方案整体描述

本项目旨在构建一套自动化、高品质的金融研究报告生成系统。我们围绕“**规划-搜集-生成-优化**”的核心思想，设计了一套模块化、可扩展的智能体（Agent）工作流。

技术上，该系统深度融合了多引擎网络爬虫（如DDG、搜狗）与结构化数据处理能力，构建了全面的数据基础。在内容生成环节，我们集成了Mermaid、Matplotlib等可视化工具，支持将复杂数据与逻辑关系自动转换为流程图、数据图表，确保报告图文并茂、论据清晰。

针对不同报告类型的核心痛点，我们设计了三套差异化的工作流：

1. **公司/个股研报:** 采用“**监督式迭代优化**”流程，模拟“分析师初稿 -> 主管审查 -> 多人协作修改”的模式。由一个Supervisor Agent主导，通过多轮评审和调用专用工具Agent，对报告进行精细打磨，追求深度与准确性。
2. **行业/子行业研报:** 采用“**决策驱动的章节式生成**”流程。由一个MainDecision核心决策Agent，根据当前信息完整度，动态决定是继续“搜索信息”还是“生成章节”，以结构化、迭代的方式高效构建报告。
3. **宏观经济/策略研报:** 采用“**研究导向的闭环探索**”流程。其核心在于为每个章节构建“知识库”，通过“评估-搜索-入库”的循环，直至信息量达到“充足”阈值后才开始撰写，以保证内容的广度与论证的充分性。

不同方案的详细描述如下：

公司/个股研报自动化生成器

预估推理时间

约 40 - 50 分钟

推理资源

使用火山引擎的web api, 推理模型为`deepseek-v3-250324`.

