MUJ HACKATHON

AI-Driven Financial Document Processing System

- Ankit Anand
(Delhi Technological University)

Abstract

Financial document processing is a time-consuming task that requires manual data extraction, categorization, and analysis. This paper presents an AI-driven solution leveraging **Optical Character Recognition (OCR)** and **Natural Language Processing (NLP)** techniques to automate the extraction, classification, and visualization of financial data from invoices, tax returns, and bank statements. The system processes documents in multiple formats, extracts structured financial data, categorizes transactions, and generates insightful visual summaries for budgeting and financial health assessment.

The solution is built using **R with Tesseract OCR**, **magick for image preprocessing**, and **tidyverse for data manipulation**. It supports **salary slips, ITR-16 forms, cheques, and bank statements**, classifying them intelligently and extracting key financial metrics. The system generates **Excel reports, transaction summaries, and visual dashboards** to aid financial decision-making.

Keywords: OCR, NLP, Financial Automation, Budgeting, R, Tesseract

1. Introduction

1.1 Problem Statement

Manual financial document processing is error-prone and inefficient. Businesses and individuals struggle with:

- Extracting structured data from unstructured financial documents.
- Automating expense categorization for budgeting.
- Generating real-time financial insights.

1.2 Proposed Solution

An AI-driven system that:

1. **Extracts text** from scanned financial documents using OCR.
2. **Classifies documents** (invoices, tax forms, bank statements).
3. **Parses structured data** (amounts, dates, categories).
4. **Generates visual summaries** (trends, expense breakdowns).

2. Methodology

2.1 System Architecture

The system follows a **modular pipeline**:

1. **Document Preprocessing**
    - Image enhancement (grayscale, contrast adjustment).
    - Noise reduction for better OCR accuracy.
2. **OCR Text Extraction**
    - Uses **Tesseract OCR** (trained on financial documents).
    - Handles currency symbols (₹, $) and numeric formats.
3. **Document Classification**
    - Rule-based NLP checks for keywords:
        - *Salary Slip* → "salary", "net pay"
        - *Bank Statement* → "transaction", "balance"
        - *ITR-16* → "PAN", "assessment year"
4. **Structured Data Extraction**
    - **Salary Slips**: Employee ID, net pay, deductions.
    - **Bank Statements**: Transaction dates, amounts, categories.
    - **Tax Forms**: PAN, taxable income, deductions.
5. **Data Analysis & Visualization**
    - Expense categorization (food, rent, utilities).
    - Monthly cash flow trends.
    - Tax liability estimation.

3. Implementation

3.1 Key Functions

1. process_image()

- **Input**: Scanned document (JPG/PNG).
- **Processing Steps**:
    - Converts to grayscale.
    - Enhances contrast.
    - Runs Tesseract OCR.
- **Output**: Raw extracted text.

2. classify_document()

- Uses regex to detect document type:

```
# Document classifier
classify_document <- function(text) {
  case_when(
    str_detect(text, "(?i)salary\\s*slip|payslip") ~ "SALARY_SLIP",
    str_detect(text, "(?i)form\\s*16|itr|income\\s*tax") ~ "ITR_16",
    str_detect(text, "(?i)cheque|check|drawee") ~ "CHEQUE",
    str_detect(text, "(?i)bank\\s*statement|account\\s*summary") ~ "BANK_STATEMENT",
    TRUE ~ "UNKNOWN"
  )
}
```

After Classifying document, it will extract major components from each sheet:

3. Parsers (parse_salary_slip()**,** parse_bank_statement()**, etc.)**

- Extract structured fields:

