**Digital Text Extraction and OCR**

***Comprehensive PDF Information Extractor and CSV Generator***

**1. Introduction**

***Overview of the Project****:*

This project focuses on optimizing and enhancing an existing OCR software system, which is used to convert scanned documents into text and extract specific information such as issue dates, invoice numbers, and supplier details. The enhancement includes skipping unnecessary OCR processes for already digital documents, which improves the system's efficiency and accuracy.

***Purpose and Objectives****:*

The main objective was to improve the existing OCR system by introducing a function to handle digital documents without performing OCR, thereby avoiding potential errors in text extraction from already digitalized documents.

**2. Existing System Overview**

**Description of the Old Software:**

The previous system was capable of performing OCR on any document, extracting necessary details, and storing them in a journal for tracking purposes. It included a main function that managed the OCR and information extraction processes.

**Identified Issues and Limitations**:

A significant problem was that the system performed OCR on all documents, including those that were already digital (e.g., online invoices), which was unnecessary and could lead to inaccuracies when the text was extracted from images generated during the OCR process.

**3. New Enhancements**

**SkipOCR Functionality**:

The newly introduced function, SkipOCR, does not determine if a document is digital. Instead, a separate function, Is-OCRed, is responsible for checking whether a PDF document is already OCR-ed or not. If the document is found to be digital, the SkipOCR function is invoked to extract text directly using the pdftotext tool, ensuring that the extracted information is accurate and free of OCR-induced errors.

**Improvements in Efficiency and Accuracy**:

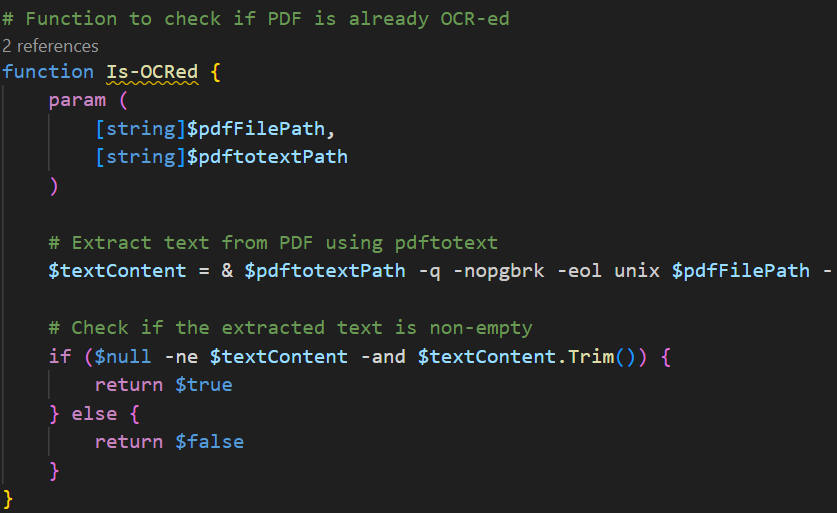
By checking if a document is already OCR-ed and skipping the OCR process when it is not needed, the new method significantly reduces processing time and eliminates errors introduced during the OCR of already digital documents.

**4. Technical Implementation**

***Main Function Workflow****:*

**Is-OCRed Function**:

This function checks if the PDF document is already OCR-ed by using the pdftotext tool to extract text from the document. If the extracted text is non-empty, the document is deemed already OCR-ed. The parameters are $pdfFilePath which is the path of the current document in the process, and $pdftotextPath is the path of the library I used for extraction of the text.



**SkipOCR Function**:

If the document is found to be already OCR-ed, the system bypasses the OCR process and directly extracts text using pdftotext. The extracted text is then passed to another function that processes it to extract relevant information such as invoice numbers, issue dates, and supplier details.

**Process-AllPDFs Function**:

This function iterates over all PDF files in the specified directory. For each file, it first creates a backup and then checks if the file is already OCR-ed. Depending on the result, it either extracts the information directly or performs OCR and after that extracting the information.

A screen shot of a computer code

Description automatically generated

**PDF Text Extraction Process**:

Text extraction for digital documents is handled by pdftotext, which converts the entire PDF content into plain text. This approach preserves the format and ensures that no data is lost or corrupted.

**CSV Files: Suppliers and Journal**:

***Two CSV files are integral to the system:***

1. **Suppliers CSV**:

Contains supplier information, including multiple keywords to identify relevant details in the text.

SupplierName;FriendlyName;InvoiceCode;DateKey;DateKeyDirection;InvKey;InvKeyDir;InvNumLength;**DateFormat;**SupplierCUI;SupplierREG

1. **Journal CSV**:

Logs the extracted information, including the supplier name, invoice number, issue date, and other key data.

