Solstice: LLM-Orchestrated System for Medical Document Fact-Checking

An AI-Native Approach to Evidence Extraction and Verification

1 Introduction

Solstice is a multi-step system that automatically fact-checks medical claims against scientific literature and clinical documents. It combines layout detection, multimodal language models, and an orchestrated chain of LLM calls to extract and verify evidence from PDFs that contain text, tables, and figures.

The system is designed to handle real-world medical documentation at scale, processing complex clinical documents containing text, tables, and figures.

2 System Architecture

2.1 Document Ingestion Pipeline

The ingestion pipeline transforms unstructured PDFs into queryable structured documents through multiple stages:

1. **Layout Detection**: Uses Detectron2 with PubLayNet-trained models (Mask R-CNN + ResNet-50-FPN). The pipeline relies on pre-trained weights rather than custom training because PubLayNet provides extensive annotated data.
2. **Box Consolidation**: Merges overlapping layout detections and resolves conflicts. Medical PDFs often contain complex overlapping elements that require special handling.
3. **Text Extraction**: Uses PyMuPDF for vector text extraction within bounding boxes.
4. **Figure/Table Extraction**: Saves visual elements as PNG at 400 DPI. Images are stored separately for later multimodal analysis rather than embedded as base64.
5. **Reading Order**: Computes reading order with column detection and vertical positioning; this is necessary for the multi-column layouts common in medical journals.

2.2 LLM-Based Fact-Checking System

The fact-checking pipeline processes text evidence through three sequential stages (Extract → Check Completeness → Verify) followed by parallel image analysis:

- **Evidence Extraction Step**:
  - *Inputs*: Document content JSON, claim text (from config)
  - *Processing*: Searches full document text for supporting quotes using gpt-4.1
  - *Outputs*: Array of `extracted_evidence` with quote text and relevance explanations
- **Completeness Checker**:
  - *Inputs*: Document content JSON, evidence extractor output
  - *Processing*: Re-searches document for quotes not found in initial extraction
  - *Outputs*: `combined_evidence` array merging existing and new quotes (duplicates removed)
- **Evidence Verification Step**:
  - *Inputs*: Document content JSON, completeness checker output
  - *Processing*: Verifies each quote exists in document (with OCR tolerance) and supports claim
  - *Outputs*: `verified_evidence` and `rejected_evidence` arrays with reasons

- **Image Evidence Analyzer**:
  - *Inputs*: Individual image file, image metadata (page, type), claim text
  - *Processing*: Analyzes visual content using o4-mini vision model
  - *Outputs*: Per-image analysis with `supports_claim` boolean, reasoning, and evidence found

After all LLM processing completes, an Evidence Presenter step consolidates all verified text and image evidence into structured JSON reports with coverage assessment.

2.3 LLM Pipeline Flow

The fact-checking pipeline orchestrates multiple LLM calls in a specific sequence:

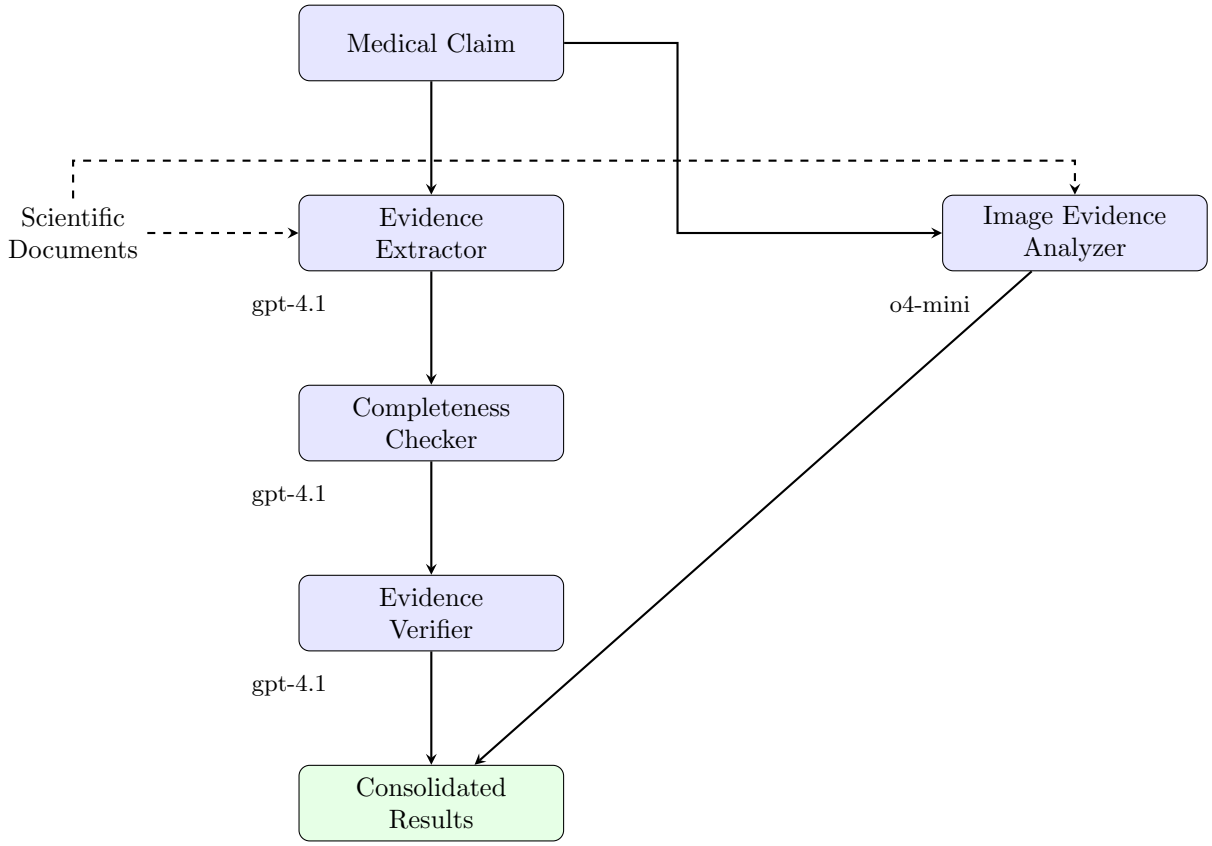

Figure 1: LLM Pipeline: Sequential text evidence processing (Extract→Completeness→Verify) with parallel image analysis. Each step receives the previous step's output plus the full document.

3 Technical Implementation

3.1 Core Services Architecture

Solstice is built on three logical services, with the Gateway service running in Docker for isolation:

- **Ingestion Service**: Handles PDF processing, layout detection, and text/figure extraction. Stores processed documents in the document cache.
- **Gateway Service**: Provides the HTTP API and acts as a proxy for all LLM requests. Runs in Docker to ensure consistent environment and dependencies.
- **Fact Check Service**: Orchestrates claim processing, manages the multi-step evidence analysis pipeline, and consolidates results into the final JSON format.

The following diagram illustrates how these services interact:

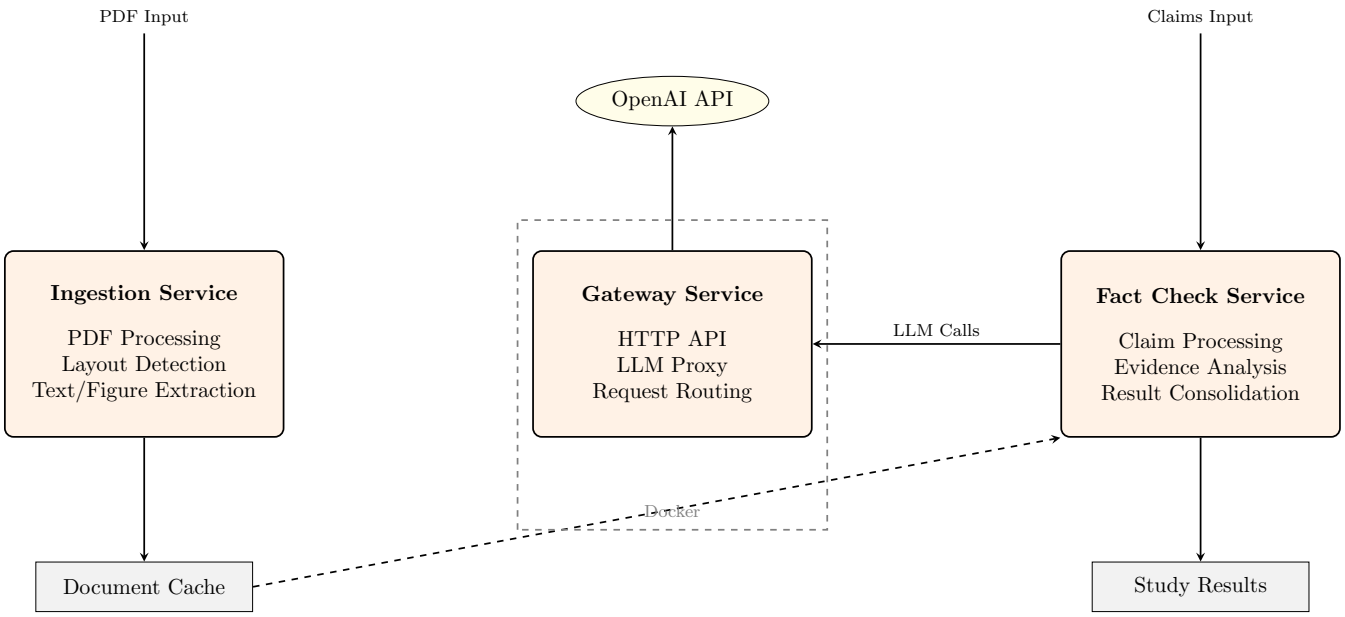

Figure 2: Core Services Architecture: Gateway runs in Docker as the central LLM interface

3.2 Orchestration Layer

The system uses asynchronous Python with strategic architectural decisions:

- **Hierarchical Orchestration**: Two-level design with StudyOrchestrator → ClaimOrchestrator → processing steps. Enables both study-level and claim-level parallelism control.
- **Step Isolation**: Each processing step runs independently with explicit input/output contracts via Pydantic models. Enables testing and development in isolation.

3.3 Where to Find the Data Artifacts

Solstice writes intermediate and final results to disk in human-readable form. This allows users to inspect the system, debug individual stages, or create visualisations without rerunning the full pipeline.

1. Fact-Checking Study Results After you run a fact-checking study (`python -m src.cli run-study`), the main results file containing all claims with their supporting text, tables, and figures is located at:

`data/studies/consolidated_results.json`

This single JSON file contains all the verified evidence for each claim. Here's the structure with a stubbed example:


```
{
  "study_name": "unknown",
  "timestamp": "2025-07-30T23:47:44.122369",
  "claims": [
    {
      "claim": "Flublok ensures identical antigenic match with
               WHO- and FDA-selected flu strains.",
      "evidence": [
        {
          "type": "text",
          "quote": "The primary amino acid sequence of the rHA
                   produced using baculovirus...is identical
                   to the HA from the wild type virus...",
```

```
          "explanation": "The quote directly supports the claim...",
          "document": "Arunachalam_et_al.__2021_"
        },
        {
          "type": "image",
          "filename": "figure_p2_det_1_000.png",
          "reasoning": "The table provides data showing Flublok
                        contains 45 µg HA per strain...",
          "document": "CDC_Influenza_vaccines"
        }
      ]
    },
    {
      "claim": "Flublok contains 3x the hemagglutinin (HA) antigen
               content of standard-dose flu vaccines...",
      "evidence": [
        // Additional text and image evidence entries
      ]
    }
    // Additional claims...
  ]
}
```

Each claim contains an array of evidence entries that can be either:

- **Text evidence**: Exact quotes from documents with explanations of how they support or refute the claim
- **Image evidence**: References to figures/tables with reasoning about their relevance

The evidence entries always include the source document name, making it easy to trace back to the original PDFs in the scientific or marketing cache.

2. Marketing Material Cache Marketing PDFs like FlublokOnePage are processed with special attention to layout preservation. To see the marketing page layout, look at:

```
data/marketing_cache/FlublokOnePage/visualizations/page_001_layout.png
```

This visualization shows the detected layout elements (text blocks, figures, headers) overlaid on the original page, helping verify proper extraction of complex marketing layouts.

