A chess piece with a shadow

Description automatically generated

BANK STATEMENTS e-PDF INFORMATION EXTRACTION

February 2024

***Table of Contents***

|  |  |  |
| --- | --- | --- |
|  | *Overview* | *Pg 3* |
|  | *Workflow* | *Pg 4* |
|  | *Methodology*  *3.1 Text, Table, and Image Extraction Libraries*  *3.2 OCR Libraries*  *3.3 Bank Statement Classification Module*  *3.4 Table Extraction*  *3.5 Information Extraction* | *Pg 5*  *Pg 6*  *Pg 6*  *Pg 8*  *Pg 10* |
|  | *Evaluation Metrics* | *Pg 11* |
|  | *Test Results* | *Pg 12* |
|  | *Conclusion* | *Pg 15* |
|  |  |  |
|  |  |  |

1. Overview

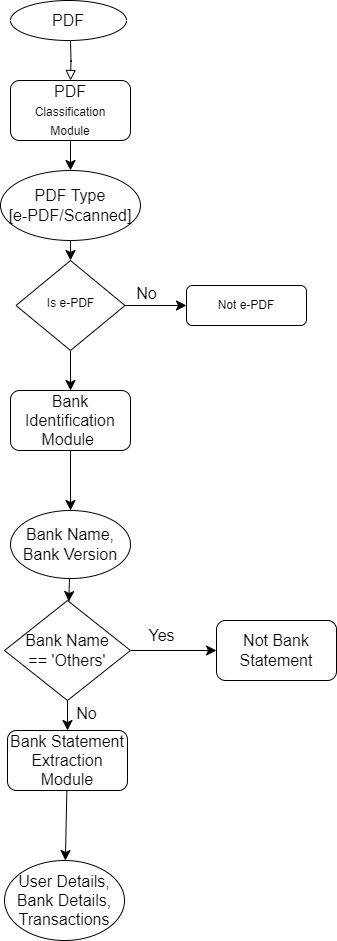
This project's core focus is to streamline the process of identifying, categorizing, and extracting e-PDF statements submitted by users. This entails a multi-step approach where we first ascertain whether a file is indeed an e-PDF document and subsequently determine if it constitutes a bank statement. Additionally, we classify these statements based on the specific financial institution they originate from. Once these initial steps are completed, the subsequent phase involves employing tailored extraction methods designed for each bank. By utilizing bank-specific extractors, we can effectively retrieve pertinent user details and transaction data encapsulated within the e-PDF file.

A graph of a number of blue bars

Description automatically generated

Based on the analysis of the test data provided, it was observed that the top 10 banks, according to their frequency of occurrence, collectively covered approximately 82% of the dataset. This indicates that developing extractors specifically tailored to these top 10 banks would offer extensive coverage, capturing a significant portion of the data. Also, it was observed that each bank on average had 3 to 5 different bank statement versions for which the custom extractor function needs to be written.

1. Workflow



***PDF Classification Module –*** The PDF Classification Module will take PDF file path as input and classifies it as either e-PDF or Scanned.

***Bank Identification Module –*** The Bank Identification Module receives the path of a valid e-PDF file as input. It determines the bank name associated with the statement and identifies the bank statement version of the PDF if the bank name is not ‘Others’.

***Bank Statement Extraction Module –*** The Bank Statement Extraction Module requires the PDF file path, Bank name and Version as input. It then extracts the details and transactions from the PDF accordingly.

1. Methodology

**Libraries Evaluation and Selection**

**Text, Table and Image Extraction Libraries Explored**

* [Pdfplumber](https://pypi.org/project/pdfplumber/) –
  + Python library for data, table and image extraction from pdf files.
  + Pdfminer.six library provides the foundation for pdfplumber. It primarily focuses on parsing PDFs, analyzing PDF layouts and object positioning, and extracting text. It provides tools for table extraction or visual debugging.
* [PyMuPDF](https://pypi.org/project/PyMuPDF/) –
  + A high-performance Python library for data extraction, analysis, conversion & manipulation of PDF (and other) documents.
  + PyMuPDF is substantially faster than pdfminer.six (and thus also Pdfplumber) and can generate and modify PDFs, but the library requires installation of non-Python software (MuPDF). It also does not enable easy access to shape objects (rectangles, lines, etc.), and does not provide table-extraction or visual debugging tools.
* [Camelot-py](https://pypi.org/project/camelot-py/) –
  + Python library which primarily focuses on table extraction from pdf files.
  + Camelot focuses primarily on extracting tables. In some cases, it might be better suited to the table you are trying to extract. But it does not provide tools for image and text extraction.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **PyMuPDF** | **Pdfplumber** | **Camelot-py** |
| Text Extraction | Yes (faster) | Yes | No |
| Table Extraction | No | Yes | Yes (faster) |
| Image Extraction | Yes | Yes | No |

