# Invoice Processing Agent with RAG + Local LLM

**Version:** 2.0 - RAG Edition  
**Author:** r4 Technologies, Inc 2025  
**Notebook:** Invoice\_Processing\_Agent.ipynb

## Overall Goal

This notebook implements an end-to-end AI-powered invoice processing system that:

1. **Extracts invoice processing rules** from contract documents using RAG (Retrieval-Augmented Generation)
2. **Validates incoming invoices** against the extracted rules automatically
3. **Operates completely locally** using Ollama (no cloud API fees)

## Part 1: RAG-Powered Rule Extraction (Cells 0-17)

### Architecture & Logic

#### Setup Phase (Cells 0-9)

* **Cell 0:** Optional diagram extraction from reference PDF
* **Cells 1-4:** Documentation and requirements
* **Cell 5-6:** Package installation (document processing + RAG stack)
* **Cell 7:** Platform detection and environment configuration
* **Cell 8:** Import libraries (LangChain, FAISS, pdfplumber, pytesseract)
* **Cell 9:** Initialize and test Ollama models (gemma3:270m LLM + nomic-embed-text embeddings)

#### Helper Functions (Cells 10-11)

* **Cell 10:** Garbled text detection (validates OCR quality)
* **Cell 11:** Invoice term validation (checks for relevant keywords)

#### RAG Agent Implementation (Cells 12-17)

**Cell 12:** InvoiceRuleExtractorAgent class

* Parses PDF/DOCX contracts with OCR fallback
* Creates FAISS vector store from document chunks (800 chars, 200 overlap)
* Uses RAG to extract 4 rule categories:
  + **Payment terms:** Net days, PO requirements
  + **Approval process:** Who approves, timeframes
  + **Late penalties:** Fees, rejection criteria
  + **Submission requirements:** Required invoice fields

#### Rule Extraction Workflow

1. **Document Loading:** Parse contract (PDF/DOCX/images)
2. **Text Chunking:** Split into semantic chunks using RecursiveCharacterTextSplitter
3. **Indexing:** Create vector embeddings with FAISS
4. **Retrieval:** Find top-k relevant chunks per query (adaptive k)
5. **Generation:** LLM generates structured rules from context
6. **Refinement:** Structure rules into JSON format with metadata

#### Execution (Cells 14-17)

* **Cell 14:** Initialize agent
* **Cell 15:** Process contract and extract rules
* **Cell 16:** Save rules to JSON
* **Cell 17:** Display formatted rules

## Part 2: Invoice Processing Engine (Cells 18-24)

### Architecture & Logic

#### Invoice Processor Class (Cell 19)

**Input:** Extracted rules JSON file

**Capabilities:**

* **Parse invoices** (PDF, DOCX, PNG, JPG with pytesseract OCR)
* **Extract fields** using regex patterns:
  + Invoice number, PO number, dates, amounts, vendor info
* **Validate against contract rules:**
  + Check required fields (PO, dates, amounts)
  + Validate payment terms (Net 30/60 compliance)
  + Detect overdue invoices
  + Calculate late penalties
* **Make decisions:**
  + APPROVED: All rules satisfied → auto-approve
  + FLAGGED: Warnings (e.g., overdue) → review recommended
  + REJECTED: Missing required fields → manual review
  + ERROR: Processing failure → system error

#### Validation Logic

1. Check required fields (from submission\_requirements rule)
2. Validate payment terms (Net days calculation)
3. Check overdue status and apply penalties
4. Generate compliance report with issues/warnings

#### Execution (Cells 21-24)

* **Cell 21:** Initialize processor with rules
* **Cell 22:** Process single invoice
* **Cell 23:** Batch process folder of invoices
* **Cell 24:** Generate detailed processing report with statistics

## Part 3: Testing & Utilities (Cells 25-32)

### Sample Data Generation (Cell 26)

Creates realistic test data in multiple formats:

