Intelligent Invoice Line Items Extraction and Retrieval System

Yassin Abdulmahdi

January 2026

Abstract

This report details the development of an automated pipeline for extracting, structuring, and retrieving line-item data from PDF invoices. By leveraging Retrieval-Augmented Generation (RAG), the system ingests unstructured PDF documents, converts them into structured tables using Docling, stores them in a LanceDB vector database, and utilizes the Llama-3 Large Language Model (LLM) to answer natural language user queries.

1 Introduction

Processing invoices manually is time-consuming and prone to human error. The goal of this project is to build an intelligent system capable of ingesting PDF invoices and allowing users to query specific details (e.g., "What is the unit price of Item X?") using natural language. The project repository can be found at:
`https://github.com/Yassin522/Intelligent-Invoice-Line-Items-Extraction-Retrieval-Syst`
`tree/main`

2 System Architecture

The system follows a linear Extract-Transform-Load (ETL) pipeline followed by a RAG inference loop.

2.1 Data Flow

1. **Ingestion:** PDF invoices are processed using *Docling*, which specializes in extracting table structures from documents.
2. **Stitching & Transformation:** Raw tables are logically stitched together, and rows are serialized into text chunks to preserve context.
3. **Vector Storage:** Enriched chunks are embedded and stored in *LanceDB* for semantic search.
4. **Inference:** A user query triggers a semantic search. Top-k relevant context chunks are retrieved and passed to *Llama-3* to generate a grounded answer.