***After evaluating all the 3 libraries for text, table and image extraction, Pdfplumber emerged as the most suitable option for meeting all our requirements efficiently. Pdfplumber excelled in fulfilling all the necessary criteria, providing comprehensive support for both image and table extraction. Moreover, Pdfplumber's documentation proved to be superior compared to alternative libraries, ensuring ease of use and seamless integration into our workflow. Therefore, Pdfplumber stands out as the preferred choice, offering a streamlined solution that aligns perfectly with our needs.***

**OCR Libraries Explored**

* [**Pytesseract**](https://pypi.org/project/pytesseract/) **–** 
  + Python-tesseract is a python wrapper for Google's Tesseract-OCR.
* [**EasyOCR**](https://pypi.org/project/easyocr/) **–** 
  + End-to-End Multi-Lingual Optical Character Recognition (OCR) Solution.

***After trying out the mentioned OCR libraries, we found that EasyOCR performed better compared to Pytesseract for noisy images and images where text is present on unconventional background. Also, EasyOCR provided GPU support and an option to process images in batch.***

**Bank Statement Classification**

**Approaches Tried**

* ***IFS Code Method –***

This methodology relies on the assumption that the IFS Code (Indian Financial System Code) for a particular bank will have the same four-letter prefix. For instance, for 'State Bank of India', the IFS Codes are of the format 'SBIN\*\*\*\*\*\*'. The approach utilizes the pdfplumber library to open the PDF file and extract the header of the first page. From this header, the methodology employs regular expressions (regex) to identify the IFS Code. By analyzing the first page's header, the methodology attempts to extract and validate the IFS Code, providing a means to detect the bank associated with the document.

* ***Using Bank Logo –***

This methodology assumes that each bank statement will include the bank's logo. The approach involves utilizing pdfplumber to open the PDF file and extracting all the images present on the first page. Subsequently, with the aid of easyOCR, the methodology attempts to extract text from the logo image(s). By analyzing the extracted text from the logo image, the methodology identifies the bank associated with the document. Additionally, by utilizing the coordinates of the bank logo, the methodology offers a means to version bank statements for a particular bank, enhancing document management and identification processes.

A blue circle with orange and black text

Description automatically generated

***Based on testing 500 files, found out that ‘Bank Logo’ method provided better accuracy than ‘IFSC Method’ as many bank statements did not include the bank IFS Code in it’s header. OCR also did not give 100% accuracy as there were some formats in which bank logo was not present. But there were no formats in which both logo and IFSC Code are not present. So using both the approaches in combination while giving priority to ‘Bank Logo’ method will provide better results.***

A green pie chart with white text

Description automatically generated

**PDF Table Extraction**

**Approaches Tried**

* ***Co-ordinate Based Approach -***

In the coordinate-based approach, we utilize the positions of columns within the transactions table to extract data from the bank statement PDF. Here's how it works:

*Coordinates (x0 and x1):*

* Each column in the table has two horizontal coordinates associated with it: 'x0' and 'x1'.
* 'x0' represents the starting point of the column horizontally, while 'x1' represents the end point.

