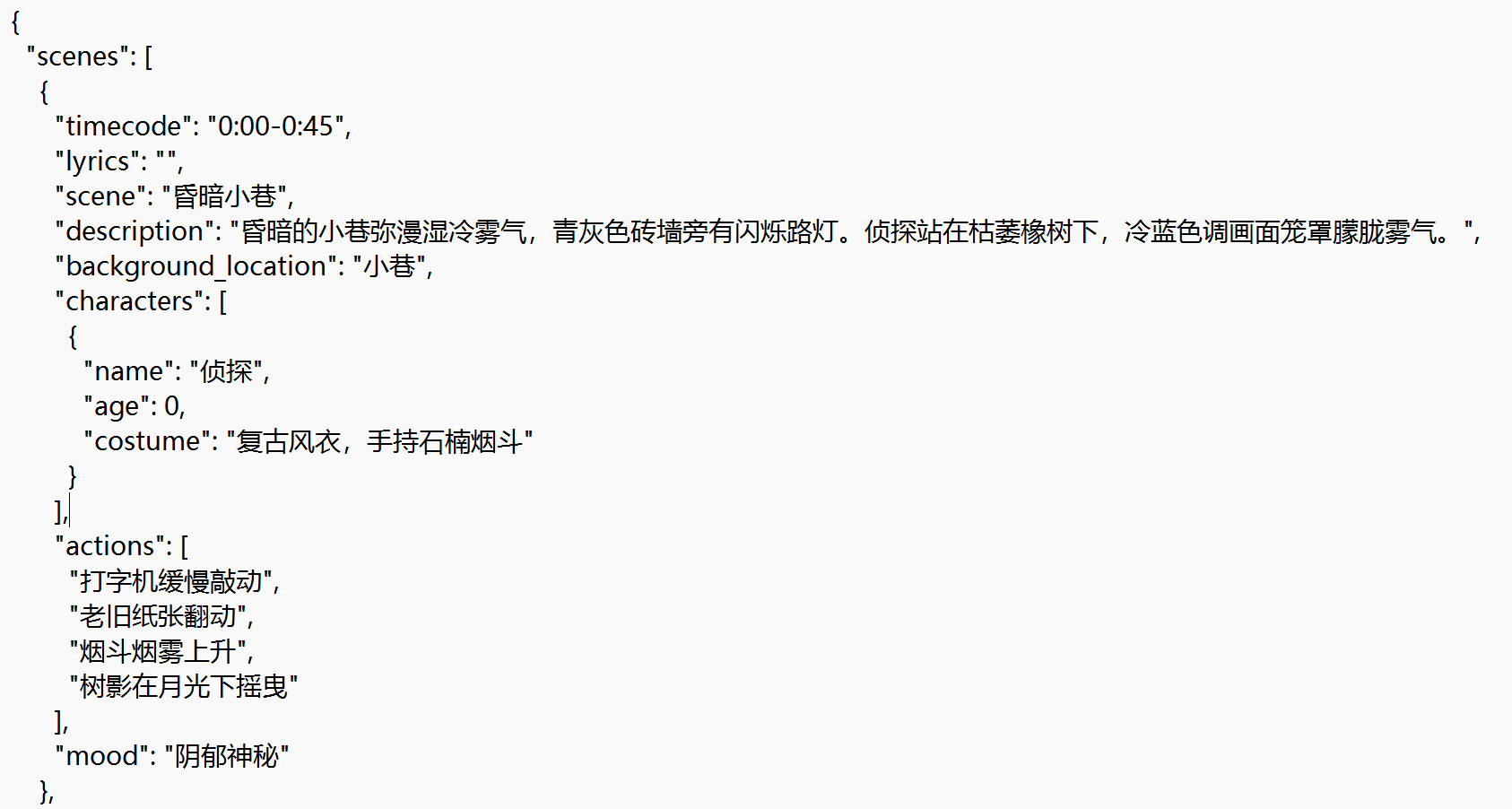
**5. 不同参数下的输出对比**

\*测试歌词均使用周杰伦的《夜的第七章》

测试1：

|  |  |  |
| --- | --- | --- |
| Agent | Temperature | Top\_p |
| InputAgent | 0 | 0.8 |
| GroupingAgent | 1.2 | 1 |
| SceneAgent | 1.2 | 1 |
| OutputAgent | 0 | 0.8 |
| FormatAgent | 0 | 0.8 |