For example if we take

I)Salary slip:

```
parse_salary_slip <- function(text) {
  # Initialize all fields with NA
  result <- list(
    document_type = "SALARY_SLIP",
    name = NA_character_,
    employee_id = NA_character_,
    payment_date = as.Date(NA),
    gross_earnings = NA_real_,
    total_deductions = NA_real_,
    net_pay = NA_real_
  )

  try({
    result$name <- str_extract(text, "(?i)name\\s*:\\s*([A-Za-z ]+)") %>%
      str_remove("(?i)name\\s*:\\s*") %>% na_if("")

    result$employee_id <- str_extract(text, "(?i)employee\\s*id\\s*:\\s*([A-Za-z0-9-]+)") %>%
      str_remove("(?i)employee\\s*id\\s*:\\s*") %>% na_if("")

    result$payment_date <- str_extract(text, "(?i)payment\\s*date\\s*:\\s*(\\d{1,2}/\\d{1,2}/\\d{4}|\\d{1,2}-[A-
      str_remove("(?i)payment\\s*date\\s*:\\s*") %>%
      dmy() %>% as.Date()

    earnings <- str_extract_all(text, "(?i)(basic\\s*salary|meal\\s*allowance|transport\\s*allowance)[\\s:\\w]*\
      str_extract_all("\\d+[,\\.]\\d{2}") %>%
      unlist() %>%
      str_replace(",", ".") %>%
      as.numeric() %>%
      sum(na.rm = TRUE)

    deductions <- str_extract_all(text, "(?i)(tax|insurance|pf)[\\s:\\w]*\\d+[,\\.]\\d{2}")[[1]] %>%
      str_extract_all("\\d+[,\\.]\\d{2}") %>%
      unlist() %>%
      str_replace(",", ".") %>%
      as.numeric() %>%
      sum(na.rm = TRUE)

    net_pay <- str_extract(text, "(?i)net\\s*pay\\s*[₹$]?\\s*\\d+[,\\.]\\d{2}") %>%
      str_extract("\\d+[,\\.]\\d{2}") %>%
      str_replace(",", ".") %>%
      as.numeric()

    result$gross_earnings <- ifelse(length(earnings) > 0, earnings, NA_real_)
    result$total_deductions <- ifelse(length(deductions) > 0, deductions, NA_real_)
    result$net_pay <- ifelse(length(net_pay) > 0, net_pay, NA_real_)
  })

  return(as.data.frame(result))
}
```

Similarly for the procedure of Cheque , Bank statement and all other statements:

```
# Cheque Parser
parse_cheque <- function(text) {
  cheque_no <- str_extract(text, "(?i)cheque\\s*no\\.?\\s*:\\s*\\d+") %>%
    str_remove("(?i)cheque\\s*no\\.?\\s*:\\s*")

  amount <- str_extract(text, "(?i)amount\\s*[₹$]?\\s*\\d+[,\\.]\\d{2}") %>%
    str_extract("\\d+[,\\.]\\d{2}") %>%
    str_replace(",", ".") %>%
    as.numeric()

  payee <- str_extract(text, "(?i)pay\\s*:\\s*([A-Za-z ]+)") %>%
    str_remove("(?i)pay\\s*:\\s*")

  date <- str_extract(text, "\\d{2}/\\d{2}/\\d{4}") %>%
    dmy()

  list(
    document_type = "CHEQUE",
    cheque_number = cheque_no,
    amount = amount,
    payee_name = payee,
    date = date
  )
}

# Bank Statement Parser
parse_bank_statement <- function(text) {
  account_no <- str_extract(text, "(?i)account\\s*no\\.?\\s*:\\s*[\\d-]+") %>%
    str_remove("(?i)account\\s*no\\.?\\s*:\\s*")

  transaction_lines <- str_extract_all(text, "\\d{2}/\\d{2}/\\d{4}\\s+[^\\n]+?\\s+[\\d,]+\\.[\\d]{2}\\s+[\\d,]+\\.

  if (length(transaction_lines) > 0) {
    transactions <- map_dfr(transaction_lines, function(line) {
      parts <- str_split(line, "\\s{2,}")[[1]]
      data.frame(
        date = dmy(parts[1]),
        description = parts[2],
        debit = ifelse(length(parts) > 2, as.numeric(str_remove_all(parts[3], ",")), NA),
        credit = ifelse(length(parts) > 3, as.numeric(str_remove_all(parts[4], ",")), NA),
        stringsAsFactors = FALSE
      )
    }) %>%
      mutate(
        amount = ifelse(is.na(debit), credit, -debit),
        category = case_when(
          str_detect(description, "(?i)salary|payroll") ~ "Salary",
```

