舆情分析智能体系统 (Opinion Analysis Agent System)

本项目是一个基于 **GLM (智谱AI)** 大模型的高级舆情分析系统，采用 **PocketFlow** 框架进行编排。系统能够自动对大规模社交媒体博文（特别是突发事件，如极端天气、火灾等）进行深度分析，从数据增强、多维指标统计到最终生成图文并茂的分析报告，实现了全流程自动化。

🚀 核心工作流与运行机制

本项目采用 **中央调度模式 (Central Dispatcher)**，由 `DispatcherNode` 根据配置的 `run_stages` 参数动态决定执行路径。整个流程分为三个顺序依赖的阶段：

Phase 1: 数据增强 (Data Enhancement)

目标：将原始非结构化文本转化为结构化数据。

系统提供两种运行模式，适用于不同数据规模：

1. Async 异步并发模式 (适合 < 2000 条数据)

- **机制**：利用 Python `asyncio` 和 `AsyncBatchNode` 实现高并发处理。
- **并发控制**：通过信号量 (Semaphore) 限制最大并发数，防止 API Rate Limit。
- **断点续传**：内置 Checkpoint 机制，每隔固定时间或条数自动保存进度，任务中断无需重头开始。
- **处理节点**：
  - `AsyncSentimentPolarityAnalysisBatchNode`: 调用 **glm-4v-plus** (多模态) 或 **glm-4.5-air** 判断情感极性 (1-5分)。
  - `AsyncSentimentAttributeAnalysisBatchNode`: 提取细粒度情绪标签（如：愤怒、焦虑）。
  - `AsyncTwoLevelTopicAnalysisBatchNode`: 进行一级与二级主题分类。
  - `AsyncPublisherDecisionAnalysisBatchNode`: 识别发布者身份类型。

2. Batch API 批处理模式 (适合 > 10000 条数据)

- **机制**：完全绕过本地并发限制，利用智谱 Batch API 将任务上传至云端离线运行。
- **流程自动化** (`BatchAPIEnhancementNode`):
  1. **Generate**: 将待处理博文转换为 JSONL 请求文件。
  2. **Upload & Run**: 提交批量任务 ID 至服务器。
  3. **Monitor**: 轮询任务状态（云端并行度极高）。
  4. **Download & Integrate**: 下载结果行并解析，自动回填至原始 JSON 数据结构中。
- **优势**：成本降低 50%，吞吐量极大提升。

Phase 2: 深度分析 (Deep Analysis)

目标：将结构化数据转化为可视化图表，并提取出"数据无法直接体现"的深度洞察。

1. 工作流标准化分析 (Standardized Workflow)

通过 `ExecuteAnalysisScriptNode` 自动执行一系列 Python 脚本，生成标准化的核心指标图表：

- **时序分析**: 生成情感趋势图 (`sentiment_trend`)，识别舆情爆发点与消退期。
- **地理空间分析**: 绘制全国热力图 (`geo_heatmap`) 与省份柱状图，定位舆论中心。
- **发布者画像**: 统计不同类型账号（官方/个人/营销号）的发声占比与影响力。
- **信念网络 (Deep Belief Network)**:
  - **逻辑**: 基于 NetworkX 构建图模型。
  - **节点**: 提取博文中的认知要素，分为"风险感知"、"归因信念"、"行动呼吁"三类。
  - **边**: 计算要素间的共现频率（Co-occurrence），形成带权重的无向图。
  - **产出**: 高分辨率拓扑结构图，直观展示公众"为什么愤怒"或"归因给谁"。

2. LLM 视觉分析 (Visual Analysis Node)

系统不仅仅是"生成"图片，而是让 Agent 真正"看懂"图片。

- **节点**: `ChartAnalysisNode`
- **模型**: **glm-4.5v** (Thinking / 视觉思考模型)。
- **机制**:
  - 将 Matplotlib 生成的每一个图表传入 VLM (Vision Language Model)。
  - **Prompt 策略**: 要求模型不仅仅描述图表，必须提取**"极值点 (Max/Min)"、"拐点 (Inflection Point)"、"异常值 (Outliers)"** 以及数据的整体**"分布偏态"**。
  - **输出**: 针对每张图生成一段 300-500 字的专业数据解读文本，作为后续报告生成的原子素材。

3. Agent 自主探索 (Agent Exploration) (可选)

- **模型**: **glm-4.6** (开启 `enable_reasoning=True`)。
- **思考-行动循环 (ReAct Loop)**:
  1. **观察**: Agent 发现"河南"地区的负面情绪异常高。
  2. **思考**: "我需要探查河南地区具体在讨论什么话题。"
  3. **工具调用**: 动态选择 `analyze_region_topics(region='henan')` 工具。
  4. **结果反馈**: 发现该地区主要讨论"封路"与"大学生"话题。
  5. **结论**: 自动将此发现补充进洞察库。

Phase 3: 智能报告生成 (Iterative Reporting)