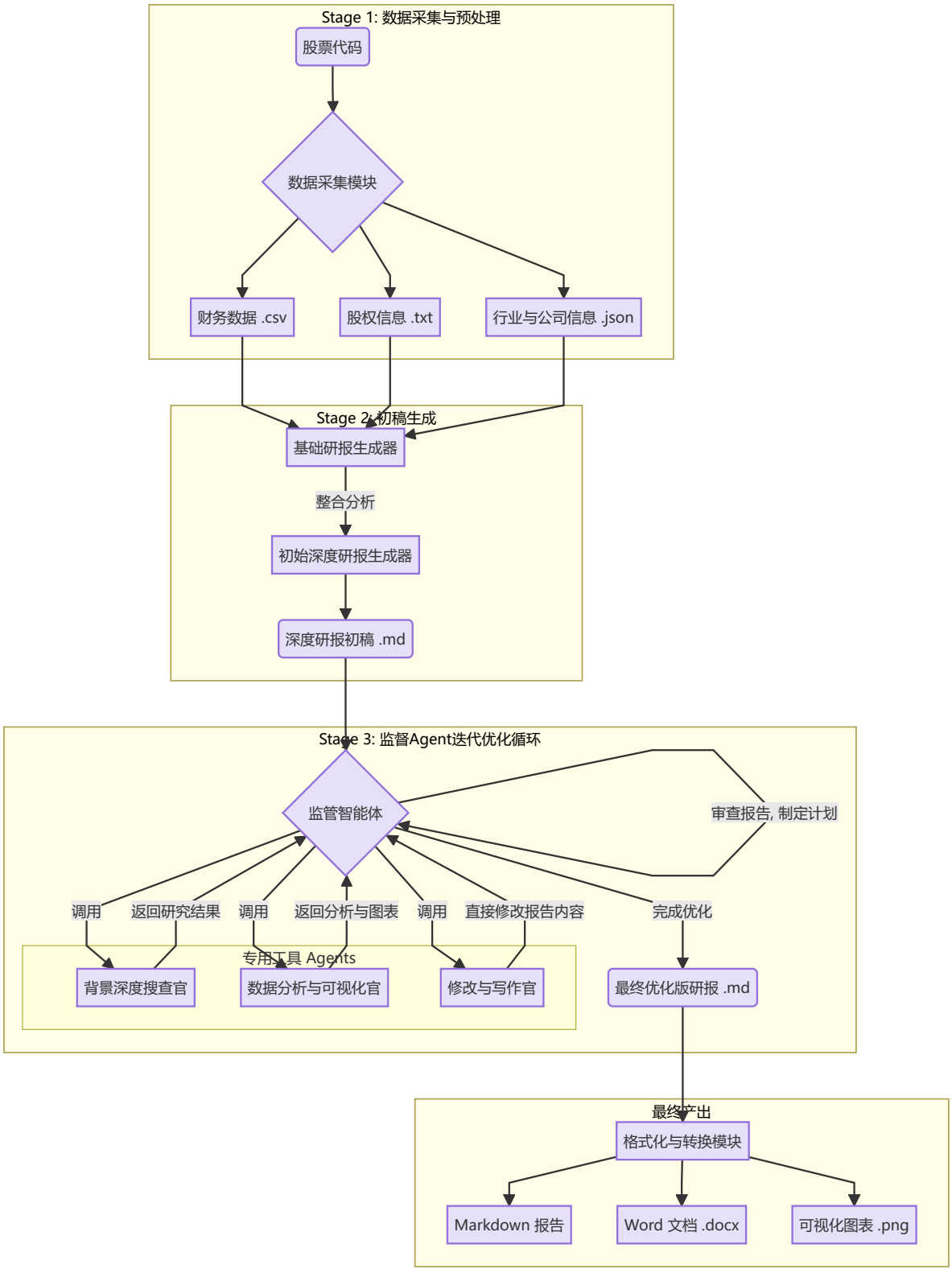
算法描述

本项目的核心是一个**三阶段、带监督优化**的自动化报告生成流程。它模拟了人类分析师“搜集资料 -> 撰写初稿 -> 审查修改”的工作模式，旨在确保报告的深度、准确性和逻辑连贯性。

Baseline：项目最初的版本是一个单通道工作流，即“数据采集 -> 财务分析 -> 报告生成”，一次性输出完整报告。这种方式速度快，但内容可控性差，容易出现事实错误和逻辑断裂。

当前架构：为了解决上述问题，我们引入了**监督式优化机制**。整个流程被重构为三个主要阶段，特别是增加了由多个专用Agent协作的迭代优化环节。

各个阶段的运转逻辑和算法流程如下图所示：



算法迭代过程

1. **第一次迭代：Prompt工程优化与引用修复**
   针对Baseline版本中的引用缺失、链接格式不完整（如仅有标题无链接）等问题，我们通过精细化的Prompt工程进行了修复。
2. **第二次迭代：图表解读与量化分析**

第二次解决了一个关键问题，让大模型读懂图，有两种实现方式，第一种是使用图生文大模型，这种方式简单直接，但需要耗费更多的时间，且火山模型不支持图像的传入。对于可视化图表，因其具有结构化的特征，因此更合理的方式是直接将基于Echarts等声明式语言，将所需的数据以及图表格式声明给到大模型。基于baseline的方式，我们在`quick_analysis`流程中强制模型在生成图表（如matplotlib代码）的同时，必须输出用于绘图的**核心数据**，并基于这些数据撰写**量化分析**。这确保了报告中的图文是强相关的，避免了看图说话式的空泛描述。

3. **第三次迭代：引入数据搜查Agent与Mermaid增强**

   第三次将搜索改为深度搜索，独立了一个专门的数据搜查官用于检索。该Agent能够通过多轮"搜索-评估"的循环，针对复杂问题搜集更全面、更深入的背景资料。

   同时，我们扩充了对`Mermaid`的支持，让模型能根据文本内容直接生成流程图、饼图等简单的示意图，丰富了报告的表现形式。

4. **第四次迭代："生成-评审-修改"监督式优化流程**
   针对生成内容的可控性，我们考虑了两种模式，完全多agents协作模式或是在现有工作流的基础上增加监管的机制。完全多agents协作流容易导致上下文过长，且耗时很长。权衡之下，我们转而在现有工作流的基础上，增加了一个独立的**监督式优化阶段**。
   在深度研报初稿生成后，由一个`Supervisor Agent`接管。它会：