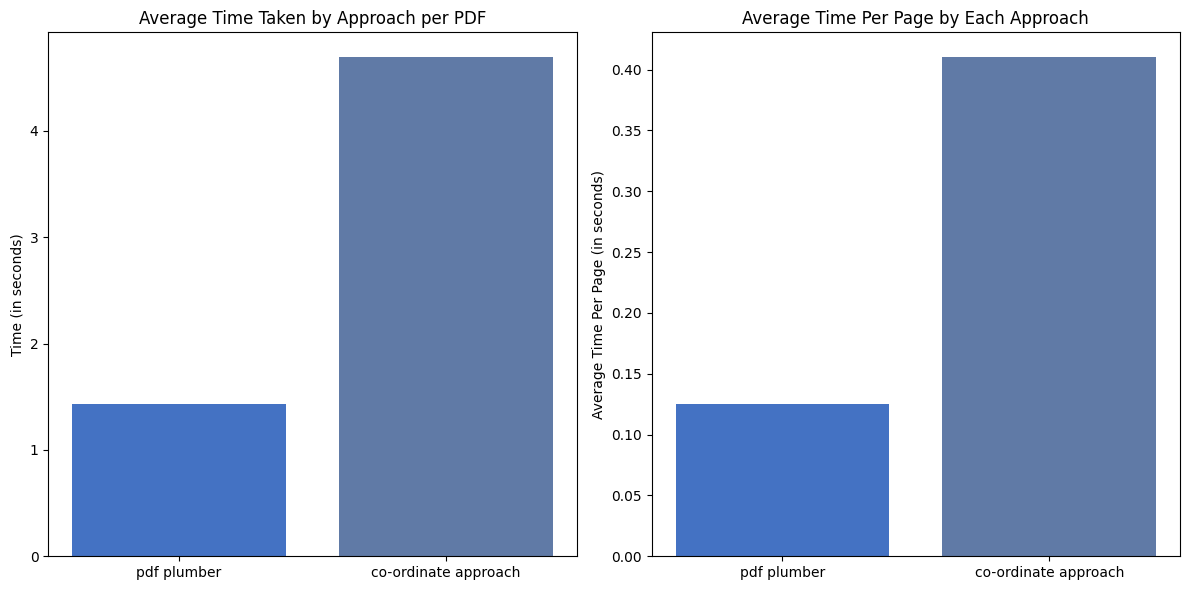
*Start and End Points:*

* Additionally, we identify specific text markers within the document that signify the start ('starts\_after') and end ('ends\_before') points of the table.
* These markers are typically strings of text that appear before and after the table in the document.

*Extraction Process:*

* By providing the 'x0' and 'x1' coordinates for each column along with the 'starts\_after' and 'ends\_before' strings to a general function, we can determine the location of the table on the page.
* Using this information, we can precisely extract the table data from the PDF.
* ***Using Pdf-plumber*** –

In this approach, Pdf-plumber's extract\_method() is utilized to extract the table from the PDF. However, different bank statement versions might have varying designs, such as some having lines separating columns and rows, while others may not. To handle this variability, the table\_settings parameter in extract\_method() needs to be adjusted according to the specific bank statement version being dealt with.



***After testing both the approaches, pdf plumber approach was 70% faster compared to the co-ordinate approach. So, we decided to move forward with the pdf plumber approach.***

**PDF Information Extraction**

**Approaches Tried**

***Regular Expression:***

* This method involves using regular expressions directly on the text extracted from the bank statements.
* It's a straightforward approach where predefined patterns are matched against the text to extract relevant information.
* However, it may fail to extract information correctly in cases where the text is structured in a complex manner, such as when the information is located within information boxes or tables.

***Information Box Co-ordinates:***

* This method addresses the limitations of Approach 1 by identifying the coordinates of information boxes on the document.
* Once the coordinates of the information boxes are known, the text within these boxes can be extracted directly. This ensures that the extracted text remains structured and coherent, avoiding the mixing up of text from different sections of the document.
* By focusing only on the text within the identified information boxes, this approach improves the accuracy of information extraction, especially in cases where the information is presented in a non-linear or structured format.

In summary, while ‘Regular Expression’ approach relies on direct pattern matching using regular expressions, ‘Information Box Co-ordinates’ approach enhances the accuracy of information extraction by targeting specific information boxes based on their coordinates, thus overcoming the limitations of ‘Regular Expression’ approach in handling structured text formats like information boxes.

***As ‘Regular Expression’ approach is faster than ‘Information Box’ approach it is preferred, but in places where ‘Regular Expression’ cannot be used ‘Information Box’ approach is being used.***

1. Evaluation Metrics

***1. Initial Dataset:***

Our test [dataset](https://growthsourceft-my.sharepoint.com/:x:/g/personal/aditya_m_protium_co_in/EZ5_3CpJabJCqm-m15n31f0BK3Hr5H-nwiuLTzVPt2DCtQ?e=noQcBI) comprised approximately 65,000 PDF files. Among these, 46,000 files were PDFs without any password protection, making them suitable for testing.

***2. Ground Truth Acquisition:***

Initially, we utilized the Perfios-ID of the PDF files to extract information such as bank name, account details, and transaction data. This served as our benchmark or ground truth for evaluating the performance of our code.

