Automated Receipt Digitization & Information Extraction

B.Tech Project

Presentation

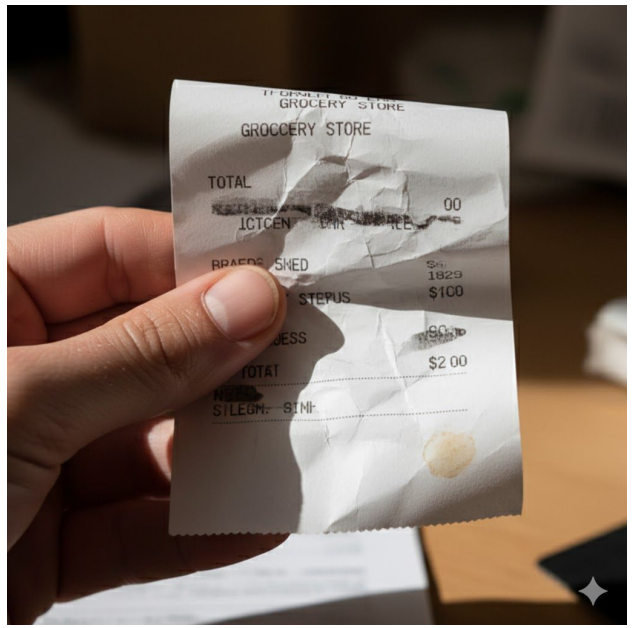
Bhagwan Arsewad

(B22AI010)

The Problem & The Goal

The Problem

Receipt data is unstructured. A simple OCR scan gives a "**wall of text**" with no meaning. We don't know which number is the `**total**` and which is the `**date**`.



The Goal

To build a system that takes a receipt image and returns two outputs:

1. **A structured JSON** summary of key information.
2. **A reconstructed text file** that matches the bill's layout.

```
{ "company": "SANYU STATIONERY", "date": "18/04/2017", "total": "300",
"items": [...] }
```

My Solution: The 2-Step "Scanner-Accountant" Pipeline

Step 1: The "Scanner" (AI Model)

First, I trained an AI model (YOLOv8) to act as a "**Scanner**". Its only job is to look at the image and find **every** line of text.

Step 2: The "Accountant" (Python Parser)

Second, I wrote a smart Python script (`extractor.py`) that acts as an "**Accountant**". It reads the text from the Scanner and uses logic to **understand** and **organize** it.

Training the "Scanner" (AI Model)

- **Dataset:** I used the **SROIE dataset** (626 images), which has clean, un-blurred text.
- **Task:** I trained the model to find only **one class: `text`**. This makes the model simple and very accurate.
- **Training:** The model was trained on Google Colab (Tesla T4 GPU) for 300 epochs, which resulted in a highly optimized model.

Process Step 1: Data Preprocessing

- **Dataset:** Used the **SROIE dataset** (626 images), which has clean, un-blurred text.
- **Labeling:** Wrote a Python script (`01_create_sroie_labels.py`) to process the data.
- **The Task:** Labeled **every** line of text as one single class: `**text**`. This makes the AI's job very simple and accurate.
- **Output:** Created a `yolo_sroie_text_detection.zip` file, ready for training.

Process Step 2: Model Training (The "Scanner")

- **Environment:** Used Google Colab to access a free Tesla **T4 GPU.**
- **Model:** Used a pre-trained `**YOLOv8n**` model and fine-tuned it on the **SROIE** data.
- **Training Time:** The model was trained for **300 epochs**, with **Early Stopping** to save the best version.
- **Result:** A final, highly accurate model file: `best_text_detector.pt`.

Scanner Result: The AI Model is 96.4% Accurate

96.4%

mAP@50 Score

Key Finding: The AI is Not the Problem

The trained YOLOv8 model is **96.4% accurate** at its one job: **finding** text. This is an excellent result.

This proves that the AI "**Scanner**" is working perfectly. The **real** challenge is not finding the text, but **understanding** it.