   - **评审**初稿，识别内容缺失、逻辑错误、图表渲染失败等问题。
   - **规划**一系列修改任务。
   - **调用**专用的工具Agents（背景搜查官、数据分析官、写作修改官）来执行任务。
   - 通过多轮迭代，对报告进行精细打磨，增强最终内容的质量和可控性。

待办与未来展望

在开发过程中，我们探索了更多可能性，但部分功能因复杂度、成本效益等原因，暂未完全集成，作为未来的优化方向：

1. **精准网页内容爬取与解析**：第一个是对于搜索到的网页的html的完整爬取，我们尝试了协作的方式，一个agent用于搜索并根据description及时删除不需要的网页，第二个agent利用上一个生成的网页进行完整页面的爬取与总结，但这样往往存在大量的无用冗余信息，例如网页表头，百科内大量的冗余信息。改进的措施包括将原始搜索存入向量数据库中，检索后给到大模型。但仍然无法解决爬取结构性表格，或者下载网页内pdf等文件的问题。因此，考虑到对项目产生的优势较小，且需要消耗大量token，推理时间较长，我们并未集成精确爬取。
2. **集成金融数据接口 (如Akshare)**：Akshare等库提供了海量的金融数据接口。要让Agent能够有效利用，需要对庞大的接口文档进行梳理，通过RAG（检索增强生成）的方式，让Agent在需要特定数据时能自动查找并调用正确的接口。

Get Started

1. 安装了docker之后在本目录下执行以下代码

```
docker build -t <镜像名称>:<镜像版本号> .  # 构建镜像
docker run -it --name <容器名称> <镜像名称>:<镜像版本号>  # 启动容器
```

2. 进入容器后执行以下脚本：

```
./run.sh
```

或是自定义参数直接调用python脚本：

```
python3 run_company_research_report.py \
    --company "商汤科技" \
    --code "00020" \
    --market "HK" \
    --search-engine "ddg"
```

3. 执行完毕后会在当前目录下生成 `深度研报_YYYYMMDD_..._final.docx`

行业/子行业研报自动化生成器

预估推理时间

约10分钟

推理资源

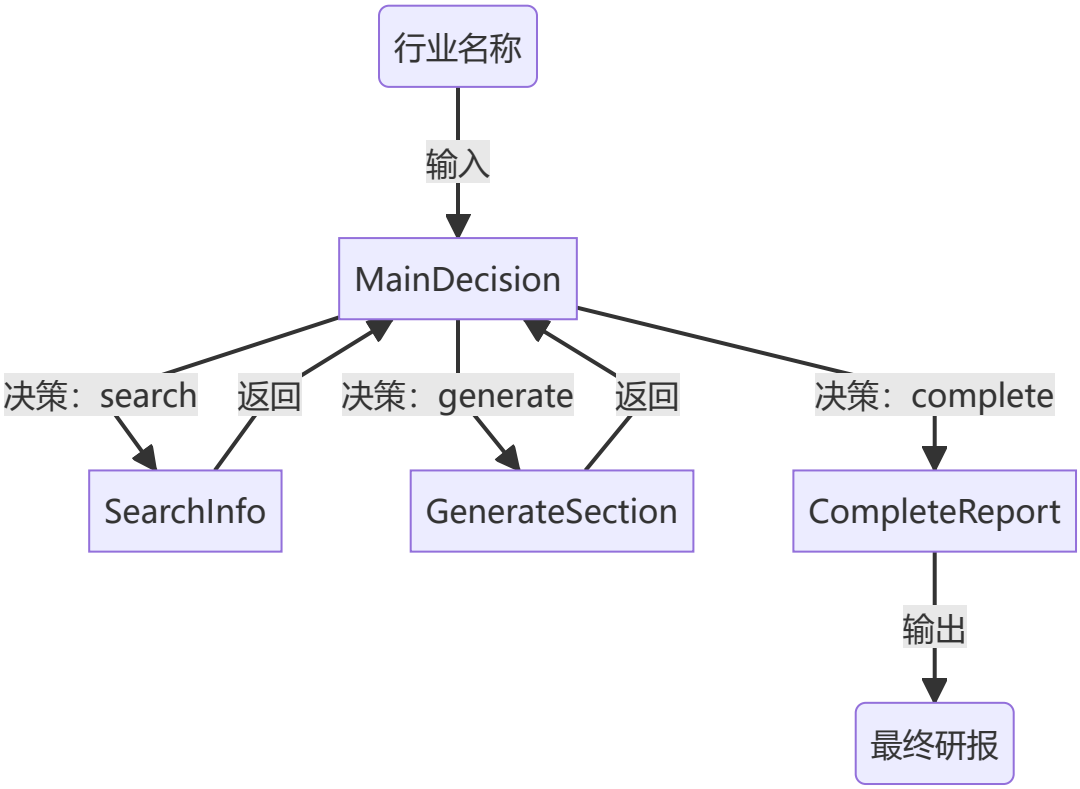
使用火山引擎的web api, 推理模型为 `deepseek-v3-250324`.