***3. Bank-Wise Testing:***

In the subsequent step, we focused on PDF files from specific banks and conducted testing on a bank-by-bank basis.

***4. Comparison Criteria:***

To assess the accuracy of our code, we compared the output of our system with the data extracted by Perfios. Four key parameters were used for comparison:

* Credit Count
* Debit Count
* Total Credit
* Total Debit

***5. Success Criteria:***

A successful table extraction was determined when all four parameters matched precisely with the data provided by Perfios. This ensured that our system accurately captured the transactional information from the PDF files.

1. Test Results
2. ***State Bank of India Test Results***

***A green and blue circles with black text

Description automatically generated***

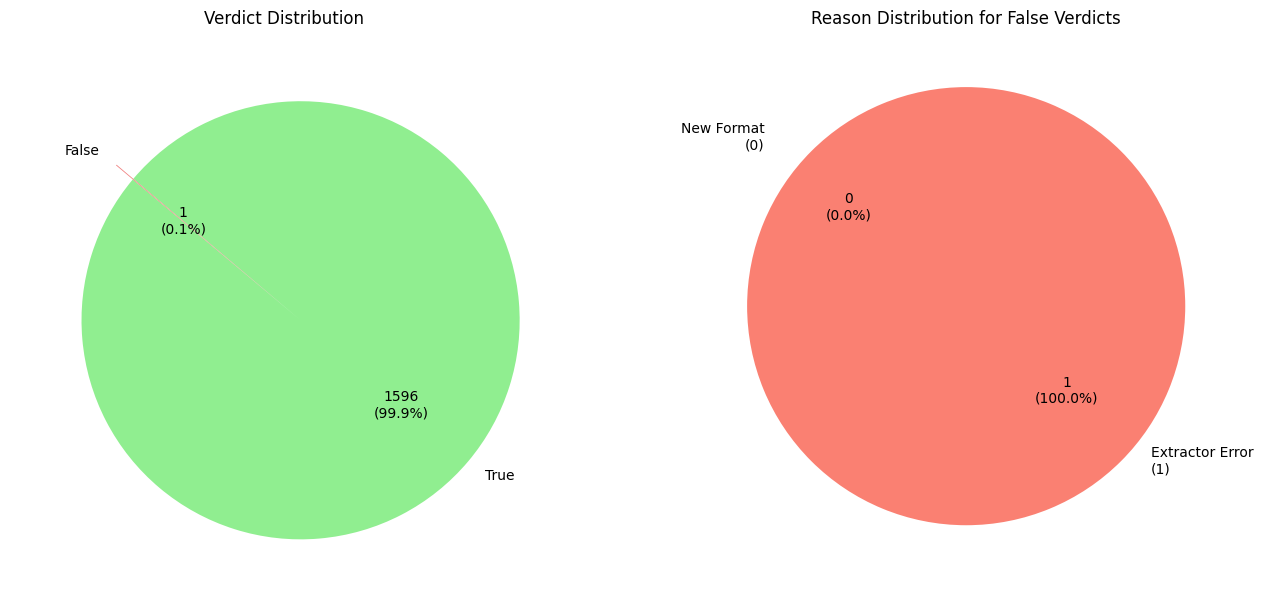
Results: [State Bank of India\_final\_results.csv](https://growthsourceft-my.sharepoint.com/:x:/g/personal/aditya_m_protium_co_in/EQloToNhD91LogcPKhyXCqQBwcizaHqSxQGrUaAuByvAvw?e=7S31OP)

The SBI bank statement extractor currently is coded to handle the major 4 formats which we found in the train dataset.

After testing on 1557 State Bank of India (SBI) bank statements of different types, our SBI extractor module successfully matched 1491 statements exactly as expected by Perfios. However, we found issues in the remaining 66 files.

Out of these, 19 had some differences compared to Perfios output as in these files user had merged multiple bank statements into a single pdf file leading to error, we labeled them as "Extractor Error." Additionally, 47 statements were in a new format that our extractor wasn't designed to handle currently.