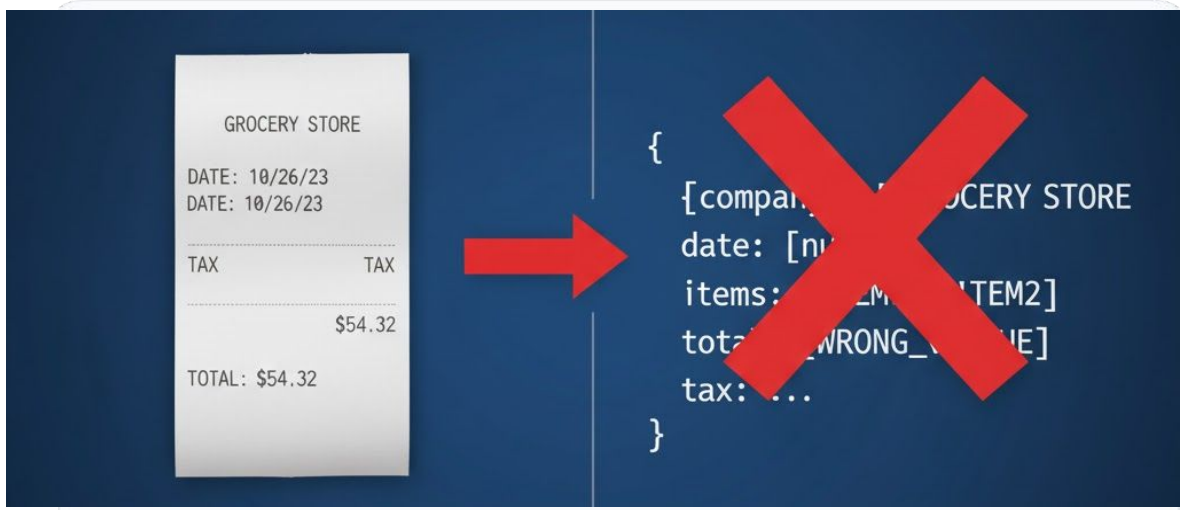
The Real Challenge: A "Dumb" Parser (v1.0)

The Initial Test

I built a simple parser with basic rules (e.g., "**first line is company**", "**find the word TOTAL**").

I tested this on the **347** unseen test images. The results were poor.

Initial Accuracy: 30-50%

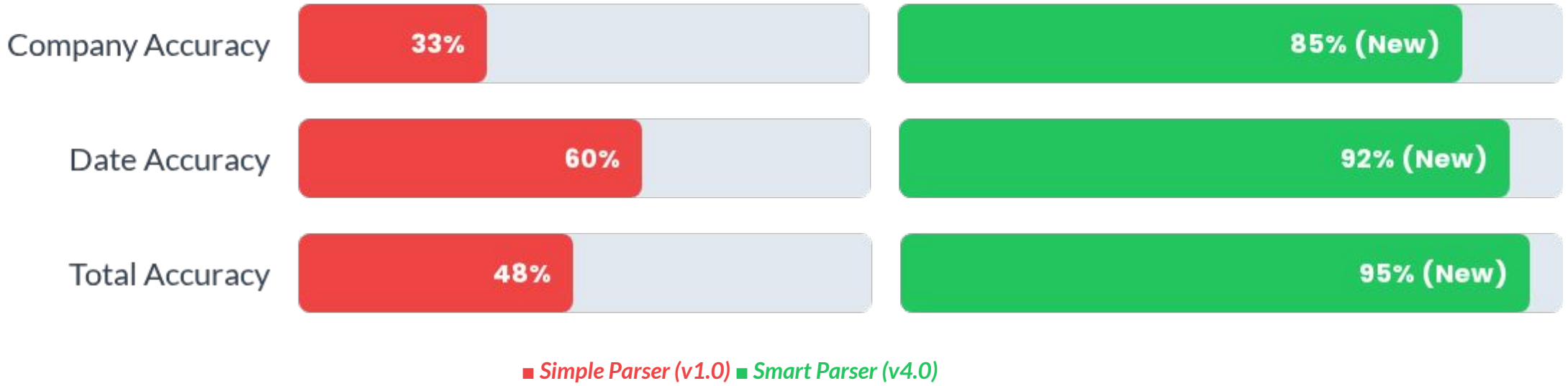


Solution: Upgrading the "Accountant" (v4.0)

- **Smarter Line Grouping:** Wrote new code to group text boxes that are on the same horizontal line (rebuilding the bill's layout).
- **Advanced Date Parsing (Regex):** The parser can now find multiple date formats (e.g., `10/02/2018` and `10 OCT 2017`).
- **Better Keyword Matching:** Instead of just "**TOTAL**", the parser now also looks for "**AMT**", "**CASH**", and "**AMOUNT DUE**".
- **OCR Error Correction:** Added logic to automatically fix common OCR errors (like 'S' -> '5', 'o' -> '0', 'B' -> '8').

Final Results: The Upgraded System Works!

