Assignment: PDF Data Extraction and Rapid Prototyping

Page 1: Flask Web App for Nanonets OCR

Overview

This documentation outlines a simple Flask web application designed to facilitate Optical Character Recognition (OCR) using the Nanonets API. The application allows users to upload both PDF and image files, extracts text content via Nanonets OCR, and generates a CSV file for easy data analysis.

Key Features

1. File Uploads:The application supports the upload of both PDF and image files through a user-friendly form.

2. Nanonets OCR Integration:Utilizes the Nanonets OCR API for efficient and accurate text extraction from uploaded files.

3. CSV File Generation: After OCR processing, the application generates a CSV file containing the extracted text for further analysis.

4. User Interface: The minimalistic web interface provides a straightforward means for users to interact with the OCR functionality.

How to Use

1.Run the Flask App:

-Ensure that Flask is installed (pip install Flask).

-Execute the Flask application (python script.py).

-Open a web browser and navigate to http://127.0.0.1:5000/.

2.Upload Files:

-Use the file input in the form to upload a PDF

3. OCR Processing:

- The application uses the Nanonets OCR API to extract text content from the uploaded file.

4. CSV File:

- A CSV file is generated with the extracted text and opened in the default system application for user convenience.

Page 2: Challenges Faced during Development

Challenges and Solutions

1. Data Extraction Limitations with Python Libraries

Issue: Faced challenges with existing Python OCR libraries (e.g., OCRmypdf, PyPDF) as they struggled to provide desirable output on various projects.

Solution: Transitioned to Nanonets OCR API due to its higher accuracy and versatility in handling diverse document types.

2. Integration with Flask

Issue: Encountered difficulties while integrating the OCR script with Flask, leading to compatibility issues with OCRmypdf, Ghostscript, and Poppler utilities.

Solution: Resolved integration challenges by streamlining the Flask app's architecture and opting for a cloud-based solution with Nanonets OCR.

3. Scanned PDFs Reading Issues

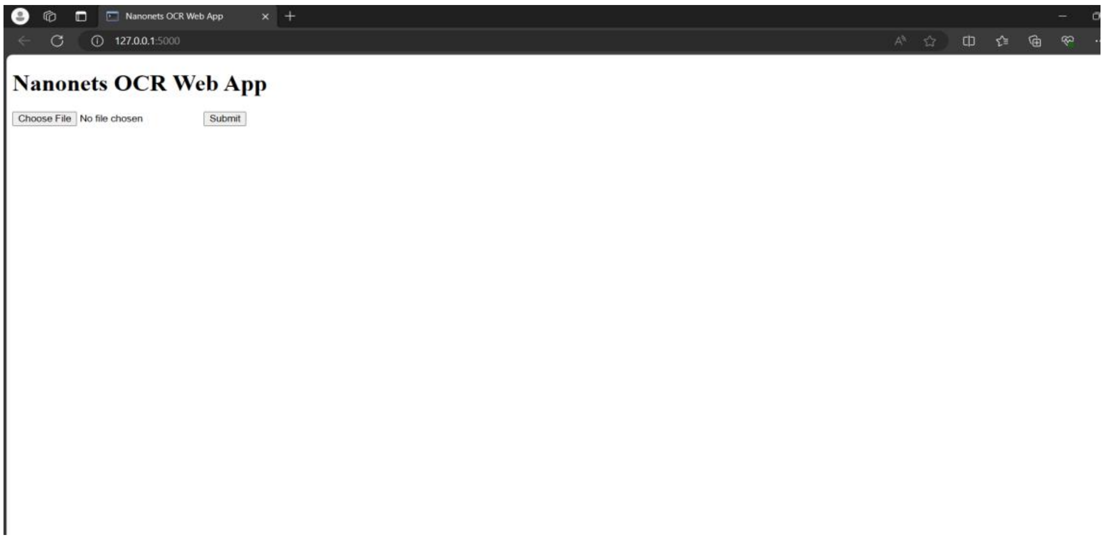
Issue: Traditional Python libraries (e.g., PyPDF) faced difficulties in reading scanned PDFs, hindering text extraction.

Solution: Nanonets OCR, designed for modern document challenges, provided efficient text extraction from scanned documents, addressing this specific issue.

Conclusion

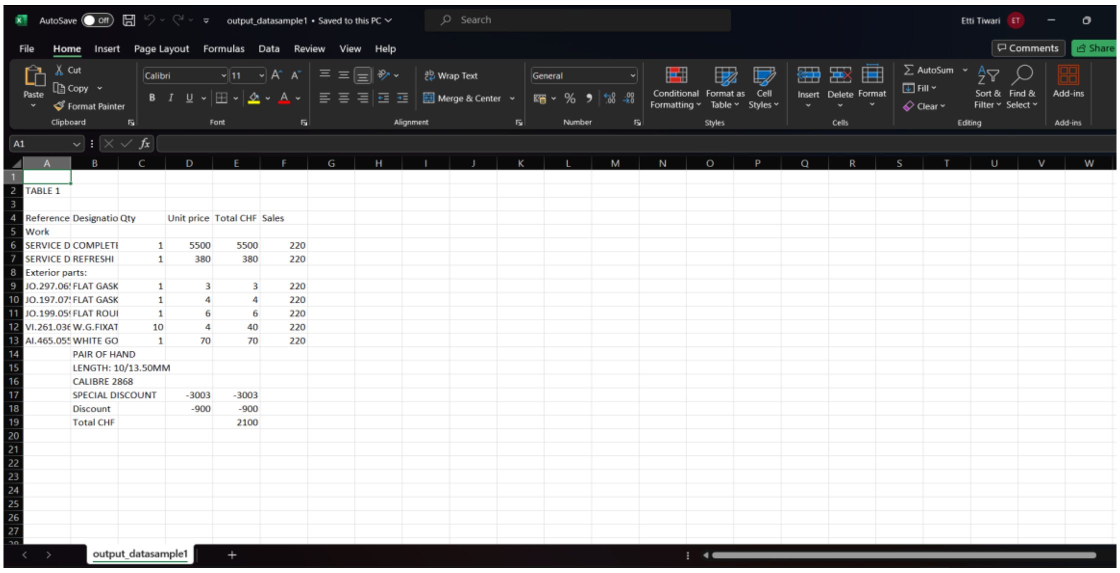
The development of this Flask web application was not without challenges. Overcoming issues related to the limitations of Python OCR libraries and achieving seamless integration with Flask led to the adoption of the Nanonets OCR API. This decision was rooted in the API's ability to address challenges associated with data extraction, integration complexities, and reading scanned PDFs. The result is a robust OCR solution that efficiently handles a variety of document types, providing accurate and versatile text extraction capabilities.

Output:



Nanonets OCR Web App

Choose File No file chosen Submit



	A	B	C	D	E	F
1						
2	TABLE 1					
3						
4	Reference	Designatio	Qty	Unit price	Total CHF	Sales
5	Work					
6	SERVICE D	COMPLETE	1	5500	5500	220
7	SERVICE D	REFRESHI	1	380	380	220
8	Exterior parts:					
9	JO.297.06	FLAT GASK	1	3	3	220
10	JO.197.07	FLAT GASK	1	4	4	220
11	JO.199.05	FLAT ROUI	1	6	6	220
12	VI.261.036	W.G.FIXAT	10	4	40	220
13	AI.465.055	WHITE GO	1	70	70	220
14		PAIR OF HAND				
15		LENGTH: 10/13.50MM				
16		CALIBRE 2868				
17		SPECIAL DISCOUNT		-3003	-3003	
18		Discount		-900	-900	
19		Total CHF			2100	

----Etty Tiwari