UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Malaysia-Japan
International
Institute of Technology
(MJIIT)

SMJE4263 COMPUTER INTEGRATED MANUFACTURING

Assignment
Extracting Information from Receipts and Invoices

Name: Chew Calvin

Matric No: A19MJ0023

Lecturer: Prof. Madya Ir. Dr. Zool Hilmi bin Ismail

Introduction

Receipts and invoices are vital documents in business transactions, containing critical information such as reference numbers, dates, and amounts. Extracting this information manually can be time-consuming and error-prone. To address this challenge, this report presents a Python code that utilizes Optical Character Recognition (OCR) techniques and regular expressions to automate the extraction process. The code aims to extract key information accurately and efficiently from receipts and invoices, streamlining data processing and analysis.

Methodology

The code employs several libraries and modules to obtain the required information from invoice and receipt. The "re" module is utilized for regular expression matching operations, allowing the code to search for specific patterns within the extracted text. The "pytesseract" library acts as a Python wrapper for the Tesseract OCR engine, providing powerful OCR functionality. The "PIL" (Python Imaging Library) module is used for image manipulation tasks such as opening and processing the images. Additionally, the "os" module facilitates file handling operations, while the "pandas" library provides efficient data structures for data handling and analysis.

The code comprises two main functions:

1. extract_information_from_receipt(image_path)
2. extract_information_from_invoice(image_path)

These functions take the file path of an image as input and execute a series of steps to extract relevant information. Here is an overview of the steps performed by each function:

The image is loaded using the "PIL" library, preparing it for OCR processing. OCR is applied to the image using the "pytesseract.image_to_string()" function, which converts the image into text.

Regular expressions are employed to search for specific patterns within the extracted text. These patterns include reference numbers, invoice number, dates, and amounts. The extracted information, such as the reference number, invoice number, date, and amount, is stored in variables. The extracted information is returned as output from the function.

The code also includes an main section that obtain the receipt and invoice images that are stored in a specific folder. It iterates through the files in the folder, checks for keywords related to invoices or receipts using regular expressions, and calls the appropriate extraction function. The extracted information is stored in separate lists ("invoicetable" and "receipttable") and then converted into Pandas DataFrames ("df1" and "df2") for easier analysis and manipulation.

Results

The code successfully extracts information from receipts and invoices, storing the results in separate DataFrames ("df1" and "df2"). The "df1" DataFrame contains the extracted information from invoices, including the invoice number, date, and amount. The "df2" DataFrame holds the extracted information from receipts, encompassing the reference number, date, and amount. These DataFrames provide a structured representation of the extracted information, facilitating further analysis, reporting, or integration with other systems. Figure 1 shows the output of receipt and invoice in a table form.

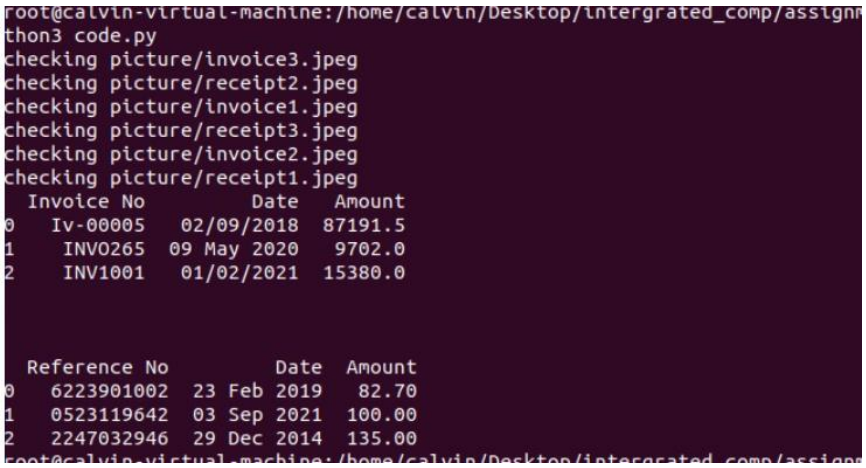

Figure 1: Show the Output from The Program

Discussion

The code effectively utilizes OCR techniques and regular expressions to extract information from receipts and invoices. Regular expressions allow the code to search for specific patterns within the extracted text, providing flexibility in pattern matching. However, it is essential to consider the variations in text format and layout that different receipt or invoice formats may exhibit. The code may require adjustments or enhancements to accurately handle diverse document formats and extract information correctly.

The code leverages the Tesseract OCR engine through the "pytesseract" wrapper, which is a widely-used and reliable OCR solution. OCR technology has significantly improved, but it can still be influenced by factors such as image quality, font style, and layout complexity. To enhance the accuracy of OCR results, preprocessing techniques like image enhancement or filtering could be applied to the images before executing the code for information extraction.