2024-08-19";"D:\Invoices\\_ocr\2024-07-16\_ORANGE\_COMMUNICATIONS\_RO9010105\_TKR-240305324193\_OCR.pdf";"ORANGE\_COMMUNICATIONS";"RO9010105";"J40/8926/1997";"2024-07-16";"TKR-";"240305324193"

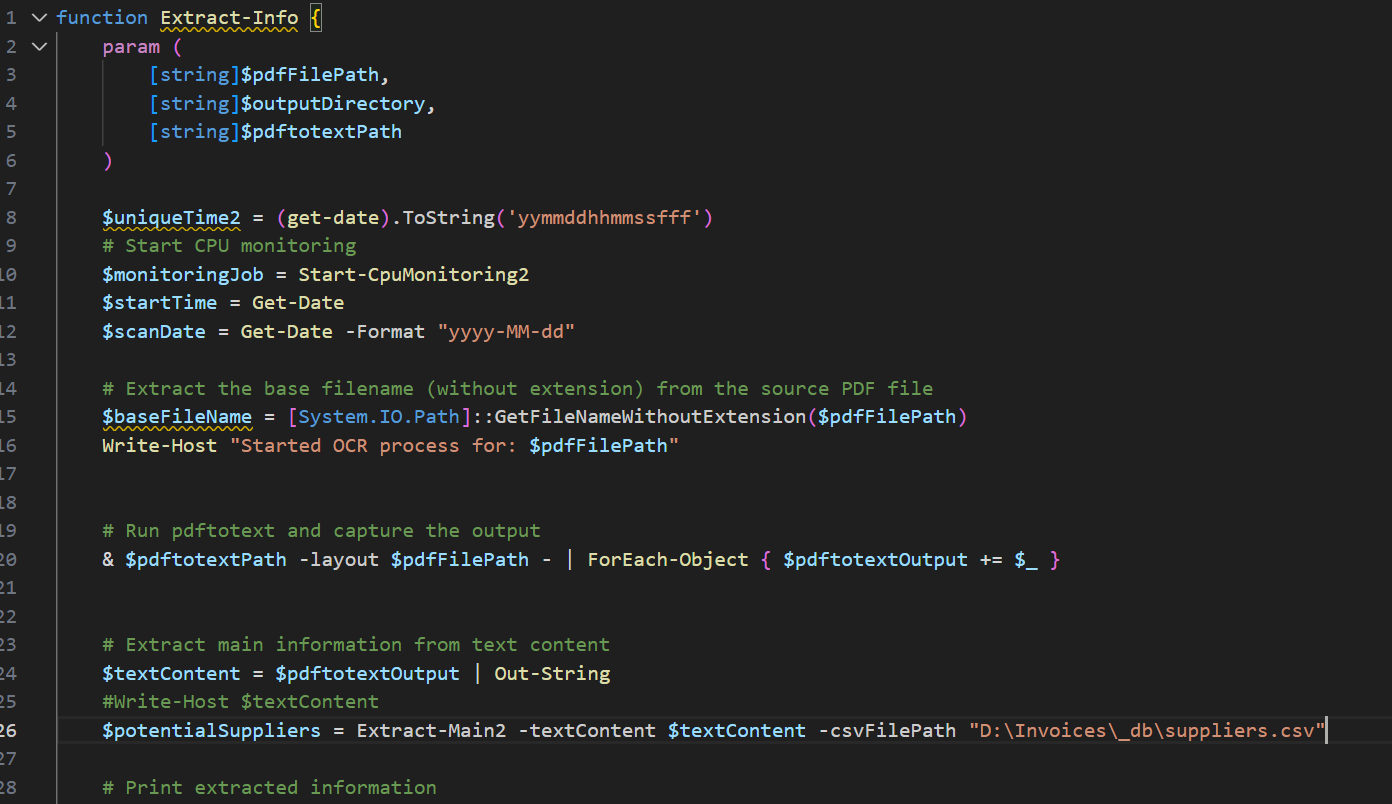
***After processing the documents, they are renamed with the name of the information that has been extracted , for easily searching/finding of a specific document.***

***Keyword Search and Information Extraction:***

The system searches for predefined keywords in the extracted text to locate and extract relevant information. This process includes handling various date formats and supplier-specific codes.

***skipOCR function Workflow:* Functions Details:**

1. ***Extract-Info:***



 **Purpose:**

* Orchestrates the entire extraction process for invoice information from a PDF file.
* Renames the PDF based on the extracted data and logs the information into a CSV file.

