Technical Documentation

BigDataIA-Fall2023-Team8/Assignment_01

Authors: Soham Deshpande, Tanmay Zope, Anvi Jain

Date: Friday, October 6th, 2023

Contribution & Attestation

Soham Deshpande:	33%	PyPDF, Nougat Tunneling, Streamlit App for Part A, Tech Doc
Tanmay Zope:	33%	Pandas Profiling Origination, Great Expectations, Streamlit App for Part B
Anvi Jain:	33%	Pandas Profiling Monthly Performance, Streamlit App for Part B, Tech Doc

WE ATTEST THAT WE HAVEN'T USED ANY OTHER STUDENTS' WORK IN OUR ASSIGNMENT AND ABIDE BY THE POLICIES LISTED IN THE STUDENT HANDBOOK

Table of Contents

Part A

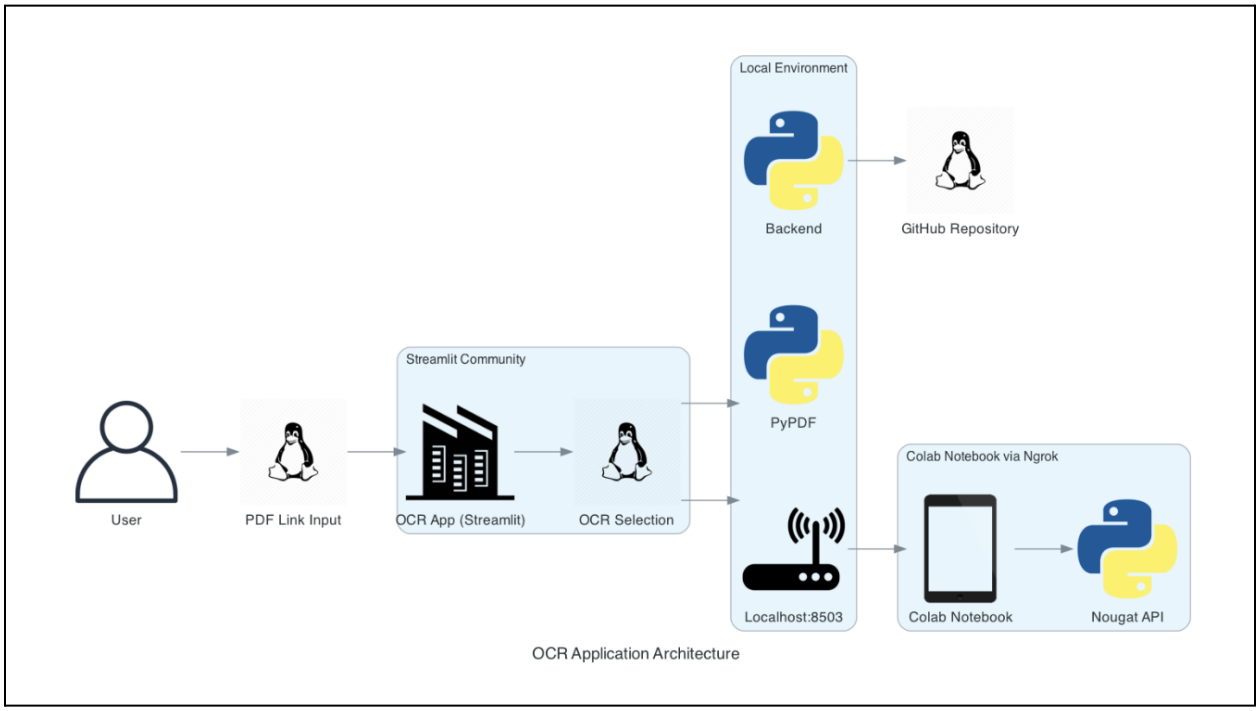
Instructions

Your team is tasked with building a tool for analysts to be able to load a pdf document and get a summary. You are evaluating nougat[3] or pypdf[5] as possible libraries. The files will be coming from SEC[4].