The full directory structure for marketing materials:

```
data/marketing_cache/FlublokOnePage/
|-- extracted/
|   |-- content.json          # Extracted text and layout structure
|   |-- document.html         # HTML version of the document
|   |-- document.md           # Markdown version
|   |-- document.txt          # Plain text version
|   `-- figures/
|       |-- figure_p1_04314352.png
|       |-- figure_p1_6de74a29.png
|       `-- figure_p1_d9d91bd1.png
|-- pages/
|   `-- page-000.png          # Full page image
`-- visualizations/
    `-- page_001_layout.png   # ** MARKETING LAYOUT VISUALIZATION **
```

3. Scientific Document Cache Scientific papers undergo the same extraction process but may include additional agent processing for fact-checking. For example, Treanor_et_al.__2011_ contains:

```
data/scientific_cache/Treanor_et_al.__2011_/
```

```
|-- extracted/
|   |-- content.json          # Extracted text and layout structure
|   |-- document.html         # HTML version
|   |-- document.md           # Markdown version
|   |-- document.txt          # Plain text version
|   `-- figures/              # Extracted figures and tables
|       |-- figure_p1_det_0_005.png
|       |-- figure_p3_det_2_000.png
|       |-- table_p3_det_2_009.png
|       `-- table_p4_det_3_003.png
|-- pages/                    # Individual page images
|   |-- page-000.png
|   |-- page-001.png
|   `-- ...
|-- visualizations/           # Layout detection overlays
|   |-- all_pages_summary.png
|   |-- page_001_layout.png
|   `-- ...
|-- raw_layouts/              # Intermediate layout data
|   `-- raw_layout_boxes.json
`-- agents/                   # Fact-checking results (if processed)
    `-- claims/
        `-- claim_000/
            |-- evidence_extractor/
            |-- completeness_checker/
            `-- evidence_verifier/
```

The key difference is that scientific documents may have an `agents/` directory containing fact-checking results for individual claims.

4. Quick Directory Cheatsheet For newcomers, these are useful places to explore:

- `src/cli/` – entry-point scripts that orchestrate the pipelines.
- `src/injestion/shared/processing/` – layout detection, reading-order logic, text extractors.
- `src/fact_check/agents/` – implementation of the four specialised LLM processing steps plus formatting.
- `data/scientific_cache/` – processed scientific PDFs (inspect `content.json`).
- `data/marketing_cache/` – processed marketing PDFs.
- `data/studies/` – end-to-end fact-checking results ready for consumption.

4 Implementation Notes

4.1 Model Integration and Multimodal Processing

The system uses different models for text and image analysis, reflecting a key architectural decision to process these modalities separately:

4.1.1 Text Processing with gpt-4.1

All text-based evidence extraction and verification uses gpt-4.1, chosen specifically for its needle-in-haystack capabilities:

- **Evidence extraction**: Searches through hundreds of pages to find exact quotes matching claims
- **High-precision retrieval**: Locates specific sentences in dense medical literature with minimal false positives
- **Completeness checking**: Re-scans documents to catch any quotes missed in the first pass
- **Quote verification**: Confirms extracted quotes exist in source documents (with tolerance for OCR corrections) and actually support the claims

4.1.2 Image Analysis with o4-mini

Visual evidence (figures, tables, charts) is processed using o4-mini, a reasoning model chosen for its ability to interpret complex visual information:

- **Visual reasoning**: Interprets data trends, statistical significance, and relationships shown in charts
- **Table comprehension**: Understands structured data and can identify relevant rows/columns supporting claims
- **Diagram analysis**: Reasons about flowcharts, mechanisms, and visual explanations of processes
- **Contextual interpretation**: Understands what visual evidence actually demonstrates in context

4.1.3 Why Separate Text and Image Processing?

Processing text and images separately leverages the distinct strengths of different model types:

1. **Model specialization**: gpt-4.1 excels at needle-in-haystack tasks—finding specific quotes in massive documents—while reasoning models like o4-mini are better at interpreting visual information and drawing conclusions
2. **Task optimization**: Text evidence extraction is fundamentally a search and retrieval problem where precision matters most. Image analysis requires reasoning about relationships, trends, and visual patterns
3. **Prompt engineering**: Extraction tasks benefit from deterministic responses while visual reasoning tasks need more flexible interpretation
4. **Performance characteristics**: gpt-4.1 can process enormous text contexts efficiently for quote extraction, while reasoning models handle the cognitive load of understanding charts, diagrams, and data visualizations

4.1.4 Technical Implementation

- All LLM calls route through an HTTP gateway service for centralized request handling
- Images are stored as files (not base64) for easier inspection and debugging
- Parallel image processing is limited to prevent overwhelming the API
- Both pipelines use structured JSON schemas for predictable output formats

4.2 Three-Stage Text Evidence Pipeline

Extraction, completeness checking, and verification are run as distinct LLM-driven steps:

- **Stage 1 - Extract**: Find all potentially relevant quotes from the document (optimized for high recall)
- **Stage 2 - Expand**: Search again for additional quotes not caught initially (maximize coverage)
- **Stage 3 - Verify**: Check each quote exists in document and genuinely supports claim (ensure precision)

Each stage produces structured JSON output that feeds into the next:

- **Extractor→Completeness**: Passes `extracted_evidence` array
- **Completeness→Verifier**: Passes `combined_evidence` array
- **Verifier→Presenter**: Passes `verified_evidence` array

This separation enables independent optimization, testing, and debugging of each stage.