 **Parameters:**

* $pdfFilePath (string): The full path of the PDF file to process.
* $outputDirectory (string): The directory where processed PDF files will be saved.
* $pdftotextPath (string): Path to the pdftotext executable for extracting text from the PDF.

 **Workflow:**

1. **CPU Monitoring:**
   * Starts CPU monitoring to measure resource usage during the process.
2. **Text Extraction:**
   * Utilizes pdftotext to extract text from the PDF.
3. **Data Extraction:**
   * Extracts relevant information such as supplier name, invoice number, and issue date using the Extract-Main2 function.
4. **File Renaming:**
   * Renames the PDF file based on the extracted invoice data.
5. **Journal Entry:**
   * Logs the extracted information into a CSV journal.
6. **CPU Monitoring Summary:**
   * Outputs the average CPU usage during the process.

 **Outputs:**

* Renamed PDF file.
* Updated CSV journal with extracted invoice information.

1. ***Extract-Main2***

***A screenshot of a computer program

Description automatically generated***

* **Purpose:**
  + Identifies the supplier in the text content and extracts invoice-related data.
* **Parameters:**
  + $textContent (string): The extracted text content from the PDF.
  + $csvFilePath (string): Path to the CSV file containing supplier information and keys.
* **Workflow:**
  + **CSV Loading:**
    - Loads supplier information from the specified CSV file.
  + **Supplier Matching:**
    - Matches suppliers based on a pre-defined pattern in the text content.
  + **Data Extraction:**
    - Calls Extract-SupplierInfo2 to extract specific details like invoice number and issue date.
  + **Fallback:**
    - Returns placeholder data if no match is found.
* **Outputs:**
  + An object containing the matched supplier's information, or placeholder data if no match is found.

1. ***Extract-SupplierInfo2***

***A screen shot of a computer

Description automatically generated***

* **Purpose:**
  + Extracts specific supplier-related information such as invoice numbers and issue dates.
* **Parameters:**
  + $textContent (string): The full text content from the PDF.
  + $csvFilePath (string): Path to the CSV file with supplier details.
  + $identifiedSupplier (string): The name of the supplier identified in the text.
  + $words (string[]): The words extracted and processed from the text content.
* **Workflow:**
  + **CSV Row Matching:**
    - Identifies the correct supplier row in the CSV based on the supplier name.
  + **Invoice and Date Extraction:**
    - Uses Find-InvoiceNumber2 and Find-NearestDate2 to extract the invoice number and issue date.
  + **Return Data:**
    - Creates and returns an object containing the extracted supplier information.
* **Outputs:**
  + A PowerShell object containing extracted supplier details.

1. ***Find-NearestDate2***

***A screen shot of a computer code

Description automatically generated***

* **Purpose:**
  + Finds and extracts the date nearest to a specified keyword in the text.
* **Parameters:**
  + $dateKey (string): The keyword to locate near which the date will be extracted.
  + $dateKeyDirection (string): Direction (Right or Left) to search relative to the dateKey.
  + $dateFormat (string): Expected date format (e.g., yyyy-MM-dd).
  + $words (string[]): Array of words extracted from the text.
* **Workflow:**
  + **Flatten Words:**
    - Joins the words into a single string for processing.
  + **Pattern Matching:**
    - Converts the expected date format into a regex pattern using ConvertDateFormatToRegex2.
  + **Date Extraction:**
    - Identifies and validates the date based on the keyword's position and the regex pattern.
* **Outputs:**
  + The nearest date found in the specified format or null if no valid date is found.

1. ***ConvertDateFormatToRegex2***

***A screen shot of a computer program

Description automatically generated***

* **Purpose:**
  + Converts a given date format into a regex pattern for date extraction.
* **Parameters:**
  + $dateFormat (string): The expected date format (e.g., yyyy-MM-dd).
* **Workflow:**
  + **Replace Placeholders:**
    - Replaces date format placeholders with equivalent regex patterns.
  + **Handle Special Characters:**
    - Escapes special characters used in the date format.
* **Outputs:**
  + A regex pattern string that matches the expected date format.

1. ***GenerateDateFormatVariants2***

***A screen shot of a computer program

Description automatically generated***

* **Purpose:**
  + Generates variations of a given date format to account for differences in how dates might be presented.
* **Parameters:**
  + $dateFormat (string): The original date format.
* **Workflow:**
  + **Generate Variants:**
    - Creates variations by substituting parts of the format (e.g., dd to d, MM to M).
* **Outputs:**
  + An array of possible date format variants.

