Hackathon: Intelligent Document AI for Field Extraction from Invoices

Description

In modern financial institutions, automated extraction of key details from invoices, quotations, and other semi-structured business documents is crucial for accelerating credit decisioning, vendor reconciliation, and loan disbursal workflows. Manual entry slows down operations and introduces errors, hence, intelligent, cost-efficient, and language-agnostic extraction solutions are a fundamental layer in banking automation.

In this hackathon, your goal is to **build a Document AI system that extracts key fields from invoice-type documents** (for example in this case, *tractor loan quotations*). These invoices vary in **structure, layouts, language** (English and vernaculars like Hindi/Gujarati) **and quality** (scanned, handwritten, or photographs of invoice). The model should robustly handle layout diversity and output structured data with high accuracy and confidence.

While this challenge uses **tractor loan quotations** as a running use case, you should design your solution to **generalize to any type of invoice**: retail invoices from shops, industrial invoices etc.

Goal

The primary goal:

> Given an input document (in PDF format), extract the following fields and return them as a structured JSON output:
>
> - Dealer Name (Datatype: **Text**, Evaluation: fuzzy match)
> - Model Name (Datatype: **Text**, Evaluation: exact match)
> - Horse Power (Datatype: **Numeric**, Evaluation: exact match)
> - Asset Cost (Datatype: **Numeric**, Evaluation: exact match)
> - Presence of Dealer Signature (Datatype: **Binary along with bounding box co-ordinates**, Evaluation: IOU >0.5 with GT Bounding box)
> - Presence of Dealer Stamp (Datatype: **Binary along with bounding box co-ordinates**, Evaluation: IOU >0.5 with GT Bounding box)

You are expected to:

- Build an **end-to-end pipeline** that reads PDF invoices, processes them via OCR and vision layers, and extracts structured fields with confidence scores.
- **Design your own architecture and code** supporting explainability and reproducibility.
- **Estimate cost and latency per document**, proposing cost-accuracy tradeoffs for scale.

- **Handle lack of ground truth:** create pseudo-labels, annotation workflows, or self-checking techniques.
- **Demonstrate performance (≥95% document-level accuracy)** on the shared dataset of sample quotations.

Bonus points are awarded for:

- Creative **EDA and visualization** on the dataset
- **Error analysis and model diagnostics**
- Clear **architecture diagrams and modular code structure**
- **Low-cost inference design (**using open-source components like PaddleOCR or YOLO, or SLMs)

Data

Dataset Overview

- Total documents: ~**500** images ***(PII data like customer name and phone no is redacted)***
- Composition: digital, scanned, and handwritten quotations from multiple states
- Languages: English, Hindi, Gujarati, and other vernacular mixes

Fields to Extract

Field	Description	Matching Type
Dealer Name	Extract dealer information and perform fuzzy matching (≥90%) against master file	Fuzzy
Model Name	Tractor or asset model extracted and matched exactly to asset master	Exact
Horse Power	Numeric value (e.g., “50 HP” → 50)	Exact
Asset Cost	Total cost (digits only, no words or currency symbols)	Exact
Dealer Signature	Check presence and extract bounding box	Exact
Dealer Stamp	Check presence and extract bounding box	Exact

Your system should output one JSON object per document containing all fields and confidence scores.

Architecture Guidelines

Teams are encouraged to explore multiple strategies that blend **Computer Vision** and **Natural Language Processing (NLP)** concepts or to experiment with integrated **Visual Language Models (VLMs)** like Qwen2.5-VL (2B/7B) capable of understanding both text and layout in documents.

A well-designed system typically includes the following high-level stages:

1. **Document Ingestion and Interpretation**
   Convert input documents into an analyzable representation: either as images or structured text layouts. Ensure the system handles diverse formats, including scanned, digital, and multilingual invoices.
2. **Visual and Textual Understanding**
   Extract information from both the visual and textual dimensions of the document. This may involve techniques to read embedded text, interpret tabular structures, distinguish regions, or understand contextual positioning of key-value pairs.
3. **Field Detection and Entity Recognition**
   Identify targeted elements such as names, numeric values, or visual markers (like signatures and stamps). Participants may adopt rule-based, statistical, or learning-based methods depending on their design philosophy and resource constraints.
4. **Semantic Reasoning and Structuring**
   Link and standardize extracted content into consistent fields (e.g., dealer names, model identifiers, numeric attributes). Apply logical consistency checks or schema-based validations to ensure coherent outputs.
5. **Post-Processing and Quality Assurance**
   Refine results by reconciling near-duplicates, normalizing textual variations, confirming numeric accuracy, and applying threshold-based confidence scoring.
6. **Output Generation and Evaluation**
   Present final results in a structured JSON format containing both extracted values and associated confidence measures. Measure performance along dimensions of accuracy, cost, and latency.