`temperature` = 0.7

算法描述

构建 `MainDecision`, `SearchInfo`, `GenerateSection`, `CompleteReport` 四个节点，分别负责：决策下一步操作，爬取指定网络信息，生成单章报告，整合输出最终报告。其中，`MainDecision`, `GenerateSection` 均为基于不同上下文和Prompt的智能体。

各个节点之间的运转逻辑和算法流程如下图所示：



算法迭代过程

baseline：仅包含两个节点，爬虫工具节点`SearchInfo`和撰写报告智能体节点`GenerateReport`，当智能体判断信息不足时自动调用爬虫工具进行补充；

1. **第一次迭代**

   直接完成完整的报告可能上下文过长，更容易出现幻觉，因此改为`GenerateSection`节点，每次仅生成一个章节，再通过`CompleteReport`节点整理输出完整报告；

2. **第二次迭代**

   `GenerateSection`节点需要兼顾爬虫工具调用和章节内容生成，输出格式较为混乱，因此独立设置一个决策节点`MainDecision`，此后`GenerateSection`节点仅负责根据给定内容生成文本；同时将节点之间传递的信息固定为yaml格式，便于格式化报告文本和工具调用；

3. **第三次迭代**

   在`GenerateSection`节点中的智能体prompt中，添加了mermaid代码的模板，并在`CompleteReport`节点中添加了将mermaid转化为对应图表的功能，使生成的报告内容更加丰富。

Get started

1. 安装了docker之后在本目录下执行以下代码

```
docker build -t <镜像名称>:<镜像版本号> .  # 构建镜像
docker run -it --name <容器名称> <镜像名称>:<镜像版本号>  # 启动容器
```

2. 进入容器后执行以下脚本：

```
./run.sh
```

或是自定义参数直接调用python脚本：

```
python3 run_industry_research_report.py \
    --industry_name <行业名称> \
    --output_dir <临时文件的输出目录> \
    --max_decide <最大迭代次数>
```