1. ***Find-InvoiceNumber2***

***A screen shot of a computer program

Description automatically generated***

* **Purpose:**
  + Extracts the invoice number from the text using a key pattern.
* **Parameters:**
  + $InvKey (string): The key string near which the invoice number is expected.
  + $InvKeyDir (string): Direction (Right or Left) to search for the invoice number relative to the InvKey.
  + $InvNumLength (int): Expected length of the invoice number.
  + $words (string[]): Array of words extracted from the text.
* **Workflow:**
  + **Flatten Words:**
    - Joins the words into a single string for easier processing.
  + **Pattern Matching:**
    - Uses a regex pattern based on the expected length of the invoice number to find potential matches.
  + **Extract Invoice Number:**
    - Identifies the invoice number based on its proximity to the key string.
* **Outputs:**
  + The found invoice number or null if no match is found.

1. ***Start-CpuMonitoring2 & Stop-CpuMonitoring***

***A screen shot of a computer program

Description automatically generated***

* **Purpose:**
  + Monitors the CPU usage during the extraction process.
* **Parameters for Start-CpuMonitoring2:**
  + None directly; it uses the current process ID.
* **Workflow:**
  + **CPU Monitoring:**
    - Continuously records CPU usage until stopped.
* **Outputs:**
  + A job object used for monitoring, which is later stopped and processed by Stop-CpuMonitoring.

A computer screen shot of code

Description automatically generated

* **Parameters for Stop-CpuMonitoring:**
  + $monitoringJob (object): The job object returned by Start-CpuMonitoring2.
* **Workflow:**
  + **Job Completion:**
    - Stops the CPU monitoring job.
  + **CPU Data Calculation:**
    - Computes the average CPU usage during the monitored period.
* **Outputs:**
  + Displays the average CPU usage in percentage.

**How the script works:**

1) Put all the pdf documents that you want in the “Invoices” folder. The script will automatically process every pdf from this folder.

A screenshot of a computer

Description automatically generated

2) Open powershell or cmd and run the script:

A black background with white text

Description automatically generated

3) The output is the extracted text and the extracted information about the invoice:

A screenshot of a computer

Description automatically generated

A computer screen with numbers and letters

Description automatically generated

A screenshot of a computer

Description automatically generated

**The following data was inserted in the invoices journal :**

"2024-08-20";"D:\Invoices\\_ocr\2024-05-11\_ORANGE\_COMMUNICATIONS\_RO9010105\_TKR-240303647267\_OCR.pdf";"ORANGE\_COMMUNICATIONS";"RO9010105";"J40/8926/1997";"2024-05-11";"TKR-";"240303647267"

"2024-08-20";"D:\Invoices\\_ocr\2024-04-17\_STEINER\_RO38171300\_STE-0020\_OCR.pdf";"STEINER";"RO38171300";"J26/1652/2017";"2024-04-17";"STE-";"0020"

"2024-08-20";"D:\Invoices\\_ocr\2024-07-31\_ETA2U\_RO1801821\_ETA-1202410240\_OCR.pdf";"ETA2U";"RO1801821";"J35/703/1992";"2024-07-31";"ETA-";"1202410240"

**Then the pdfs were moved to the processed pdfs folder:**

A close up of a number

Description automatically generated

**After processing the documents, they are renamed with the name of the information that has been extracted , for easily searching/finding of a specific document.**

**If the information cannot be found and extracted, a placeholder object is defined so the document is renamed with those variables and know that there is a problem with the extraction.**

**The problem with this is that if we have multiple pdfs that have these values , all of them will have the same name, so after processing them and putting them in the “processed” folder, there will be just the last processed pdf because the rest were overwritten .**

**A computer code with numbers and symbols

Description automatically generated**

**So I modified the placeholder to be unique, handling the situation when we have multiple Pdfs with problems. I added an unique time stamp at the start of the process function , in milliseconds, so at every pdf that is processed there will be a different value. In this way we can store all of the “problem” pdfs and identify them.**

**A screen shot of a computer code

Description automatically generated**

**Let’s test the Efficiency of the new function and compare the Total duration for extracting the required informations from the pdfs. I will put the same invoice, one I will run with the new function skipOCR, and the other one I will run with the old script .**

**The digital pdf (skipOCR) :**

A screen shot of a computer

Description automatically generated

**The digital pdf ( old script) :**

**A screen shot of a computer

Description automatically generated**

**We can see a difference of 9 seconds. It’s not a big difference if processing just a few documents. But when we have more documents to process then the saved time will be efficient.**

**OCR Function inherently is prone to mistakes . ( scan quality , similar characters miss-recognition example: “O instead of 0” etc. )**

**But more important is that the new function makes the processing of the digital documents 100% accurately, the text is extracted without mistakes, so there will be no errors from the digital documents.**

**So I achieved my goal for this optimization project.**