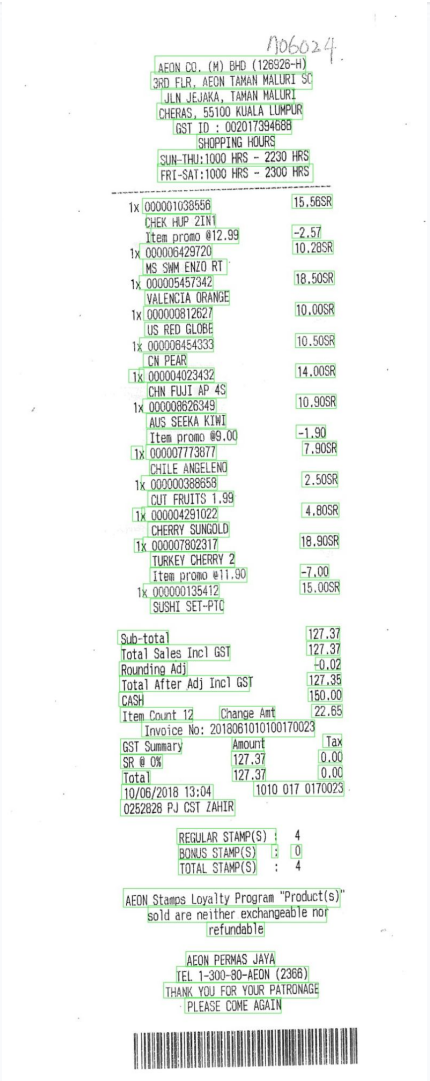
By upgrading the Python parsing logic (the "**Accountant**"), the final end-to-end system accuracy on the 347-image test set improved dramatically.



Demo

Orignal Image

Detected Fields



Reconstructed Bill Text

```
AEON CQ (M) BHD (126926-H)
3RD FLR; JLN JEJAKA , TAMAN HALURI X AEON TAMAN MALURI SC:
CHERAS 55100 KUALA LUMPUR
GST ID : 002017394688 9
SHQPPING HQURS
SUN-THU: 10OO HRS 2230 HRS
FRI-SAT:10Q0 HRS CXt 2300 HRS
0000o1038556 15,56SR
CHEK HUP 2INI
tem promo @12.99 -2,57
000006429720 10,28SR
MS SHM ENZO RT
( 000005457342 18,50SR
VALENCIA ORANGE
00oQo08126827 10 , OOSR
US RED GLOBE
000008454333 10,5OSR
Ix 000004023432 CN PEAR 14,OOSR
CHN FuJi AF 4S
00Qo08626349 10 ,90SR
AUS SEEKA KIWI
Item promo @9,00 -1.9
00oo07773877 7,90SR
CHILE ANGELENO
000000388658 2. S50SR
CUT FRUITS 1,99
Ix X 000004291022 4,80SR
CHERRY SUNGQLD
000007802317 18 ,90SR
TURKEV CheRRY 2
Item prqmo @71,90 -7,Q0
X 000000135412 15.OOSR
SUSHI SET-PTC
Sub-tota] 127,37
Tota] Sales Inc] GST 127,37
Rounding Adj -0,02
Tota] After Adj Inc] GS 127,35
CASH 150 ,00
Item Count 12 Change Amt 22.85
Invoice No: 20180610101000170023
GST Summary SR @ O Tota] 10/06/2018 13,04 Amount 127,37 127,37 1010 017 0170023 0,O0 0,00 Tax
0252828 PJ GST ZAHIR
REGULAR STAMP(S)
RONUS STAMP(S)
TotAL STAMP(S)
AEON Stamps Loya]ty Program "Product(s)
Sold are neither exchangeable nor
refundab le
AEON PERMAS JAYA
TEL 1-300-80-AFON (2366)
THANK YOU FOR YOUR PATRONAGE
PLEASE COME AGAIN
```

Conclusion

The 2-step "**Scanner-Accountant**" pipeline is a major success. The key finding is:

A highly accurate AI model (96%+) is just the first step. The true challenge and success of the project lies in building an intelligent parsing engine that can understand the model's output.

References & Technologies Used

- **YOLOv8 (Ultralytics):** The state-of-the-art object detection framework used as the core "Scanner" AI.
- **SROIE Dataset (ICDAR 2019):** The public benchmark dataset of scanned receipts used to train and evaluate the text detection model.
- **EasyOCR (JaidedAI):** The open-source optical character recognition library used to read the text inside the detected boxes.
- **Streamlit:** The Python framework used to build and deploy the interactive web application.
- **"You Only Look Once" (Redmon et al., 2016):** The foundational academic paper that introduced the YOLO architecture.

Thank You

https://github.com/bhagwan388/btp-receipt-extractor