- Build a streamlit based app that can take the link of the pdf file and use either of the libraries (The user should be able to select which one)
- Try out 5-10 different pdf files from the SEC[4] site
- Provide your recommendations for pros/cons and which tool you would recommend.
- Create and embed the architecture of the project using diagrams[6] within Streamlit

Homepage

Design Architecture



OCR App Architecture

Streamlit

- Multi Page Streamlite OCR Application:
    - Architecture:
        - The above architecture has been created on the first page using Python Package - diagrams.

- Design Order

```
user >> pdf_link >> app
app >> select_ocr
select_ocr >> pypdf
select_ocr >> api_router >> colab >> nougat_api
backend >> github
```

- Main Code File:
  - This file contains the actual streamlit application which includes the functions of the 2 OCR Packages, their functions and the main application code.
  - PyPDF2 Code Snippet

```
def perform_pypdf_ocr(pdf_file):
    pdf_text = ""
    # PyPDF2-based OCR
    pdf_reader = PdfReader(io.BytesIO(pdf_file))
    for page_num in range(len(pdf_reader.pages)):
        page = pdf_reader.pages[page_num]
        pdf_text = pdf_text + page.extract_text()
    return pdf_text
```

  - Nougat Code Snippet

```
def perform_nougat_ocr(pdf_file):
    try:
        # Perform OCR using Nougat API
        #url = "http://127.0.0.1:8503/predict/"
        url = "https://0728-34-125-184-65.ngrok-free.app/predict"
        files = {'file': pdf_file}
        response = requests.post(url, files=files)

        if response.status_code == 200:
            pdf_text = response.text
            # Display the OCR output
            output_placeholder.write("OCR Output (Nougat):")
            output_placeholder.write(pdf_text)
        else:
            st.error(f"Failed to perform OCR (Nougat). Status code: {response.status_code}")
    except Exception as e:
        st.error(f"An error occurred (Nougat): {str(e)}")
```

  -
  - Running both functions on streamlit:

```python
if st.button("Perform OCR"):
    if url:
        try:
            # Fetch the PDF content from the URL
            response = requests.get(url)
            if response.status_code == 200:
                pdf_content = response.content

                if option == "PyPDF":
                    pdf_text = perform_pypdf_ocr(pdf_content)
                    # Display the OCR output for PyPDF
                    output_placeholder.write("OCR Output (PyPDF):")
                    output_placeholder.write(pdf_text)

                elif option == "Nougat":
                    # Perform OCR using Nougat
                    # Display a loading message while OCR is in progress
                    output_placeholder.write("Performing OCR (Nougat)... Please wait.")
                    perform_nougat_ocr(io.BytesIO(pdf_content))

                else:
                    st.error("Please select a package first!")
            else:
                st.error(f"Failed to fetch the PDF from the URL. Status code: {response.status_code}")
        except Exception as e:
            st.error(f"An error occurred: {str(e)}")
    else:
        st.warning("Please enter a URL.")
```

Nougat-API

Further, since there were many complications while hosting the Nougat API locally, we had to create a tunnel using ngrok to host it somewhere else. The NGROK link has been generated on Google Colab document - Nougat_API_Hosting.ipynb. This document has been added to Github in the same format

- Code Snippet of NGROK functions

```
[ ] from pyngrok import ngrok, conf
    import getpass

[ ] print("Enter your auth token, which can be found on https://dashboard.ngrok.com/auth")
    conf.get_default().auth_token = "2WOifmbAGzodJjvenWeNDm5dRtC_YyUFkQJ94xD7seDr1S1p"

    Enter your auth token, which can be found on https://dashboard.ngrok.com/auth

[ ]

[ ] port = 8503
    public_url = ngrok.connect(port).public_url
    print(f"Tunnel 'http://127.0.0.1:{port}' to '{public_url}'")
    print(f"Visit the SwaggerAPI doc on '{public_url}/docs'")

    Tunnel 'http://127.0.0.1:8503' to 'https://4bda-35-240-219-27.ngrok.io'
    Visit the SwaggerAPI doc on 'https://4bda-35-240-219-27.ngrok.io/docs'
```