Then afterwards, it analyse the finances of the company, indivuval:

4. analyze_finances()

- Aggregates data into trends:

```r
analyze_finances <- function(processed_data) {
  # Salary analysis
  salary_summary <- if (!is.null(processed_data$salary_slips) {
    processed_data$salary_slips %>%
      mutate(
        month = ifelse(!is.na(payment_date),
                       floor_date(payment_date, "month"),
                       as.Date(NA))
      ) %>%
      filter(!is.na(month)) %>%  # Only include records with valid dates
      group_by(month) %>%
      summarize(
        avg_gross = mean(gross_earnings, na.rm = TRUE),
        avg_net = mean(net_pay, na.rm = TRUE),
        avg_tax = mean(total_deductions, na.rm = TRUE),
        count = n()
      ) %>%
      filter(count > 0)  # Only include months with data
  } else NULL


  # ITR analysis
  itr_summary <- if (!is.null(processed_data$itr_16_forms)) {
    processed_data$itr_16_forms %>%
      summarize(
        avg_gross_salary = mean(gross_salary, na.rm = TRUE),
        avg_tax_paid = mean(tax_payable, na.rm = TRUE),
        tax_rate = mean(tax_payable/gross_salary, na.rm = TRUE),
        count = n()
      )
  } else NULL
```

In this it extract basic components like Average gross earnings, net pay, deduxtions from a month particularly from all type of four statements I have provided

But since every statement have some unique variables, so from particularly from ITR statements it extrcats: gross salary, avg tax paid, tax rate , etc

```r
  # ITR analysis
  itr_summary <- if (!is.null(processed_data$itr_16_forms)) {
    processed_data$itr_16_forms %>%
      summarize(
        avg_gross_salary = mean(gross_salary, na.rm = TRUE),
        avg_tax_paid = mean(tax_payable, na.rm = TRUE),
        tax_rate = mean(tax_payable/gross_salary, na.rm = TRUE),
        count = n()
      )
  } else NULL
```

Similarly from Transaction or bank statement, it extracts:

```r
# Transaction analysis
transaction_summary <- if (!is.null(processed_data$all_transactions) && nrow(processed_data$all_transactions) >
  processed_data$all_transactions %>%
    mutate(month = floor_date(date, "month")) %>%
    group_by(month, category) %>%
    summarize(
      total_amount = sum(amount, na.rm = TRUE),
      avg_amount = mean(amount, na.rm = TRUE),
      count = n(),
      .groups = "drop"
    )
} else NULL

list(
  salary_summary = salary_summary,
  itr_summary = itr_summary,
  transaction_summary = transaction_summary
)
```

5. Visualization (plot_expense_categories(), plot_cash_flow())

- Uses **ggplot2** for:

```r
# Visualization functions
plot_salary_trends <- function(salary_summary) {
  if (is.null(salary_summary)) return(NULL)

  ggplot(salary_summary, aes(x = month)) +
    geom_line(aes(y = avg_gross, color = "Gross Salary"), linewidth = 1) +
    geom_line(aes(y = avg_net, color = "Net Salary"), linewidth = 1) +
    labs(title = "Monthly Salary Trends", x = "Month", y = "Amount", color = "Metric") +
    scale_color_manual(values = c("Gross Salary" = "blue", "Net Salary" = "green")) +
    theme_minimal()
}

plot_expense_categories <- function(transaction_summary) {
  if (is.null(transaction_summary)) return(NULL)

  category_totals <- transaction_summary %>%
    filter(total_amount < 0) %>%
    group_by(category) %>%
    summarize(total = sum(abs(total_amount)))

  ggplot(category_totals, aes(x = reorder(category, total), y = total, fill = category)) +
    geom_col() +
    coord_flip() +
    labs(title = "Expenses by Category", x = "Category", y = "Total Amount") +
    theme_minimal() +
    theme(legend.position = "none")
}

plot_cash_flow <- function(transaction_summary) {
  if (is.null(transaction_summary)) return(NULL)

  monthly_flow <- transaction_summary %>%
    group_by(month) %>%
    summarize(
      income = sum(total_amount[total_amount > 0], na.rm = TRUE),
      expenses = sum(abs(total_amount[total_amount < 0]), na.rm = TRUE),
      net = income - expenses
    )

  ggplot(monthly_flow, aes(x = month)) +
    geom_col(aes(y = income, fill = "Income"), alpha = 0.7) +
    geom_col(aes(y = -expenses, fill = "Expenses"), alpha = 0.7) +
    geom_line(aes(y = net, color = "Net Flow"), linewidth = 1) +
    labs(title = "Monthly Cash Flow", x = "Month", y = "Amount", fill = "Type", color = "Metric") +
    scale_fill_manual(values = c("Income" = "darkgreen", "Expenses" = "darkred")) +
    scale_color_manual(values = c("Net Flow" = "blue")) +
    theme_minimal()
}
```

