**模块详细介绍**

**模块 1：布局分析 (layout\_analysis.py)**

**职责**：处理矩形的空间布局分析，包括行分组、无效间隔过滤和距离计算。  
**核心功能**：

1. **行分组 (sort\_fixed\_area\_into\_lines)**
   * 按矩形的垂直坐标（y0）分组，同组内按水平坐标（x0）排序，实现 “行” 的识别。
   * **输入**：原始矩形字典（键为标识，值为坐标(x0, y0, x1, y1)）。
   * **输出**：按行分组的矩形列表，如 {y0: [(key, rect), ...], ...}。
2. **围栏过滤 (filter\_fences)**
   * 过滤行内水平间隙过大的矩形组，移除稀疏行（矩形数 < 2）。
   * **参数**：x\_threshold（水平间隙阈值，默认 50）、y\_threshold（垂直间隙阈值，默认 30）。
   * **作用**：净化布局，排除孤立元素干扰。
3. **平均距离计算 (calculate\_average\_distance)**
   * 计算同一行相邻矩形的平均水平距离，用于关系推断中的距离度量。
   * **输出**：平均距离值（默认 50，无有效距离时返回）。

**与其他模块的交互**：

* 为主模块提供分组后的矩形行数据和平均距离，供后续表格分析和关系推断使用。

**模块 2：表格结构分析 (table\_structure.py)**

**职责**：专门处理表格结构的检测，包括网格线识别、单元格划分、表头和合并单元格检测。  
**核心功能**：

1. **网格线检测 (detect\_grid\_lines)**
   * 从矩形边界提取垂直和水平网格线，合并相近坐标（阈值默认 5）。
   * **输出**：(vertical\_lines, horizontal\_lines)，如 ([x0, x1, ...], [y0, y1, ...])。
2. **单元格划分 (partition\_cells)**
   * 基于网格线将布局划分为单元格，每个单元格包含落入其中的矩形。
   * **输出**：单元格列表，每个元素包含坐标、矩形列表、行号和列号。
3. **表头识别 (identify\_header\_and\_data)**
   * 假设第一行为表头，通过格式特征（加粗、字体大小 > 12）区分表头和数据。
   * **输出**：(headers, data)，分别为表头单元格和数据单元格列表。
4. **合并单元格检测 (detect\_merged\_cells)**
   * 查找无矩形但属于合并区域的位置，通过向右 / 向下扩展检测合并范围。
   * **输出**：合并单元格列表，每个元素包含起止行和列。

**与其他模块的交互**：

* 为主模块提供表格结构信息（表头、数据、合并单元格），用于关系推断中的表格规则计算。

**模块 3：关系推断 (relationship\_inference.py)**

**职责**：整合布局和表格信息，通过多策略加权计算矩形间的逻辑关系。  
**核心功能**：

1. **多策略置信度计算**
   * **位置关系**：右侧、包含、垂直重叠规则，输出置信度（0-0.9）。
   * **格式特征**：字体加粗、大小、颜色匹配规则，输出置信度（0-0.6）。
   * **距离度量**：水平距离归一化，距离越近置信度越高（0-1）。
   * **表格结构**：同一单元格、同行列、表头 - 数据关系，输出置信度（0-0.9）。
2. **加权综合置信度 (find\_lower\_right\_rects)**
   * 按策略权重（位置 0.6、格式 0.3、距离 0.2、表格 0.3）计算总置信度，过滤低于阈值（0.3）的关系。
   * **输出**：矩形层次结构字典，如 {key\_i: [key\_j, key\_k, ...]}，按置信度降序排列。

**与其他模块的交互**：

* 依赖布局分析的矩形行和平均距离，以及表格结构的单元格映射，输出最终关系层次。

**主模块：逻辑搜索入口 (logic\_search.py)**

**职责**：作为项目入口，串联所有子模块，提供统一的逻辑搜索接口。  
**核心流程**：

1. **布局分析**：调用 layout\_analysis 进行行分组和围栏过滤。
2. **表格分析**：调用 table\_structure 检测网格线、划分单元格、识别表头。
3. **关系推断**：调用 relationship\_inference，结合布局和表格信息计算层次结构。
4. **结果输出**：返回包含层次结构和表格结构的字典。

**对外接口**：

* search(fixed\_area, format\_info=None)
  + **参数**：fixed\_area（矩形字典）、format\_info（格式信息字典，键为矩形标识）。
  + **返回**：{"hierarchy": 层次结构, "table\_structure": 表格结构}。

**模块化的优势总结**

1. **清晰的职责划分**：每个模块仅负责单一领域，如布局、表格、关系推断，降低认知复杂度。
2. **易于调试和测试**：可单独测试每个模块（如 layout\_analysis 的行分组逻辑），定位问题更高效。
3. **支持增量开发**：如需优化表格识别逻辑，只需修改 table\_structure.py，不影响其他模块。
4. **代码复用性**：子模块可被其他项目或功能复用（如单独使用表格结构分析）。

**对比差别**

| **对比维度** | **原代码（1 代码）** | **修改后代码（模块化版本）** |
| --- | --- | --- |
| **代码结构** | 单文件，4 个函数直接定义在一起 | 拆分为 4 个模块：主模块 + 3 个子模块 |
| **功能范围** | 仅实现行内矩形层次分析（基于位置关系） | 扩展支持表格结构分析、表头识别、合并单元格检测等 |
| **关系推断** | 仅考虑位置关系（右侧且 y1 更低） | 多策略加权（位置、格式、距离、表格结构） |
| **表格处理** | 无表格结构分析功能 | 新增完整的表格结构分析模块 |
| **扩展性** | 难以添加新功能（如格式分析、表格识别） | 可通过新增子模块轻松扩展功能 |
| **导入方式** | 无模块导入，所有函数直接调用 | 使用相对导入（.module）明确依赖关系 |