Guide on how to initiate it/ Tutorial:

- Step 1:
  - Once the github repo has been cloned, we need to go to the following file Assignment_01 > PartA > main.py
  - Open the integrated terminal for the same
    - Create the required virtual environment by writing
      - python -m venv .ocrenv
    - Activate the Virtual Environment
      - source .ocrenv/bin/activate
    - Then import the required modules from requirements.txt
      - pip install -r requirements.txt
  - This will set up the initial requirements
- Step 2:
  - Open the Nougat_API_Hosting.ipynb file and reset the runtime at first
  - Connect to T4 GPU4 and visit the site https://dashboard.ngrok.com/auth to generate an auth code
    - Save the auth code somewhere and add it in the file at the bottom where it is mentioned
    - ```
      #Enter your auth token, which can be found on https://dashboard.ngrok.com/auth
      #The auth token will go in below double quotes

      conf.get_default().auth_token = ""
      ```
  - Further run the entire document step by step and in the end we get an ngrok link where the local host 8503 has been hosted somewhere else
    - Example:
    - ```
      port = 8503
      public_url = ngrok.connect(port).public_url
      print(f"Tunnel 'http://127.0.0.1:{port}' to '{public_url}'")
      print(f"Visit the SwaggerAPI doc on '{public_url}/docs'")

      Tunnel 'http://127.0.0.1:8503' to 'https://4bda-35-240-219-27.ngrok.io'
      Visit the SwaggerAPI doc on 'https://4bda-35-240-219-27.ngrok.io/docs'
      ```
    - Visit the site and check if the Nougat API is working properly with the status code: 200
      - If it's not working, try deleting and creating a new runtime as well as reset the auth token for ngrok and redo step 2
- Step 3:
  - Once Step 1 and Step 2 are done, come back to the main.py file
  - On the terminal run the streamlit app with the following code:
    - streamlit run main.py
  - This will redirect the user to a streamlit app page

Note: The streamlit application has also been deployed on streamlit community and can be accessed using - https://bigdataia-fall2023-team8-assignment-01-partamain-soham-sovmhy.streamlit.app/