4.3 Filesystem-Centric Architecture

The pipeline stores intermediate data on the filesystem rather than in a database:

- Human-readable JSON can be inspected without extra tools
- Hierarchical directories (document/claim/step) reflect the processing flow
- No additional infrastructure is required
- Files can be version-controlled in Git

5 Appendix: Layout Detection Examples

The following examples demonstrate Solstice's layout detection capabilities on real documents.

5.1 Scientific PDF Layout Detection

Figure 3 demonstrates the system handling a complex scientific page containing figures, tables, and dense academic text. This example from Arunachalam et al. (2021) shows successful detection of multiple distinct elements including data visualizations, methodology diagrams, and scientific text blocks.

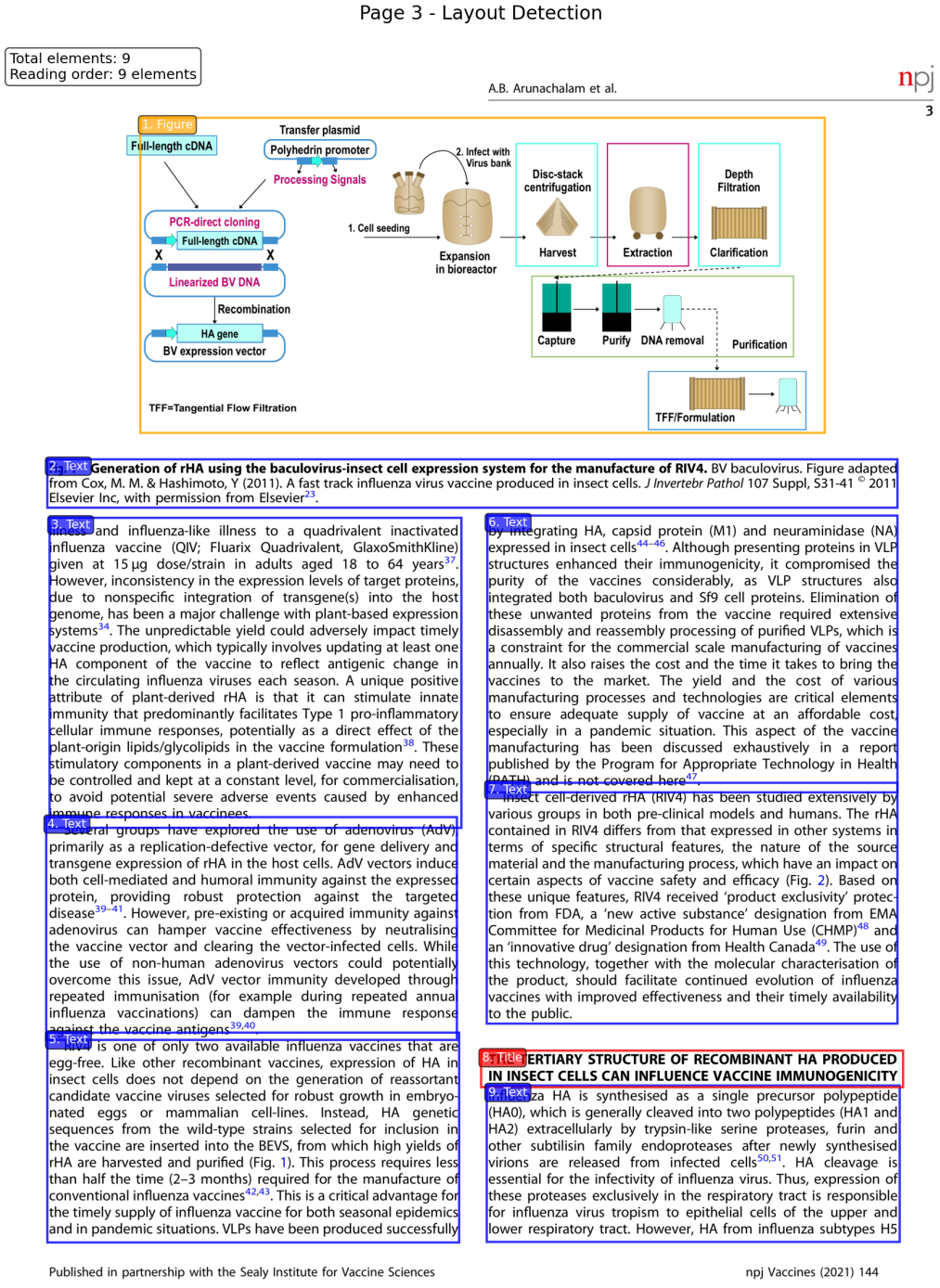

Figure 3: Layout detection on a complex scientific page (Arunachalam et al., 2021, page 3). The system correctly identifies and separates: (1) text blocks, (2) scientific figures, (3) figure