And at the end talking about the prototype, using NLP & OCR, I framed this main function, which embedded with all the sub functions that I mentioned above.

```r
# Main function to run full analysis
run_full_analysis <- function(folder_path, output_dir = "financial_reports") {
  dir.create(output_dir, showWarnings = FALSE)

  message("Processing documents...")
  processed_data <- process_financial_documents(folder_path)

  message("Analyzing data...")
  analysis <- analyze_finances(processed_data)

  message("Generating visualizations...")
  if (!is.null(analysis$salary_summary)) {
    p <- plot_salary_trends(analysis$salary_summary)
    if (!is.null(p)) ggsave(file.path(output_dir, "salary_trends.png"), p, width = 10, height = 6)
  }

  if (!is.null(analysis$transaction_summary)) {
    p <- plot_expense_categories(analysis$transaction_summary)
    if (!is.null(p)) ggsave(file.path(output_dir, "expense_categories.png"), p, width = 10, height = 6)

    p <- plot_cash_flow(analysis$transaction_summary)
    if (!is.null(p)) ggsave(file.path(output_dir, "cash_flow.png"), p, width = 10, height = 6)
  }

  message("Generating Excel report...")
  wb <- createWorkbook()

  if (!is.null(processed_data$salary_slips)) {
    addWorksheet(wb, "Salary Summary")
    writeData(wb, "Salary Summary", processed_data$salary_slips)
  }

  if (!is.null(processed_data$itr_16_forms)) {
    addWorksheet(wb, "ITR Summary")
    writeData(wb, "ITR Summary", processed_data$itr_16_forms)
  }

  if (!is.null(processed_data$cheques)) {
    addWorksheet(wb, "Cheque Summary")
    writeData(wb, "Cheque Summary", processed_data$cheques)
  }

  if (!is.null(processed_data$all_transactions) && nrow(processed_data$all_transactions) > 0) {
    addWorksheet(wb, "Transactions")
    writeData(wb, "Transactions", processed_data$all_transactions)
  }

  addWorksheet(wb, "Analysis")
  analysis_data <- list()
```

And at last, It forms all the analysis in the single excel file, through which we can further forms the analysis:

```r
  writeData(wb, "Analysis", analysis_data, startCol = 1, startRow = 1, colNames = TRUE)
  saveWorkbook(wb, file.path(output_dir, "financial_analysis.xlsx"), overwrite = TRUE)

  message("Saving raw data...")
  saveRDS(processed_data, file.path(output_dir, "processed_data.rds"))
  saveRDS(analysis, file.path(output_dir, "financial_analysis.rds"))

  message(paste("Analysis complete! Results saved to:", output_dir))
  return(list(processed_data = processed_data, analysis = analysis))
}

# Run the analysis
results <- run_full_analysis("C:/Users/ANKIT ANAND/Downloads/FLD/Utility")
```

6. Conclusion

The system successfully automates financial document processing, reducing manual effort. Future enhancements in deep learning will improve accuracy further.