目标：模拟顶级分析师团队的"撰稿-审核-修订"闭环，彻底解决大模型"幻觉"与"车轱辘话"问题。

1. 基于资产的撰稿 (Asset-Based Generation)

- **节点**: `GenerateReportNode`
- **上下文注入**: Prompt 中不包含模糊的指令，而是注入了精确的**资产清单 (Asset List)**:
  - `Available_Charts`: `[{id: "sentiment_trend", path: "./images/..."}]`
  - `Verified_Insights`: 来自 Phase 2 的经校验过的结论。
- **强约束**: 模型被给予"零容忍"指令——**严禁编造数据**，所有结论段落必须引用资产清单中的图表（如 (如图 1-1 所示)）。

2. 批判性评审 (The "Critic" Review)

- **节点**: `ReviewReportNode`
- **模型**: **glm-4.6** (Reasoning Mode)。

- **评审维度**: 扮演苛刻的审核员，基于以下维度打分 (0-20分)：
    1. **Data Support (数据支撑度)**: 每一句断言是否有对应的数据/图表支持？识别并标记出“毫无根据的推测”。
    2. **Logic (逻辑链条)**: 结论是否由数据自然推导得出？是否存在逻辑跳跃？
    3. **Visualization (图表引用)**: 图片是否插入到了正确的位置？Markdown 语法是否有效？
- **输出**: 生成一份结构化的 JSON 评审报告，包含具体的 `unsupported_conclusions` (缺乏支撑的结论列表) 和 `revision_suggestions` (修改建议)。

3. 闭环迭代 (Closed-Loop Refinement)

- **节点**: `ApplyFeedbackNode` -> `GenerateReportNode`
- **流程**:
    - 如果评审总分 < **80分**（可配置阈值），系统拒绝发布报告。
    - 将“评审意见”和“当前草稿”一并回传给生成模型。
    - 生成模型进行**针对性重写**（例如：删除被标记为无据的段落，或补充缺失的图表引用）。
- **终止条件**: 评分达标 或 达到最大迭代次数 (Max Iterations, 默认 5 次)。

项目结构说明

```
AnalysisPosts/
├── main.py                 # 系统入口：配置 Shared Context，初始化 Dispatcher
├── flow.py                 # PocketFlow 编排文件：定义节点间的流转逻辑
├── nodes.py                # 核心逻辑实现：包含所有 Node 类的具体代码（Prompt 编写、
API 调用）
├── batch/                  # Batch API 独立模块
│   ├── batch_run.py        # 批处理主控脚本
│   ├── generate_jsonl.py   # 数据格式转换器
│   └── utils/              # Batch 专用工具
├── data/                   # 数据仓库
│   ├── posts.json          # 原始输入
│   ├── enhanced_blogs.json # Stage 1 产出的结构化数据
│   └── topics.json         # 主题体系定义
├── report/                 # 产出物
│   ├── report.md           # 最终生成的分析报告
│   ├── images/             # 生成的所有统计图表
│   └── analysis_data.json  # 中间分析结果
└── utils/
    ├── call_llm.py         # GLM 模型调用封装（支持 4.5-air, 4v, 4.6）
    ├── analysis_tools/     # Python 绘图工具集（Matplotlib/Seaborn/NetworkX）
    └── mcp_server/         # Model Context Protocol 工具定义
```

快速开始

1. 环境配置

确保 Python >= 3.10，安装依赖：

```
pip install -r requirements.txt
```

重要：请在 `utils/call_llm.py` 中配置您的智谱 AI API Key，或者设置环境变量 `ZHIPU_API_KEY`。

2. 数据准备

将待分析数据（JSON 格式列表）放置于 `data/` 目录。

3. 系统配置

修改 `main.py` 中的 `init_shared` 配置：

```python
def init_shared(...):
    return {
        # ...
        "input_data_path": "data/your_posts.json",

        # 选择运行模式
        "enhancement_mode": "async",  # 小数据用 async，大数据用 batch_api
        "analysis_mode": "workflow",  # 推荐 workflow 模式以获得稳定图表
        "report_mode": "iterative",   # 开启迭代模式以获得高质量报告

        # 控制运行阶段
        "run_stages": [1, 2, 3],
    }
```

4. 启动分析

```
python main.py
```

终端将实时显示当前运行的节点、阶段进度以及 Agent 的思考过程。

✨ 创新点总结

1. **双模态增强架构**: 实现了 `asyncio` 实时并发与 `Batch API` 离线批处理的无缝切换，兼顾了灵活性与成本效率。
2. **基于视觉的图表分析**: 不仅仅是生成图表，更利用多模态大模型（GLM-4v）“看懂”图表，实现了真正的“图文对齐”分析。
3. **信念网络 (Belief Network)**: 独创性地将非结构化舆论转化为图网络结构，量化分析公众信念的演化与共现。
4. **Self-Reflective Reporting**: 通过“生成-评审-修改”的 Agent 闭环，显著减少了 LLM 幻觉，提升了报告的专业度与数据准确性。