captions, (4) methodology sections, and (5) data visualizations. The colored bounding boxes indicate different element types detected by the Detectron2 model.

5.2 Marketing Material Layout Detection

Figure 4 shows the layout detection on a pharmaceutical marketing page. The system identifies distinct marketing elements including headers, body text, and visual design components, preserving the intended visual hierarchy.

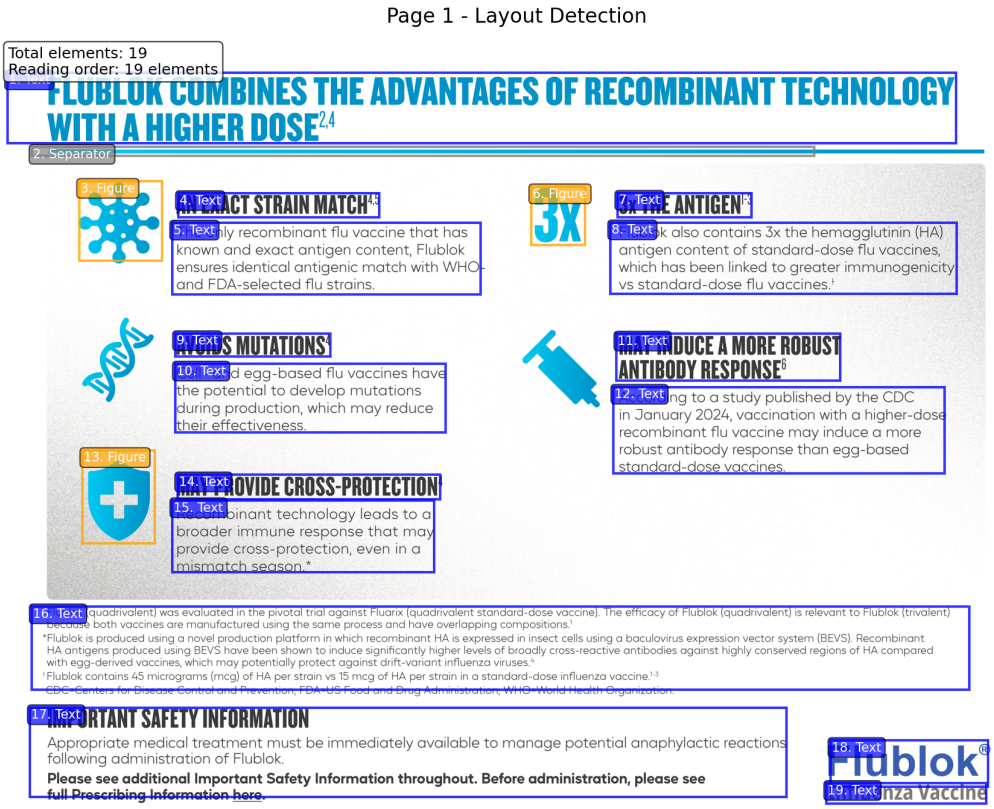

Figure 4: Layout detection on marketing material (FlublokOnePage). Different colors indicate detected elements: headers, body text, figures, and marketing-specific components. The detection preserves reading order and visual hierarchy essential for marketing documents.

These visualizations demonstrate the system's ability to handle diverse document types, from marketing materials with complex visual designs to dense scientific publications with mixed content types. The layout detection forms the foundation for accurate text extraction and evidence retrieval.