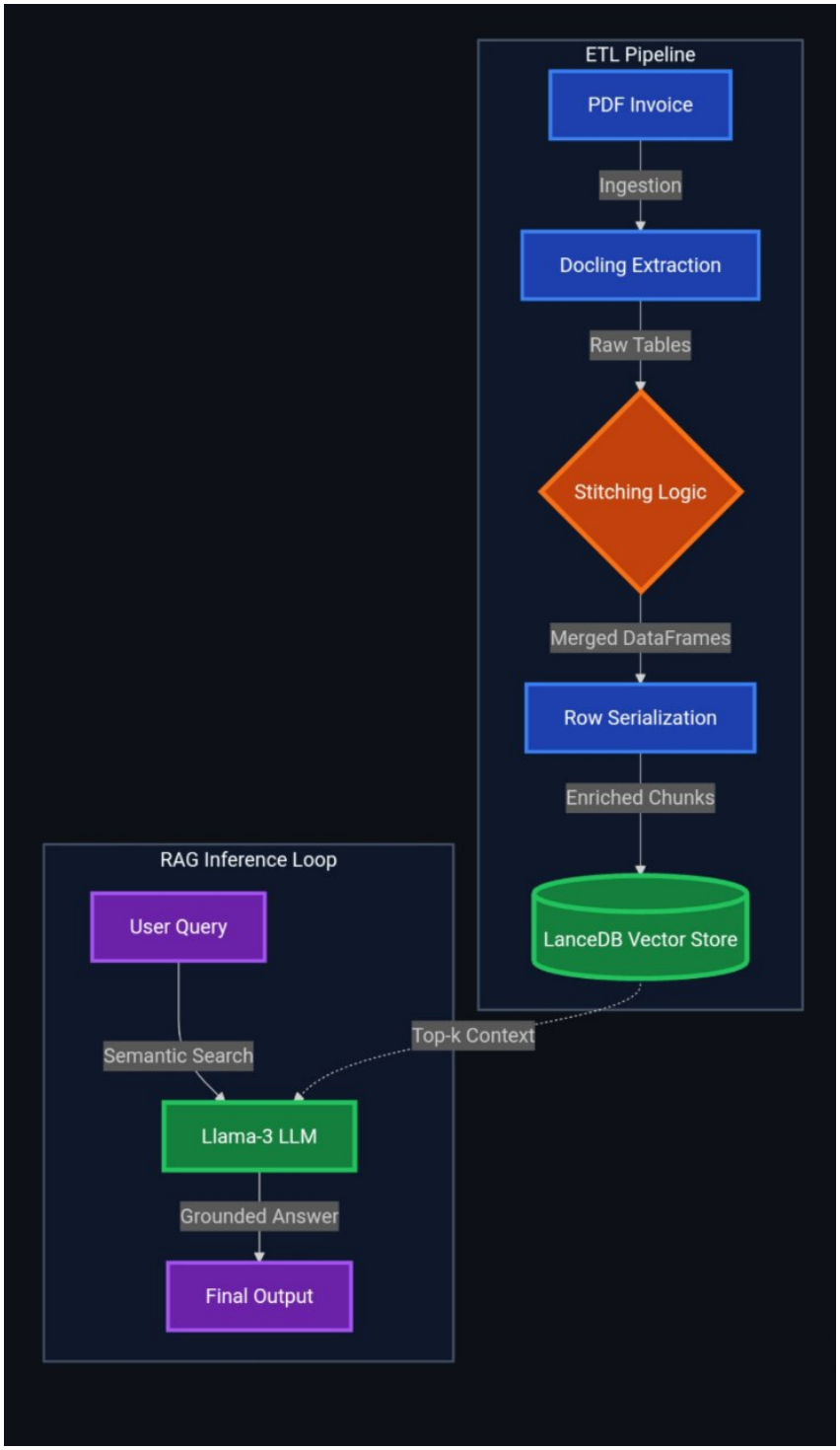

Figure 1: End-to-End System Architecture Pipeline

3 Implementation Details

The core technology stack includes:

- **Docling:** For PDF table extraction.
- **LanceDB:** A vector database for efficient storage and retrieval.

- **Llama-3:** The generative model used for synthesizing answers from retrieved context.

4 Experimental Results

The system was tested against a set of varied invoices (INV-1 through INV-6). Below are the results of the evaluation, ranging from simple lookups to complex aggregations.

4.1 Successful Retrievals

The system performs excellently when retrieving specific attribute values directly from the text.

```
Query: What is the info about INV-1
------------------------------
According to INV-1, there are two line items:
1. DESCRIPTION: Fruit | AMOUNT: 3.00
2. DESCRIPTION: Laptop | AMOUNT: 400.00
The total amount for INV-1 is $403.00.

Query: What is the quantity of NET-5G-4530
------------------------------
According to INV-5, the quantity of NET-5G-4530 is 8 units.
```

Listing 1: Successful Query: Specific Item Lookup

4.2 Category Filtering

The model successfully identifies items belonging to specific categories.

```
Query: What items are related to API Gateway Usage?
------------------------------
According to the provided invoice line items, the following items
    are related to API Gateway Usage:
* Item Code: API-CALL-5999 (Invoice ID: INV-5)
...
* Item Code: API-CALL-3245 (Invoice ID: INV-5)

All of these items have the same description, "API Gateway Usage
   (10k units) - Batch [X]".
```

Listing 2: Successful Query: Category Grouping

5 Discussion of Limitations

While direct retrieval is accurate, the testing phase revealed limitations in the LLM's ability to perform arithmetic and temporal reasoning on retrieved data.

5.1 Arithmetic Hallucinations

When asked to sum values across multiple invoices, the model sometimes retrieves correct individual numbers but hallucinates the aggregation logic or includes irrelevant numbers in the sum.

```
Query: What is the amount of Front and rear brake cables in each
   invoice
------------------------------
* INV-6: $400.00 + $77.50 + $46.50 + ... = $1,349.00
```

Listing 3: Failure Case: Complex Aggregation

Analysis: The model attempted to sum various line items in INV-6 that were likely not "brake cables," resulting in an inflated total. This suggests the retrieval step fetched the entire invoice context, and the LLM failed to filter strictly by the item name before summing.

5.2 Temporal Reasoning Issues

The system struggles when filtering by dates if the date format is ambiguous or if the specific metadata was not serialized into the vector chunk.

```
Query: Get all items from invoices dated March 2024?
------------------------------
Based on the provided invoice line items, there are no invoices
   dated March 2024...
```

Listing 4: Failure Case: Date Filtering

Analysis: Even if relevant invoices existed, the model concluded data was unavailable. This indicates a need for better metadata extraction.

6 Conclusion and Future Work

The "Intelligent Invoice Line Items Extraction Retrieval System" successfully automates the extraction of detailed line items from PDFs. To address the identified limitations, future work will focus on:

1. **Agentic RAG:** Implementing an agent that can use a calculator tool rather than relying on the LLM's internal math capabilities.

2. **Metadata Enrichment:** Extracting dates and invoice IDs as structured metadata fields in LanceDB to allow for precise SQL-like filtering before vector search.