Novel Generator 生成逻辑详解

Novel Generator 采用分层递进的生成策略，确保小说的逻辑连贯性和设定一致性。整个生成流程分为以下几个主要阶段：

🏗️ 第一阶段：小说架构生成

architecture.py 负责生成小说的整体框架，按以下顺序执行：

1. 核心种子生成 - 基于用户输入的主题、类型、章节数等参数，生成小说的核心冲突和基础设定
2. 角色动力学设定 - 根据核心种子创建3-6个核心角色，包括背景、驱动力、角色弧线和关系网络
3. 初始角色状态表 - 基于角色动力学生成角色的初始状态，保存到 character_state.txt
4. 世界观构建 - 创建三维交织的世界观（物理维度、社会维度、精神维度）
5. 三幕式情节架构 - 整合前面所有元素，构建完整的剧情框架

最终输出 Novel_architecture.txt，包含完整的小说设定基础。

📋 第二阶段：章节蓝图生成

blueprint.py 根据小说架构生成详细的章节目录：

- 支持分块生成：当章节数较多时，自动分批生成避免提示词过长
- 断点续传：可从中断处继续生成，提高容错性
- 输出 Novel_directory.txt，包含每章的标题、作用、人物、场景等详细信息

✍️ 第三阶段：章节草稿生成

chapter.py 是核心的章节内容生成模块：

智能上下文构建

1. 历史章节摘要 - 获取前3章内容并生成连贯性摘要
2. 角色状态追踪 - 读取当前角色状态，确保角色发展的一致性
3. 知识库检索 - 基于章节信息生成检索关键词，从向量库中获取相关背景知识
4. 内容过滤 - 对检索到的知识进行相关性过滤，确保内容质量

提示词构建与生成

- 整合所有上下文信息构建完整的生成提示词
- 支持用户编辑提示词，提供更大的创作灵活性
- 生成章节草稿并保存到 chapters/chapter_X.txt

🎯 第四阶段：章节定稿

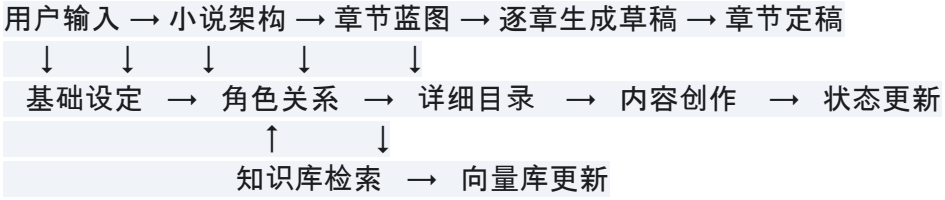
finalization.py 完成章节的最终处理：

1. 字数检查与扩写 - 如果字数不足目标的70%，自动进行扩写
2. 前文摘要更新 - 根据新章节内容更新全局故事摘要
3. 角色状态更新 - 更新角色的发展状态，为后续章节提供准确的角色信息
4. 向量库更新 - 将定稿章节加入向量库，为后续章节提供检索素材

🔍 辅助功能模块

- 一致性检查 (consistency_checker.py) - 检测剧情矛盾和逻辑冲突
- 知识库管理 (knowledge.py) - 支持导入自定义知识文档
- 向量存储 (vectorstore_utils.py) - 管理语义检索和长程上下文维护

🔄 生成流程总结



这种分层递进的设计确保了：

- 逻辑连贯性：每个阶段都基于前一阶段的输出
- 状态一致性：通过角色状态表和摘要维护角色发展轨迹
- 内容丰富性：通过知识库检索提供背景支撑
- 可控性：用户可在每个阶段进行干预和调整