1. ***HDFC Bank Results***

******

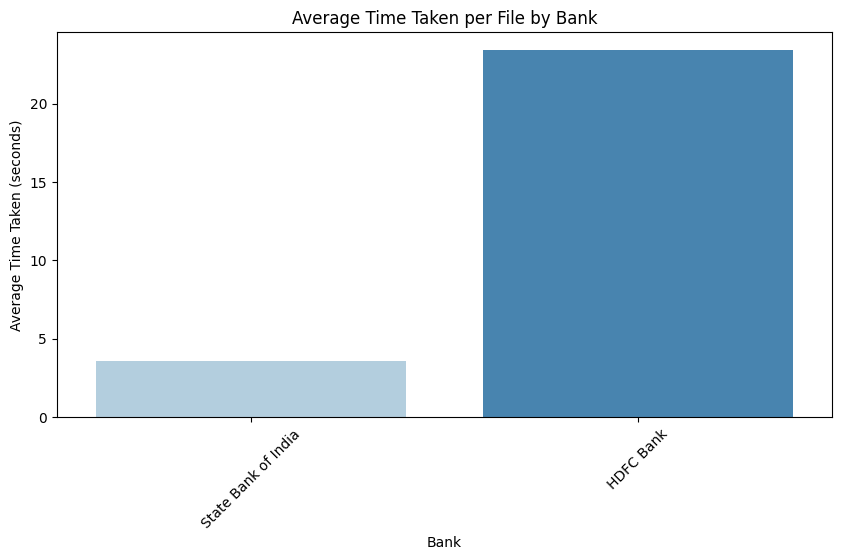
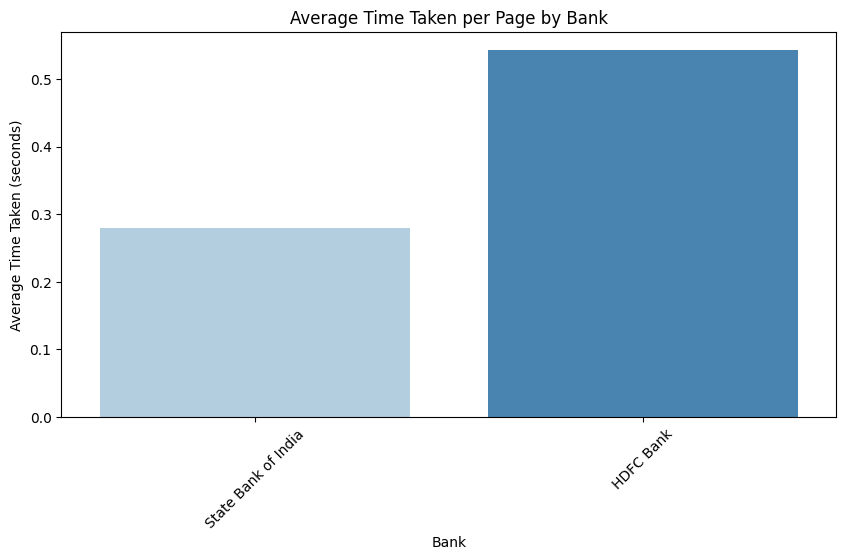
Results : [HDFC\_final\_results.csv](https://growthsourceft-my.sharepoint.com/:x:/g/personal/aditya_m_protium_co_in/EaLTja_UkwdLmafsWhEQzukBGJUD5-9PQkSd8XE3nVhQdg?e=6vCI8M)

The HDFC bank statement extractor currently is coded to handle the major 4 formats which we found in the train dataset.

After carefully testing 1597 State Bank of India (SBI) bank statements of different types, our SBI extractor module successfully matched 1596 statements exactly as expected by Perfios.

For the one failed case, Perfios gave some extra transactions which were not actually present in the given bank statement.

1. ***Time Analysis***

*** ***

1. Conclusion

***Further Work***

* Optimize the OCR Code to process images faster.
* Try to use multiprocessing so table extraction from very large pdf files becomes faster.
* Make the code production ready by using good coding practices and design patterns.
* Write extractor for top 10 most frequent banks and then perform bank wise testing.­­­­­­­­

***References***

* ***OCR***
  + [easyocr · PyPI](https://pypi.org/project/easyocr/)
  + [EasyOCR vs Tesseract vs Amazon Textract: an OCR engine comparison - (francescopochetti.com)](https://francescopochetti.com/easyocr-vs-tesseract-vs-amazon-textract-an-ocr-engine-comparison/)
* ***Pdf table and details extraction*** 
  + [pdfplumber · PyPI](https://pypi.org/project/pdfplumber/)
  + [jsvine/pdfplumber (github.com)](https://github.com/jsvine/pdfplumber)