* **Contracts:** Net 30/60 day agreements (PDF, DOCX, PNG)
* **Invoices:** 6 test cases covering:
  1. Valid invoice (should approve)
  2. Missing PO (should reject)
  3. Overdue invoice (should flag)
  4. Recent invoice (should approve)
  5. OCR test - valid (PNG image, should approve)
  6. OCR test - no PO (PNG image, should reject)

### End-to-End Pipeline Test (Cell 28)

* Extracts rules from contract using RAG
* Processes all sample invoices
* Validates against extracted rules
* Generates success/failure metrics

### Results Export (Cell 29)

* Saves processing results to JSON
* Generates detailed report with:
  + Overall statistics (approval rate, rejection rate)
  + Most common issues
  + Most common warnings
  + Recommended actions
  + Invoice-by-invoice breakdown

### Verification (Cell 32)

* Displays current processor state
* Shows loaded rules
* Provides next-step guidance

## Technology Stack

**LLM & Embeddings**

* **Ollama gemma3:270m:** Lightweight local LLM (270M parameters)
* **nomic-embed-text:** Local embedding model

**Advantages:** No API costs, complete privacy, offline operation

**RAG Components**

* **FAISS:** Fast vector similarity search (CPU-optimized)
* **LangChain:** RAG orchestration framework
* **RecursiveCharacterTextSplitter:** Semantic chunking

**Document Processing**

* **pdfplumber:** PDF text extraction
* **python-docx:** MS Word parsing
* **pytesseract:** OCR for scanned documents
* **Pillow:** Image preprocessing

## Key Results & Performance

### Rule Extraction

* Successfully extracts 4 rule categories from contracts
* RAG retrieval ensures accuracy by providing relevant context
* Handles contracts of varying lengths (adaptive chunk retrieval)

### Invoice Processing - Test Results

**Total:** 6 invoices

**Approved:** 3 (50%)

**Flagged:** 1 (16.7%)

**Rejected:** 2 (33.3%)

**Errors:** 0 (0%)

### Validation Accuracy

| **Invoice** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- |
| invoice\_001\_valid.pdf | Approve | APPROVED | **✓ Correct** |
| invoice\_002\_no\_po.pdf | Reject (missing PO) | REJECTED | **✓ Correct** |
| invoice\_003\_overdue.pdf | Flag (overdue) | FLAGGED | **✓ Correct** |
| invoice\_004\_recent.pdf | Approve | APPROVED | **✓ Correct** |
| invoice\_005\_ocr\_valid.png | Approve (OCR test) | APPROVED | **✓ Correct** |
| invoice\_006\_ocr\_no\_po.png | Reject (OCR + missing PO) | REJECTED | **✓ Correct** |

### OCR Performance

* ✓ Successfully processed PNG invoice images
* ✓ Extracted all required fields from scanned documents
* ✓ Applied same validation rules as digital PDFs

## Business Value

1. **Automation:** Reduces manual invoice review by 50%+ (auto-approves compliant invoices)
2. **Accuracy:** Rule-based validation eliminates human error
3. **Speed:** Processes invoices in seconds vs. minutes manually
4. **Compliance:** Ensures all invoices meet contract terms
5. **Audit Trail:** Complete JSON logs of all decisions
6. **Cost Savings:** Free local LLM vs. paid API services
7. **Privacy:** All processing happens locally (no data leaves machine)

## Execution Flow

### Quick Start

1. Run Cells 5-9: Install & initialize
2. Run Cell 26: Create sample documents
3. Run Cell 28: Test complete pipeline

### Step-by-Step

1. Run Cells 5-9: Setup
2. Run Cell 26: Generate test data
3. Run Cells 14-17: Extract rules from contract
4. Run Cells 21-24: Process invoices
5. Review results and reports

## Conclusion

The notebook demonstrates a **production-ready AI agent system** combining RAG, local LLMs, and intelligent automation for real-world invoice processing workflows. It showcases how modern AI techniques can be applied to automate complex business processes while maintaining privacy, accuracy, and cost-effectiveness.

Generated on:

r4 Technologies, Inc 2025