Transaction Cost Index

Transaction Verification Tool

Introduction

The Transaction Verification Tool processes screenshots or images that contain text using R, and outputs the text along with other information into a CSV file. The tool is adapted to extract information from screenshots of mobile money receipts, but can be adapted for other purposes.

Description of output

Running this file produces a CSV file with eight columns containing the following: The filename of the screenshot, the text in the screenshot, and mobile money transaction details (Agent ID, agent name, account balance, mobile money charge, fee, and tax).

Pre-requisites

The screenshots should be placed in a folder that contains only the image files that are to be processed. The code assumes they are in JPG format.

The quality of the output will depend on the image quality. R might not be able to read screenshots that are blurry, tilted, dark, or noisy. Screenshots can be pre-processed using the *magick* package.

Instructions

1. To start, install the R packages *tesseract*, *magick, tidyverse*, and *stringr*. Then, run the library. Delete the # to run the installation.

```
## 1. INSTALL AND LOAD LIBRARIES
# Uncomment the install.packages() lines to install necessary packages if not already installed.

# install.packages('tesseract')  # For OCR (Optical Character Recognition)
# install.packages('magick')     # For image processing
# install.packages('stringr')    # For advanced string operations

library(tidyverse)  # Data manipulation and tidying
library(stringr)    # String operations
library(tesseract)  # OCR for extracting text from images
library(magick)     # Image reading and manipulation
```

2. Update the file paths.

```
## 2. SET FILE PATHS
# Specify file paths for input images and output data.

# Path to the folder with the image files (assumed to be JPEG files)
image_files <- list.files(path = "C:/Users/imagefolder/", pattern = "\\.jpg$", full.names = TRUE)

# Path to the output CSV file
output_file_path <- "C:/Users/outputfolder/outputfile.csv"
```

3. Initialize the data frame.

```
## 3. INITIALIZE OUTPUT DATA FRAME
# Create an empty data frame to store extracted information
combined_data <- tibble(
  FileName = character(),
  Contents = character(),
  AgentID = character(),
  AgentName = character(),
  Balance = character(),
  Charge = character(),
  Fee = character(),
  Tax = character()
)
```

4. Create a function that cleans the text in the screenshot, removing unwanted characters.

```
# Function to read each image and clean the text
read_file_to_character <- function(file_path) {
  file_contents <- readLines(file_path, warn = FALSE, encoding = "UTF-8")
  # Replace non-ASCII characters with an empty string, but preserve original spacing
  file_contents <- stringi::stri_replace_all_regex(file_contents, "[^\\x20-\\x7E]", "")
  # Collapse the lines into a single string, if necessary
  paste(file_contents, collapse = " ")
}
```

5. Define regular expressions that extract information from the transaction receipt screenshots.

```
## 4. REGULAR EXPRESSION PATTERNS
# Define patterns for extracting various pieces of information from the receipt text.

# Agent ID (e.g., Airtel receipts)
agent_id_pattern <- "Agent\\s*[^:]*:\\s*(\\d+)"

# Agent Name (e.g., MTN receipts)
agent_name_pattern <- "from\\s+([A-Z]+\\s[A-Z]+)"

# Transaction balance
balance_pattern <- "[Bb][a-z]+\\s?:?\\s*UG[Xx]\\s*([0-9]+,?[0-9]+)"

# Transaction charges
charge_pattern <- "[Cc]hargeUGX([0-9]+\\s?,?[0-9]+)"

# Transaction fee
fee_pattern <- "[Ff]ee\\s?i?s?\\s?:?\\s?UGX\\s+([0-9]+,?[0-9]+)"

# Transaction tax
tax_pattern <- "Tax\\s?:?\\s?UGX\\s?([0-9]+,?[0-9]+)"
```

Other regular expressions can be added to extract different details. The file includes some examples that are not part of the output produced by the file.

```
# Additional patterns (examples)
phone_pattern <- "^[+]?(?:[0-9\\-.]\\s?){6,15}[0-9]$"  # Phone numbers
amount_pattern <- "UGX\\s+\\b\\d{1,3}(?:,\\d{3})*(?:\\.\\d+)?\\b"  # Amounts
email_pattern <- "^[a-zA-Z0-9]+(?:\\.[a-zA-Z0-9]+)*@[a-zA-Z0-9-]+\\.[a-zA-Z]{2,}$"  # Emails
date_pattern <- "\\d{2}-\\d{2}-\\d{4}"  # Dates (DD-MM-YYYY)
```

6. Loop through all screenshots, extract and clean the text, and apply the regular expression patterns to extract the transaction details.

```
for (file_path in image_files) {

  text <- read_file_to_character(file_path)  #Clean up text
  image <- image_read(file_path) # Read image
  text <- ocr(image) # extract text using OCR
  file_name <- basename(file_path)  # Extract file name
  balance <- str_extract(text, balance_pattern)  # Extract balance
  balance <- if(!is.na(balance)) str_match(balance, balance_pattern)[,2] else NA
  fee <- str_extract(text, fee_pattern)   # Extract fee
  fee <- if(!is.na(fee)) str_match(fee, fee_pattern)[,2] else NA
  charge <- str_extract(text, charge_pattern)   # Extract charge
  charge <- if(!is.na(charge)) str_match(charge, charge_pattern)[,2] else NA
  tax <- str_extract(text, tax_pattern)   # Extract tax
  tax <- if(!is.na(tax)) str_match(tax, tax_pattern)[,2] else NA
  agent_id <- str_extract_all(text, agent_id_pattern) %>% unlist()   # Extract the fist occurred agent ID
  agent_id <- if(length(agent_id) > 0) str_match(agent_id[1], agent_id_pattern)[,2] else NA
  agentname <- str_extract(text, agent_name_pattern)   # Extract agent name
  agentname <- if(!is.na(agentname)) str_match(agentname, agent_name_pattern)[,2] else NA
```

7. Combine the data.

```
  # Combine extracted data
  combined_data <- bind_rows(
    combined_data,
    tibble(
      FileName = file_name,
      Contents = text,
      AgentID = agent_id,
      AgentName = agent_name,
      Balance = balance,
      Fee = fee,
      Charge = charge,
      Tax = tax
    )
  )
}
```

8. Export the data to CSV

```
## 6. SAVE RESULTS
# Write the extracted data to a CSV file
write_csv(combined_data, output_file_path)
```