Q.Invoice: AI-Powered Invoice Processing System

Technical Report

Authors: Maria KATIBI and Branly DJIME
Institution: ESILV
Program: Master's in AI/Data Science

January 2026

Abstract

Q.Invoice is an intelligent document processing system that combines Optical Character Recognition (OCR), Large Language Model (LLM) extraction, and adaptive AI querying to transform financial documents into actionable intelligence. The system addresses the challenges of manual invoice processing by automating text extraction, structured data parsing, and natural language querying. Built with Streamlit, PaddleOCR, and OpenAI's GPT-4o, Q.Invoice achieves 97% average OCR confidence and provides context-aware responses to user queries through an adaptive AI engine that changes its personality based on query type.

Keywords: OCR, Document Intelligence, LLM, Natural Language Processing, Invoice Processing, Adaptive AI

Contents

1 Introduction

1.1 Problem Description

Financial document management poses significant challenges for businesses.

Manual Data Entry Issues

- Error rates of 1–3% in manual data entry
- Average processing time: 5–10 minutes per invoice
- High labor costs and low scalability
- Inconsistent data quality

Information Accessibility

- Documents stored as unstructured images or PDFs
- No searchable text or metadata
- Difficult to extract insights across multiple documents
- Limited analytical capabilities

Current Solutions Limitations

- Traditional OCR: low accuracy, no semantic understanding
- Template-based extraction: rigid and configuration-heavy
- Manual tagging: time-consuming and inconsistent

1.2 Objectives

Q.Invoice aims to:

1. Automate document text extraction with accuracy above 95%
2. Extract structured data using LLMs
3. Enable natural language querying across documents
4. Provide intelligent insights and recommendations
5. Deliver a production-ready system with modern UI/UX

1.3 Scope

In Scope

- Invoice, receipt, and quote processing
- PDF and image formats (PNG, JPG, JPEG)
- English language documents
- Single-page and multi-page documents
- Natural language querying
- JSON and Excel export

Out of Scope

- Handwritten documents
- Multi-language support
- Live camera feeds
- Accounting software integration

2 System Architecture

2.1 Overall Architecture

Q.Invoice follows a layered architecture pattern:

```
Presentation Layer (Streamlit)
        |
Application Layer
(Document Processor, Adaptive Query Engine, Export Manager)
        |
Processing Layer
(PDF Conversion, OCR, LLM Extraction)
        |
Data Layer (SQLite + File System)
```

2.2 Component Details

2.2.1 Presentation Layer

Technology: Streamlit 1.32

Features include:

- Natural language query interface

- Document library with preview
- Analytics dashboard
- Upload and chat history sidebar

2.2.2 Application Layer

Document Processor Manages the end-to-end processing pipeline, including validation, batching, and status tracking.

Adaptive Query Engine Detects query intent and selects an appropriate AI personality to generate context-aware responses.

Export Manager Handles JSON and Excel exports with configurable formatting.

2.2.3 Processing Layer

PDF Converter Uses PyMuPDF to convert PDFs into high-resolution images.

OCR Engine PaddleOCR extracts text, confidence scores, and bounding boxes.

LLM Extractor Uses GPT-4o-mini with Pydantic validation and retry logic.

2.2.4 Data Layer

SQLite Database Stores document metadata, processing results, and query history.

File System Manages uploads and exports, organized by timestamp.

3 Processing Pipeline

3.1 Document Processing Flow

```
Input Document
 → Validation
 → PDF Conversion
 → OCR Extraction
 → LLM Processing
 → Storage
 → Searchable Output
```

3.2 Query Processing Flow

```
User Query
 → Query Type Detection
 → Document Retrieval
 → Context Building
 → LLM Call
 → Response Formatting
```

3.3 Technical Details

3.3.1 OCR Configuration

```
ocr = PaddleOCR(
    use_angle_cls=True,
    lang='en',
    use_gpu=False,
    show_log=False
)
```

3.3.2 LLM Extraction Prompt

```
You are a document data extraction expert.

Extract the following from this invoice:
- document_type
- vendor
- invoice_number
- issue_date
- total_amount
- line_items

Respond in JSON format only.
```

3.3.3 Adaptive Query Detection

```
def detect_query_type(question: str) -> str:
    keywords = {
        'calculator': ['total', 'sum', 'calculate'],
        'analyst': ['compare', 'trend', 'analyze'],
        'finder': ['list', 'show', 'find'],
        'advisor': ['optimize', 'save'],
        'auditor': ['check', 'validate'],
        'forecaster': ['forecast', 'predict']
    }

    for qtype, words in keywords.items():
```

```
        if any(w in question.lower() for w in words):
            return qtype

    return 'assistant'
```

4 Evaluation and Results

4.1 Dataset

The evaluation dataset consists of 100 real invoices in PDF and image formats, covering various vendors and layouts.

4.2 OCR Performance

Metric	Value	Notes
Average Confidence	97.3%	All documents
Excellent (>90%)	85	High quality
Good (70–90%)	12	Acceptable
Poor (<70%)	3	Requires rescanning
Processing Time	2.1s	Per page

Table 1: OCR Performance Metrics

4.3 LLM Extraction Accuracy

Field	Accuracy	Notes
Document Type	98%	Very reliable
Vendor Name	95%	Minor variations
Invoice Number	97%	Stable identifiers
Issue Date	93%	Format ambiguity
Total Amount	96%	High reliability
Line Items	89%	Complex tables

Table 2: Structured Data Extraction Accuracy

5 Discussion

5.1 Key Achievements

- OCR accuracy exceeding industry averages
- Flexible LLM-based extraction
- Novel adaptive query engine
- Production-ready implementation

5.2 Limitations

- English-only support
- No handwriting recognition
- Dependency on OpenAI API
- Single-user system

6 Future Work

Future developments include multi-language support, local LLM integration, authentication, advanced analytics, and SaaS deployment.

7 Conclusion

Q.Invoice demonstrates the effective integration of OCR, LLMs, and adaptive AI querying to automate invoice processing. The system achieves high accuracy, strong usability, and introduces a novel adaptive query mechanism that enhances user experience and analytical capabilities.

References

1. PaddleOCR: https://github.com/PaddlePaddle/PaddleOCR
2. OpenAI GPT-4o: https://openai.com/research/gpt-4
3. Streamlit: https://streamlit.io
4. Pydantic: https://pydantic-docs.helpmanual.io
5. SQLAlchemy: https://www.sqlalchemy.org