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 → Completeness → Verify) followed by parallel image analysis:

- **Evidence Extraction Step**: Searches document text for claim-relevant quotes using gpt-4.1 with temperature=0. Preserves exact quotes and returns structured evidence with relevance explanations.
- **Completeness Checker**: Takes the raw extracted evidence and searches for any additional quotes that weren't initially found. Merges evidence from multiple sources to ensure comprehensive coverage.
- **Evidence Verification Step (V2)**: Validates that all extracted quotes (from both extraction and completeness steps) exist in the source document. Uses semantic matching and filters out tangentially related content, achieving high verification rates.
- **Image Evidence Analyzer**: Analyzes figures and tables using vision models to identify supporting visual evidence. Processes images in parallel with semaphore control (max 5 concurrent) and provides detailed explanations.

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:

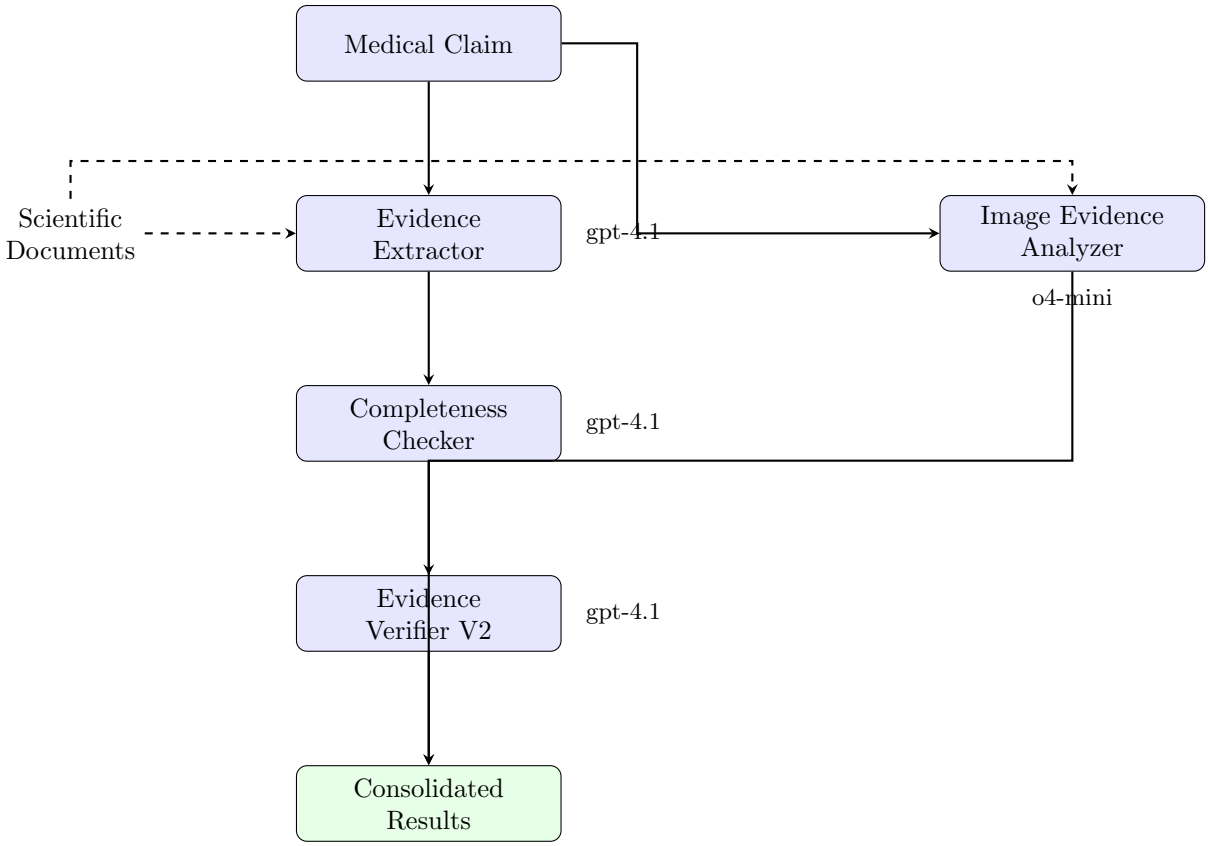

Figure 1: LLM Pipeline: Sequential text evidence processing with parallel image analysis

3 Technical Implementation

3.1 Core Services Architecture

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

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:

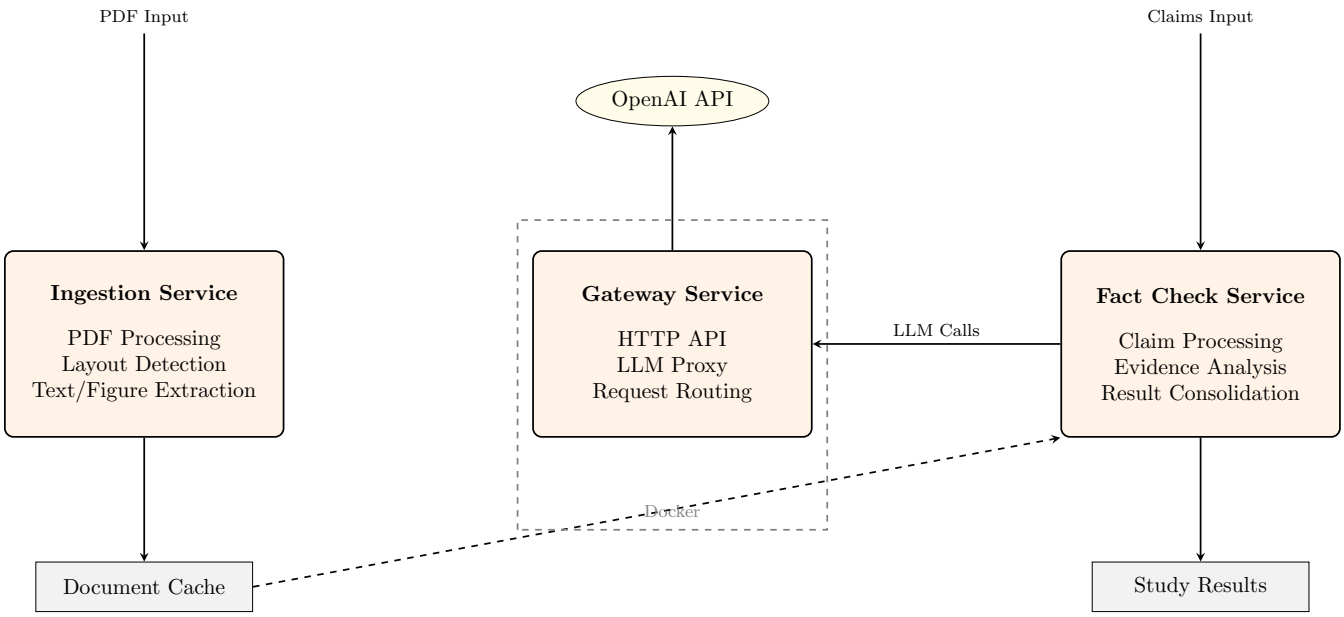

Figure 2: Core Services Architecture: Gateway runs in Docker as the central LLM interface


```
{
  "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_v2/
```

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 with temperature=0 to prioritize consistency over creativity. This model handles:
- Evidence extraction from document text
- Completeness checking to find missed quotes
- Verification that quotes exist and support claims
- Structured JSON output generation

4.1.2 Image Analysis with o4-mini

Visual evidence (figures, tables, charts) is processed separately using o4-mini vision models. This separation provides several advantages:
- **Optimized prompts**: Text and image analysis require fundamentally different prompting strategies
- **Parallel processing**: Text pipeline can complete while images are still being analyzed
- **Resource efficiency**: Vision model calls are more expensive; separation allows selective processing
- **Debugging clarity**: Failures in one modality don't affect the other

4.1.3 Why Separate Text and Image Processing?

Processing text and images separately, rather than using a single multimodal model, was a deliberate choice:
1. **Pipeline resilience**: If image analysis fails or times out, text evidence is still available
2. **Cost optimization**: Many claims can be verified from text alone; images are processed only when needed
3. **Specialized prompting**: Text extraction benefits from chain-of-thought reasoning, while image analysis needs spatial and visual reasoning
4. **Incremental development**: The text pipeline was built first and thoroughly tested before adding image support

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 5 concurrent calls via semaphore
- Both pipelines use structured JSON schemas for predictable output formats

4.2 Three-Stage Text Evidence Pipeline

Extraction, verification, and completeness are run as distinct LLM-driven steps:
- **Stage 1**: Extract all potentially relevant quotes (high recall)
- **Stage 2**: Find additional quotes not caught in initial extraction (expand coverage)
- **Stage 3**: Verify all quotes exist and support claim (high precision)

This separation allows the stages to be optimised and tested independently.

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 Conclusion

Solstice combines layout understanding, multimodal analysis, and orchestrated LLM calls to perform medical fact-checking. Tests on multiple documents and claims indicate that the architecture can process real-world medical documentation at moderate scale.