It is crucial to note that the code assumes a specific folder structure and naming conventions for the input files. If the file organization or naming schemes differ, the code may require modifications to locate and process the image files correctly.

Conclusion

In conclusion, the presented Python code offers an automated solution for extracting information from receipts and invoices using OCR techniques and regular expressions. By leveraging the OCR capabilities of the Tesseract engine and incorporating regular expressions, the code accurately identifies and extracts essential information from these documents. The extracted data is structured and stored in separate DataFrames, enabling efficient analysis, reporting, or integration with other systems. However, it is important to consider potential adaptations or enhancements to handle various receipt or invoice formats effectively. Overall, the code provides an efficient and reliable approach to streamline the extraction of information from receipts and invoices, reducing manual effort and improving data processing accuracy.

Appendix

```
import re
import pytesseract
from PIL import Image
import os
import pandas as pd

def extract_information_from_receipt(image_path):
    # Load the receipt image
    image = Image.open(image_path)

    # Apply OCR using Tesseract
    text = pytesseract.image_to_string(image)
    #print(text)
    # Extract store name
    refno = re.search(r'Reference number: (.+)', text)
    refno = refno.group(1) if refno else None
```

```
    if (refno == None and re.search(r'Accepted', text)):
        refno = re.search(r'Accepted\n(\d+)', text)
        if (refno != None):
            refno = refno.group(1)
    if (refno == None):
        refno = re.search(r'Successful\n(\d+)', text)
        if (refno != None):
            refno = refno.group(1)

    # Extract date
    date = re.search(r'(\d{2} .+2\d{3})', text)
    date = date.group(1) if date else None

    # Extract total amount
    total_amount = re.search(r'RM(\d+\.\d+)', text)
    total_amount = total_amount.group(1) if total_amount else None

    # Return extracted information
    return refno,date,total_amount


def extract_information_from_invoice(image_path):
    # Load the receipt image
    image = Image.open(image_path)

    # Apply OCR using Tesseract
    text = pytesseract.image_to_string(image)
    #print(text)
    # Extract store name
    invoiceno = re.search(r'Invoice : (.+)', text)
    invoiceno = invoiceno.group(1) if invoiceno else None
    if (invoiceno == None):
```

```
        invoiceno = re.search(r'INVOICENO —(.+)', text)
        if (invoiceno != None):
                invoiceno = invoiceno.group(1)
    if (invoiceno == None):
        invoiceno = re.search(r'INVOICE, (.+)', text)
        if (invoiceno != None):
                invoiceno = invoiceno.group(1)

    # Extract date
    date = re.search(r"\b(\d{2}/\d{2}/\d{4})\b", text)
    date = date.group(1) if date else None
    if (date == None):
        date = re.search(r'(\d{2} .+2\d{3})', text)
        if (invoiceno != None):
                date = date.group(1)
    if (date == None):
        date = re.search(r'(\d{1} .+1\d{3})', text)
        if (invoiceno != None):
                date = date.group(1)

    # Extract total amount
    pattern = r"(?<!\S)(\d{1,3}(?:,\d{3})*(?:\.\d+)?)(?!\S)"  # Match numbers with commas for thousands and decimals
    numbers = re.findall(pattern, text)

    filtered_numbers = [float(number.replace(",", "")) for number in numbers if "," in number] # Convert matched strings to floats and filter out numbers without commas

    if filtered_numbers:
        max_number = max(filtered_numbers)

    return invoiceno, date,max_number
```

```
folder_path = "picture/"  # Replace with the path to your folder
file_list = os.listdir(folder_path)
invoicetable=[]
receipttable=[]
for file_name in file_list:
    file_path = os.path.join(folder_path, file_name)
    print("checking " + file_path)
    # Load the receipt image
    image = Image.open(file_path)

    # Apply OCR using Tesseract
    textt = pytesseract.image_to_string(image)
    if(re.search(r'invoice', textt) or re.search(r'INVOICE', textt) or re.search(r'Invoice', textt) or
re.search(r'lnvoice', textt)):
            invoiceno, date, max_number = extract_information_from_invoice(file_path)
            newlist = [invoiceno, date, max_number]
            invoicetable.append(newlist)
    else :
            refno,date,total_amount = extract_information_from_receipt(file_path)
            newlist = [refno,date,total_amount]
            receipttable.append(newlist)



headers1 = ["Invoice No", "Date", "Amount"]
headers2 = ["Reference No", "Date", "Amount"]

df1 = pd.DataFrame(invoicetable, columns=headers1)
df2 = pd.DataFrame(receipttable, columns=headers2)
```

```
combined_df = pd.concat([df1, df2], axis=1)

print(df1)
print("\n\n")
print(df2)
```