3. 执行完毕后会在当前目录下生成`<行业名称>_行业研报.docx`

宏观经济/策略研报自动化生成器

预估推理时间

约60分钟

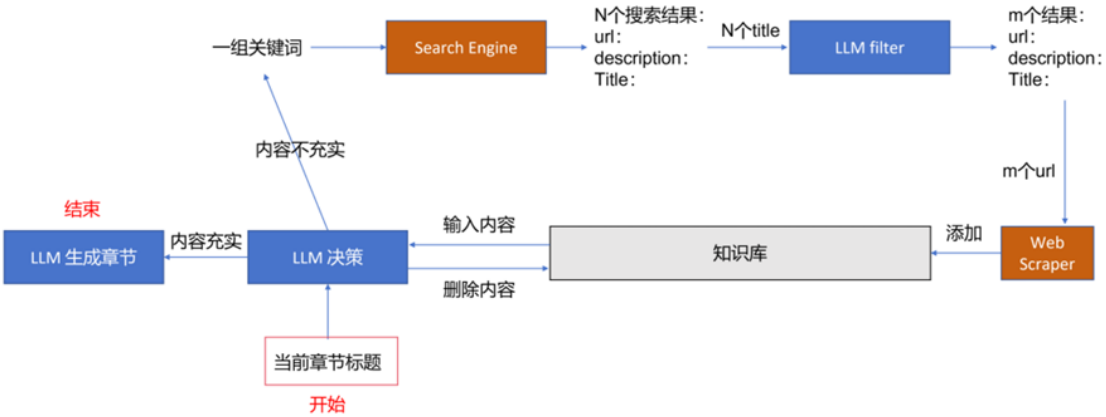
推理资源

使用火山引擎的web api, 推理模型为`deepseek-r1-250528`.

`temperature` = 0.7

算法描述

1. **章节规划初始化**
   - 利用 `PlanSections` 节点调用LLM，根据宏观主题自动输出 8-9 个章节的 `name` 与 `focus`。
   - 初始化全局状态：章节队列、搜索尝试计数器、空知识库实例 `KnowledgeBase`。
2. **循环式信息采集—决策模块**
   对每个章节反复执行信息采集与评估的闭环，直到系统判定信息"足够充实"为止。该循环的逻辑如下图所示：



   - **(1) LLM决策 (`IndustryResearchFlow`)**
     - 读取本章节已缓存的资料列表（带索引）并评估信息完整度。
     - 若信息不足 → 返回 action: search 并给出 3-5 个高精度检索词。
     - 若信息充足 → 返回 action: generate。
     - 同时可返回 remove_indices，触发对知识库冗余/低质内容的清理。

- **(2) 智能搜索 (`SearchInfo`)**
  - 对每个检索词调用统一封装的 SearchEngine（DuckDuckGo→搜狗→Bing 顺序回退）。
  - 使用hash-cache，结果先落盘缓存，避免重复下载。
- **(3) 标题过滤 (`FilterTitles`)**
  - 使用LLM对搜索结果标题进行相关性二次筛选，去除不相关的内容。
- **(4) 网页抓取 & 入库**
  - 对保留的 m 个 URL 由 Web Scraper 抓取正文，落盘并挂载章节目录。
  - 返回决策节点，进入下一轮评估。

3. **内容生成**
   - 当 `IndustryResearchFlow` 判断 "**信息已充足**" 时，`GenerateSection` 调用大模型读取当前章节全部知识库内容，按 "**概述-数据驱动分析-小结**" 的结构生成一至三千字的Markdown文档，并嵌入`mermaid`图表。
   - 生成结果存入 `shared["generated_sections"]`，继续处理下一章。
4. **报告整合与后处理**
   - `CompleteReport`：汇总所有章节，生成执行摘要、目录、过渡段及总结建议。
   - `CheckMermaidSyntax`：自动校正/补全 mermaid 代码块，保证图表可渲染。
   - `ConvertToDocx`：调用 md2docx 将最终 Markdown（含图）转换为 Word 文档。

Get started

1. 安装了docker之后在本目录下执行以下代码

```
docker build -t <镜像名称>:<镜像版本号> .  # 构建镜像
docker run -it --name <容器名称> <镜像名称>:<镜像版本号>  # 启动容器
```

2. 进入容器后执行以下脚本：

```
./run.sh
```

或是自定义参数直接调用python脚本：

```
python3 run_marco_research_report.py \
    --marco_name <宏观领域> \
    --time <时间>
```

3. 执行完毕后会在当前目录下生成 `Macro_Research_Report.docx`