**1. main.py - 主入口文件**

**主要函数**:

* run\_tests(test\_file\_path, enable\_web\_search): 运行测试用例，对测试文件中的新闻进行检测
* main(): 程序入口，解析命令行参数并决定是启动Web界面还是运行测试

main.py作为入口文件，支持通过命令行参数控制程序行为，可以选择启动Web界面或运行测试用例。

**2. config.py - 配置文件**

不包含函数，但定义了系统关键配置变量：

* LLM\_API\_CONFIG: 大模型API配置（URL、Key、模型名等）
* GOOGLE\_SEARCH\_CONFIG: Google搜索API配置
* THRESHOLDS: 检测阈值配置（情绪词比例、语法错误率、风险阈值等）
* WEB\_CONFIG: Web应用配置（标题、描述、主题等）

**3. src/core/detector.py - 检测核心**

**主要类**: FakeNewsDetector**主要方法**:

* \_\_init\_\_(enable\_web\_search, google\_api\_key, search\_engine\_id): 初始化检测器组件
* preprocess\_news(news\_data): 预处理新闻数据，提取域名等信息
* \_adjust\_rules\_weights(result): 根据情况动态调整规则权重，特别是处理无法验证的规则
* detect(news\_data): 执行完整的虚假新闻检测流程
* \_recalculate\_risk\_with\_weights(result, weights): 根据调整后的权重重新计算风险百分比
* validate\_with\_web\_search(news\_data, forced\_by\_cutoff): 执行网络搜索验证

这个文件是系统的核心，实现了完整的检测流程，包括调用大模型、解析结果、动态调整权重、网络验证等。

**4. src/rules/detection\_rules.py - 检测规则定义**

**主要变量**:

* KNOWLEDGE\_CUTOFF\_DATE: 大模型知识截止日期
* DETECTION\_RULES: 六项检测规则的详细定义数组

**主要函数**:

* extract\_date\_from\_content(content): 尝试从新闻内容中提取日期信息
* get\_combined\_prompt(news\_data): 生成包含所有规则的完整提示词，提供给大模型

该文件定义了六大检测规则的详细内容和对应的提示词模板，是系统的核心规则引擎。

**5. src/models/llm\_connector.py (可能内容)**

**主要类**: LLMConnector**主要方法**:

* \_\_init\_\_(): 初始化连接器，加载API配置
* get\_response(prompt): 调用大模型API获取回答
* \_parse\_response(response): 解析API返回的响应

此文件负责与大模型API的交互，封装了API调用细节，处理错误和重试逻辑。

**6. src/utils/text\_analyzer.py (可能内容)**

**主要类**: TextAnalyzer**主要方法**:

* extract\_domain(url): 从URL中提取域名
* extract\_keywords(text): 从文本中提取关键词
* analyze\_sentiment(text): 分析文本情感倾向
* calculate\_similarity(text1, text2): 计算两段文本的相似度

此文件包含文本分析的通用功能，为检测规则提供支持。

**7. src/utils/result\_parser.py (可能内容)**

**主要类**: ResultParser**主要方法**:

* parse\_model\_response(response): 解析大模型返回的文本，提取规则应用结果
* extract\_rules\_results(text): 从文本中提取每条规则的判断结果
* extract\_conclusion(text): 提取总体结论和风险百分比

负责解析大模型的文本响应，将其转换为结构化的检测结果。

**8. src/utils/web\_search.py (可能内容)**

**主要类**: WebSearchValidator**主要方法**:

* \_\_init\_\_(api\_key, search\_engine\_id): 初始化搜索验证器
* search(query): 执行Google搜索
* validate\_news(news\_data): 验证新闻与搜索结果的一致性
* analyze\_results(search\_results, news\_data): 分析搜索结果与新闻的关系
* calculate\_consistency(news\_content, search\_results): 计算内容一致性得分

实现网络搜索验证功能，通过Google API查询相关信息并分析一致性。

**9. src/web/app.py - Web界面**

**主要函数**:

* create\_app(enable\_web\_search): 创建Gradio Web应用
* detect\_news(title, content, url): 界面调用函数，执行检测并格式化显示结果
* launch\_app(enable\_web\_search, port): 启动Web应用服务

此文件使用Gradio框架构建了友好的Web界面，展示检测结果和详细分析。

**执行流程详解**

1. **用户输入新闻** → Web界面收集标题、内容、URL
2. **预处理数据** → 提取域名、日期等
3. **构建提示词** → 根据规则模板生成提示词
4. **调用大模型** → 发送提示词给大模型API
5. **解析结果** → 结构化大模型返回的文本
6. **动态调整权重** → 处理无法验证的规则
7. **风险计算** → 计算风险百分比和风险等级
8. **网络验证** (可选) → 对高风险新闻进行网络搜索验证
9. **结果展示** → 格式化输出规则分析、风险评估和验证结果

是一套基于大模型的结构化提示词工程方法，通过设计的问答模板引导大模型根据六大规则进行分析，然后结合网络搜索进行交叉验证，最终提供较为全面且可靠的分析结果。