Title: Automated Extraction and Visualization of Bank Statements

Author: Dhananjay Ramrao Ambatwar
dhananjayambatwar7@gmail.com

1. Introduction

With the rapid digitalization of financial services, most banks now provide their customers with account statements in PDF format. While these files are human-readable, they are often not directly machine-readable due to variations in formatting, multi-line transactions, and inconsistent layouts. This poses a challenge for organizations and individuals who want to automatically analyze financial activity for insights such as expense tracking, fraud detection, or budgeting.

Bank statements are semi-structured documents, typically containing two major sections: account details (metadata such as account number, IFSC, and account type) and detailed transaction history (date-wise records of withdrawals, deposits, and balances).

Primary Goals of This Project:

- Develop a robust Python-based pipeline to ingest PDF bank statements and automatically extract structured account and transaction data.
- Clean and transform the extracted information into standardized CSV files for account details and transactions.
- Implement transaction analysis and flagging to identify high-value transactions and transactions from specific entities.
- Generate a visualization timeline to track deposits and withdrawals over time for insights.

2. Methodology

The methodology was designed to reliably convert raw PDF statements into structured datasets and visual insights.

2.1 Initial Exploration

Several Python libraries were tested for PDF handling:

- pdfplumber – Extracts text and tables from PDFs with embedded text.
- PyPDF2 – Reads raw text but has limited support for structured tables.
- Camelot / Tabula – Specialized in table extraction but inconsistent across variable bank formats.

Observation: These libraries provided partial success, but multi-line descriptions and inconsistent layouts often led to incomplete or inaccurate extraction. To overcome this, a custom pipeline combining pdfplumber with regex-based parsing was developed, providing flexibility and precise control over the extracted data.

2.2 Account Information Extraction

Account metadata such as Account Number, Account Holder Name, IFSC, MICR, Account Type, Bank Name, and Address were extracted from the first page of the PDF.

Approach:

- Opened the first page using pdfplumber.
- Applied regex patterns to locate each field.
- Stored extracted metadata in a structured dictionary and exported it as a CSV file.

This ensures every dataset is correctly linked to the corresponding account.

2.3 Transaction Table Extraction

The focus was on extracting transaction rows with the following fields: transaction_date, description, withdrawal_amount, deposit_amount, balance.

Approach:

- Iterated through transaction pages using pdfplumber.
- Extracted rows from tables while handling multi-line descriptions.
- Replaced missing withdrawal/deposit values with zero and cleaned numeric fields.
- Assembled results into a pandas DataFrame, exported as a CSV.

2.4 Data Cleaning and Preprocessing

- Standardized column names and formats.
- Converted dates to datetime objects and numeric fields to float.
- Filled missing values and sorted transactions chronologically.
- Ensured consistency and reliability for downstream analysis and visualization.

2.5 Transaction Analysis and Flagging

Using the structured transactions:

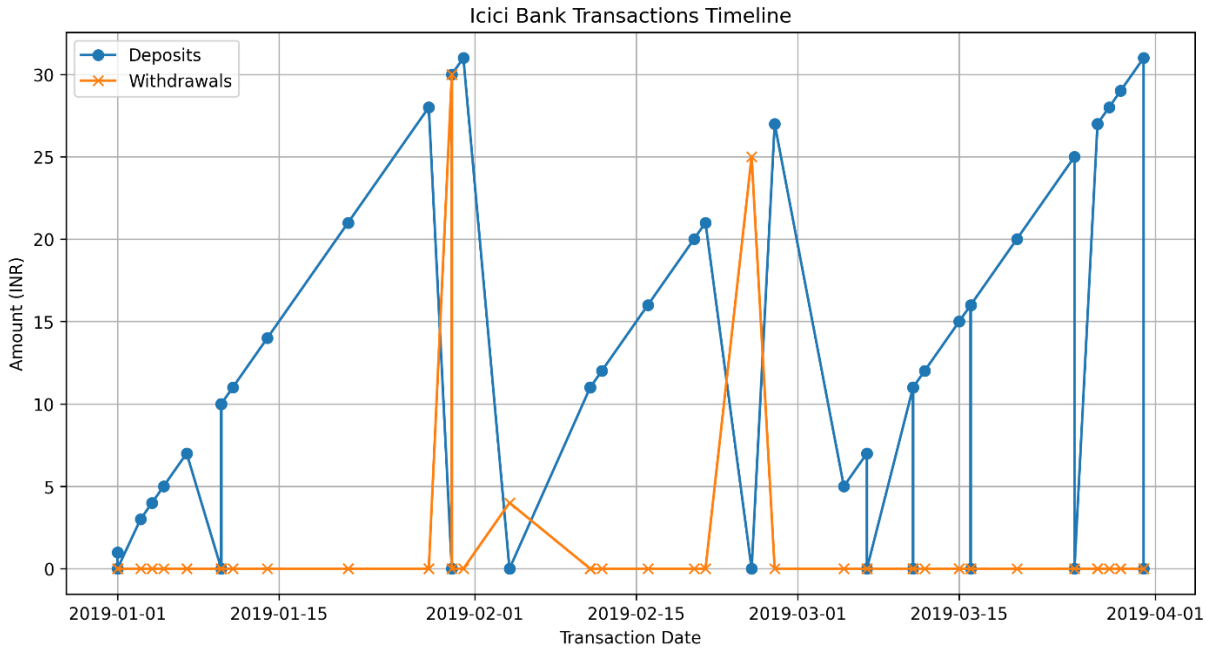
- Flagged large DD withdrawals (above INR 10,000).
- Flagged large RTGS deposits (above INR 50,000).
- Flagged transactions with specific entities (“Guddu”, “Prabhat”, “Arif”, “Coal India”), case-insensitive.

These analyses enable quick identification of high-value transactions and entity-specific interactions.

2.6 Visualization

A timeline plot was generated to visualize deposits and withdrawals over time:

- Example: For ICICI bank statement.
- Approach:
    - Used Matplotlib to plot deposits and withdrawals.
    - X-axis: Transaction dates.
    - Y-axis: Transaction amounts.
    - Added title, labels, and legend.
    - Saved as icici_timeline.png.



3. Challenges Faced

- Low Accuracy with Default Libraries: Libraries like PyPDF2, Camelot, and Tabula struggled with multi-line or non-standard tables.
- Multi-line Descriptions: Required careful merging logic to avoid splitting transactions incorrectly.
- Inconsistent PDF Layouts: Different banks and statements vary in structure, making regex patterns sensitive.
- Preprocessing Effort: Cleaning numeric fields, filling missing values, and standardizing columns took significant effort.
- Scalability: Handling multiple bank formats and ensuring pipeline flexibility for new PDFs is non-trivial.

4. Why This Approach?

- pdfplumber + Regex: Provided fine-grained control over data extraction from semi-structured PDFs.
- pandas: Allowed robust cleaning, transformation, and export to structured CSV formats.
- Matplotlib: Enabled clear and interpretable visualizations, with precise control over formatting and plotting.
- Custom Analysis Functions: Made it possible to flag high-value transactions and entity-specific activity, adding practical utility to the dataset.

This combination was chosen for accuracy, flexibility, and interpretability, ensuring the pipeline works reliably across multiple PDFs and can be extended in future for automated financial analytics.

5. Future Scope

- Deep Learning for Table Parsing: DeepDeSRT or TableNet models could handle complex and inconsistent table layouts.
- LLM + RAG-based Extraction: Large Language Models combined with document retrieval can automate extraction from diverse bank statements.
- Generalization Across Banks: Extend pipeline to handle multiple banks and PDF formats

6. Conclusion

This project developed a semi-automated pipeline to extract and visualize bank statements in PDF format. Using pdfplumber, regex, pandas, and matplotlib, raw semi-structured PDFs were successfully converted into structured datasets and meaningful visualizations.

The results demonstrate that while rule-based methods are effective for specific formats, future enhancements using deep learning and LLMs can significantly improve generalization and automation. This work thus forms a solid foundation for scalable financial document analytics.