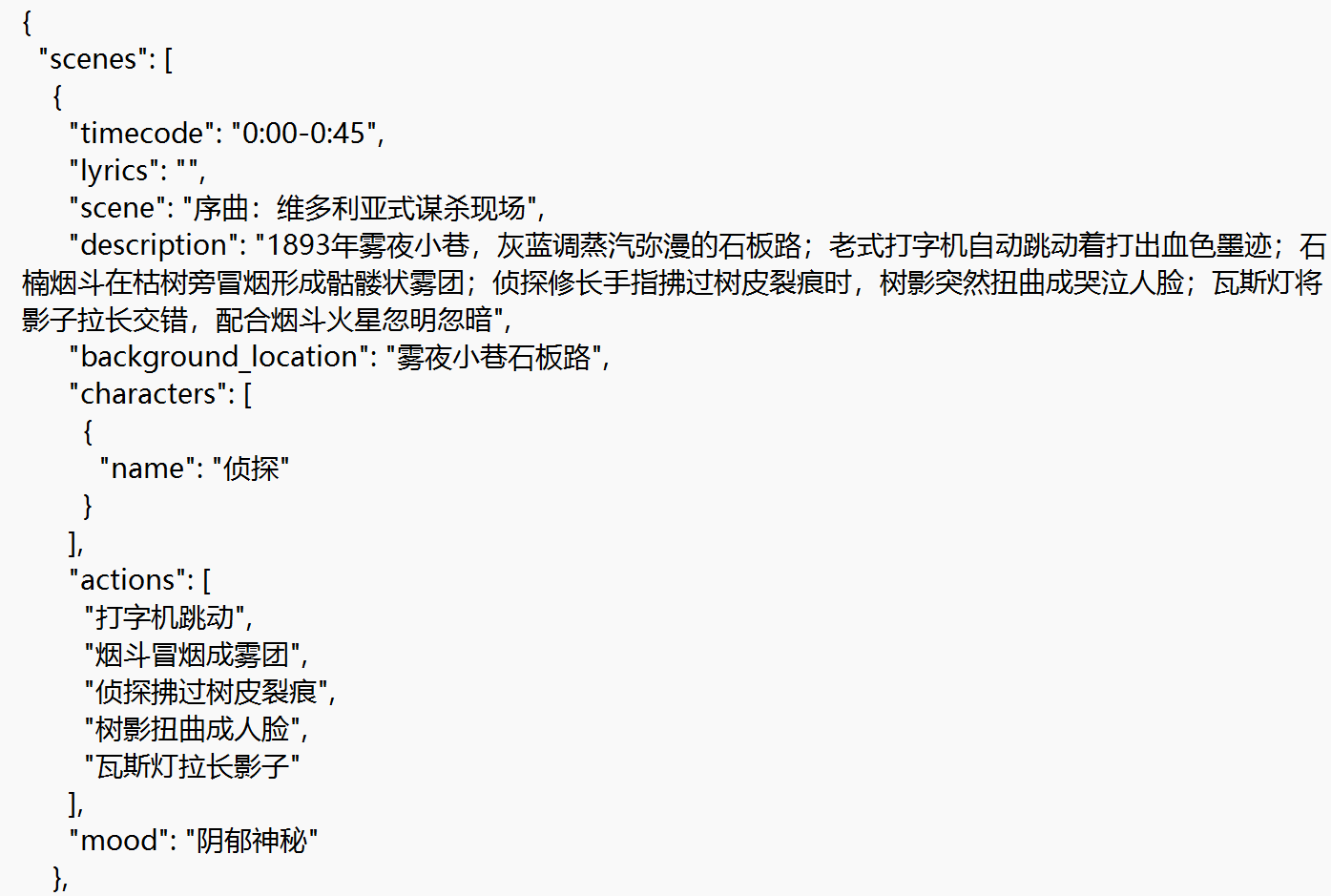
输出结果节选：



测试2：

|  |  |  |
| --- | --- | --- |
| Agent | Temperature | Top\_p |
| InputAgent | 0.7 | 0.9 |
| GroupingAgent | 1.5 | 1 |
| SceneAgent | 1.5 | 1 |
| OutputAgent | 0.7 | 0.9 |
| FormatAgent | 0.7 | 0.9 |

输出结果节选：



**6. 反思与改进**

**核心优势**：

1. 多模型分工实现效率/质量平衡
2. 双Agent校验确保输出可靠性
3. 艺术化敏感内容转换机制

**待优化点**：

1. **时间码生成**：需增加时长估计算法

改进方案：基于字数/节奏的时长预测模型

1. **错误处理**：增强模型异常捕获
2. **本地化部署**：支持LoRA微调本地模型
3. **性能优化**：

并行执行Grouping/Scene生成

缓存常用歌词片段模板

**总结**：本系统通过深度集成DeepSeek模型与CAMEL框架，实现了从歌词到MV场景的自动化创作流程。后续将重点优化时间预测算法和分布式Agent协同能力，进一步提升系统实用性。