Solstice: Multi-Agent System for Medical Document Fact-Checking

An AI-Native Approach to Evidence Extraction and Verification

1 Introduction

Solstice is a sophisticated multi-agent system designed to automatically fact-check medical claims against scientific literature and clinical documents. By combining state-of-the-art layout detection, multimodal language models, and orchestrated agent pipelines, the system extracts and verifies evidence from complex medical PDFs containing 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 (Faster R-CNN + ResNet-50-FPN). Strategic choice: pre-trained models over custom training due to PubLayNet's 360k+ annotated medical pages achieving 95%+ mAP.
2. **Box Consolidation**: Custom algorithm resolves overlapping detections through IoU-based merging (threshold: 0.7) and hierarchical nesting. Key insight: medical PDFs have systematic layout overlaps requiring domain-specific consolidation rules.
3. **Text Extraction**: PyMuPDF for vector text extraction within bounding boxes, with fallback to Tesseract OCR for scanned pages. SymSpell correction handles common OCR artifacts (e.g., "rn" → "m").
4. **Figure/Table Extraction**: Saves visual elements as PNG at 300 DPI. Strategic decision: store images separately for multimodal analysis rather than inline base64 encoding.
5. **Reading Order**: Custom algorithm using column detection and vertical positioning. Critical for multi-column layouts common in medical journals.

2.2 Multi-Agent Fact-Checking System

The fact-checking pipeline employs specialized agents orchestrated to work together:

- **Evidence Extraction Agent**: Searches document text for claim-relevant quotes using GPT-4 with temperature=0. Preserves exact quotes while correcting OCR errors and returns structured evidence with relevance explanations.
- **Evidence Verification Agent (V2)**: Validates that extracted quotes exist in the source document. Uses semantic matching to handle OCR variations and filters out tangentially related content, achieving high verification rates.
- **Completeness Checker**: Takes the raw extracted evidence and searches for any additional quotes that weren't initially found. Uses the same extraction approach but explicitly excludes already-found quotes to expand coverage.
- **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.
- **Evidence Presenter**: Consolidates all verified text and image evidence into structured JSON and HTML reports. Assesses overall evidence coverage (complete/partial/none) and produces human-readable summaries.

3 Critical Design Decisions

3.1 Agent Communication Pattern

Chose filesystem-based communication over message passing:

- Each agent reads inputs from disk and writes outputs to disk
- No shared memory or direct agent-to-agent communication
- Benefits: debuggability, resumability, testability in isolation
- Trade-off: Disk I/O overhead acceptable for our throughput requirements

3.2 Prompt Strategy

Key decisions in prompt engineering:

- **Structured Output**: Always request JSON with explicit schemas
- **Few-Shot Examples**: Avoided in favor of clear instructions (reduced token usage)
- **Error Context**: Include previous failures in retry prompts
- **Temperature=0**: Consistency critical for medical facts

3.3 Caching Strategy

Multi-level caching approach:

- **Document Cache**: Extracted content never reprocessed
- **Agent Cache**: Each agent output stored by claim ID
- **No LLM Response Cache**: Deliberate choice to always get fresh responses
- Strategic insight: Caching at semantic boundaries, not API boundaries

4 Technical Implementation

4.1 Orchestration Layer

The system uses asynchronous Python with strategic architectural decisions:

- **Hierarchical Orchestration**: Two-level design with StudyOrchestrator $\rightarrow$ ClaimOrchestrator $\rightarrow$ Agents. Enables both study-level and claim-level parallelism control.
- **Resource-Aware Parallelism**: Default 2 concurrent claims based on empirical testing showing memory usage of 2GB per claim with GPT-4 context windows.
- **Filesystem-Based State**: All intermediate results persisted to disk, enabling resumability and debugging. Trade-off: disk I/O for reliability.
- **Agent Isolation**: Each agent runs independently with explicit input/output contracts via Pydantic models. Enables testing and development in isolation.

4.2 Model Integration

Strategic model selection based on empirical performance:

- **GPT-4 (Primary)**: Evidence extraction/verification with temperature=0 for consistency. Strategic choice: reliability over creativity for medical facts.
- **Vision Models**: GPT-4V for image analysis. Key decision: process images individually with focused prompts rather than batch processing.
- **Gateway Pattern**: All LLM calls through HTTP gateway service. Benefits: centralized rate limiting, cost tracking, and provider abstraction.
- **Prompt Engineering**: Structured output formats with explicit JSON schemas. Critical: removed token limits after discovering truncation issues.

4.3 Robust Error Handling

Multiple layers of reliability based on production experience:

- **Defensive JSON Parsing**: Custom parser handles markdown code blocks, trailing commas, and incomplete responses. Learned from: 15% of GPT-4 responses wrapped JSON in markdown.
- **Smart Retry Strategy**: Exponential backoff with error context injection. Key insight: including parsing errors in retry prompts improves success rate to 95%+.
- **Pydantic Validation**: Type-safe data flow between agents. Strategic choice: fail fast with clear errors rather than propagate bad data.
- **Context Window Management**: Removed max_tokens parameter after discovering 30% of evidence was truncated. Trade-off: higher token cost for completeness.

5 Key Technical Innovations

5.1 Multimodal Evidence Integration

Strategic approach to visual evidence: separate image extraction and analysis pipeline. Key decisions:

- Store images as files, not base64, enabling browser-based debugging
- Individual image analysis with focused prompts improves accuracy over batch processing
- Semaphore-limited parallel processing (max 5) prevents API rate limit issues

5.2 Three-Stage Evidence Pipeline

Architectural choice: separate extraction, verification, and completeness into distinct agents:

- **Stage 1**: Extract all potentially relevant quotes (high recall)
- **Stage 2**: Verify quotes exist and support claim (high precision)
- **Stage 3**: Find additional quotes not caught in initial extraction (expand coverage)

This separation enables independent optimization and testing of each stage.

5.3 OCR-Resilient Text Matching

Custom fuzzy matching algorithm addresses medical PDF challenges:

- Character-level substitution rules ("0"/"O", "1"/"l", "rn"/"m")
- Whitespace normalization for column-spanning text
- Semantic similarity fallback using sentence embeddings

5.4 Filesystem-Centric Architecture

Deliberate choice of filesystem over database for intermediate storage:

- Human-readable JSON enables debugging without tools
- Natural hierarchical organization (document/claim/agent)
- Zero infrastructure requirements
- Git-friendly for tracking changes

6 Conclusion

Solstice demonstrates the power of combining modern AI capabilities—layout understanding, multimodal analysis, and orchestrated agents—to tackle the complex challenge of medical fact-checking. Processing multiple documents with numerous claims has validated the architecture's robustness and scalability for real-world medical documentation.