Each component of the system should be **evaluated independently** to understand its trade-offs in precision, scalability, and processing speed. Innovative combinations, ensemble logic, or light-weight multimodal models are highly encouraged for achieving optimal document-level accuracy and cost efficiency.

Handling Lack of Ground Truth

In this challenge, no pre-labelled data is provided. This setting reflects real-world scenarios where annotated datasets are scarce or expensive to obtain, and participants must design strategies to approximate ground truth through ingenuity and experimentation.

Teams are encouraged to simulate labelled supervision using one or more of the following approaches:

- **Manual Sampling and Annotation**
  Select a small, representative subset of documents and annotate them carefully for key fields. This curated sample can serve as a seed dataset for validation or fine-tuning lightweight models.

Research in active learning (e.g., Settles, 2009) suggests that high-quality small samples can substantially improve overall accuracy.

- **Pseudo-Labeling and Bootstrapping**
  Generate initial labels automatically by combining deterministic rules with confident model outputs. Iteratively refine these pseudo-labels as the model improves.

- **Consensus and Self-Consistency Methods**
  Compare outputs from multiple independent models or extraction pipelines, and treat the majority or most consistent prediction as the provisional label. This ensemble-based voting aligns with techniques used in co-training and multi-view learning frameworks.

- **Weak Supervision and Synthetic Data Generation**
  Create artificial or template-based invoices with known field values to pre-train or calibrate models. Weakly labeled or simulated sources can help the system learn canonical structures and regularities in real documents. Concepts from Snorkel (Ratner et al., 2017) and related frameworks may inspire such designs.

Evaluation

Primary Metric

Document-Level Accuracy (DLA):
Percentage of documents for which all six fields are accurately extracted.

Field Type	Matching Rule
Dealer / Model	≥90% fuzzy or exact textual match
HP / Cost	Numeric equality within ±5% tolerance
Signature / Stamp	Presence correct & IoU ≥ 0.5 on bounding boxes

Target: **≥95% DLA**

Secondary Metrics

- Field-level mAP 50-95 for signature and stamp
- Average latency per document (≤30 seconds)
- Cost per document (<$0.01 on CPU or low-tier GPU)

Submission Format

Teams must submit a ZIP file containing:

```
submission.zip
|
├── executable.py        # Main file to run extraction
├── requirements.txt
├── README.md            # Architecture, pipeline, cost analysis, diagrams
├── utils/               # Supporting modules (optional)
└── sample_output/
    └── result.json
```

Output JSON Format

```
{
  "doc_id": "invoice_001",
  "fields": {
    "dealer_name": "ABC Tractors Pvt Ltd",
    "model_name": "Mahindra 575 DI",
    "horse_power": 50,
    "asset_cost": 525000,
    "signature": {"present": true, "bbox": [100, 200, 300, 250]},
    "stamp": {"present": true, "bbox": [400, 500, 500, 550]}
  },
  "confidence": 0.96,
  "processing_time_sec": 3.8,
  "cost_estimate_usd": 0.002
}
```

Evaluation Process

- Evaluation dataset: ~100 unseen invoices (unknown layout and language combinations).
- Organizer team runs your executable on cloud environment.
- Scores computed on **DLA**, **latency**, and **inference cost**.
- Top 5 teams proceed to **live demonstration round**: process 10 random PDFs and present EDA, model design, and inference results.

Bonus Points

- Rich **EDA and Visualization** (some examples below)**:**
  - State-wise and language-wise document distribution
  - Correlation between language type and error rate
  - Processing time analysis

- **Error Analysis:**
    - Highlight failure cases and create error categories and their distribution
- **Architecture Diagram & Explainability:**
    - Clear, modular visual of your system pipeline.
- **Deployability:**
    - Optional Streamlit / Gradio web app for demo (not mandatory, will not be considered for final evaluation)