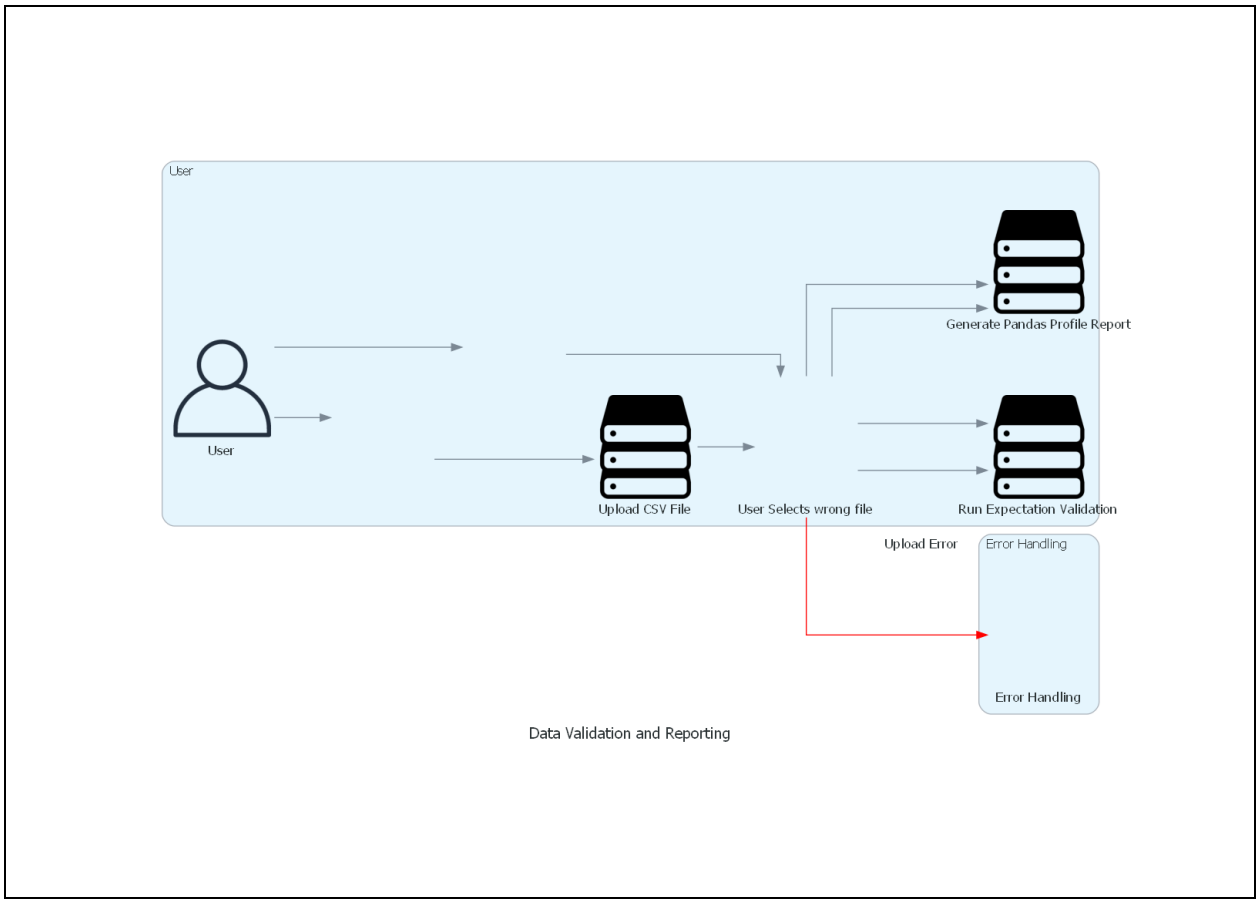
Part B

Instructions

Your team is tasked with building a tool to use the Freddie mac single family dataset [1]. You in the data
engineering group are asked to build a tool so you can evaluate the quality of the dataset and if it adheres to
the schema [2] that is published.

- Your tool will be built using streamlit
- A user would upload a csv/xls file and indicate if it is Origination/Monthly performance data
- You should use pandasprofiling to summarize the data and display to the end user
- You should run greatexpectations to ensure:
  - The data schema is correct
  - The data is valid
  - There is no missing data
  - Any other tests you can think of (2-5)

Design Architecture



Introduction

In Part B of our assignment, we have developed a Streamlit application to facilitate the interaction with data from the Freddie Mac dataset. This application enables users to upload CSV files from the Freddie Mac dataset, which can contain either "Origination Data" or "Monthly Performance Data." Our application performs data validation using Great Expectations, ensuring that the uploaded data meets specific criteria, such as credit score regex validation for origination data. Furthermore, we provide data profiling and reporting features, generating insights and statistics about the uploaded data. In cases where the uploaded file does not match expected column patterns for either data type, we employ error handling to alert users.

Streamlit

1. **File Upload:**Users can select a CSV file from their local machine and upload it to the application.
2. **Data Type Selection:** Users can specify whether the uploaded data is "Origination Data" or "Monthly Performance Data" using the provided radio button..
3. **Data Profiling and Reporting:** If the data type is recognized as "Origination Data," our application generates a comprehensive data profile report using Pandas Profiling. Users can initiate the report generation by clicking the "Generate Report: Pandas Profile" button.

```
if data_type is not None:
     st.subheader(f"This appears to be {data_type}.")
     if data_type == "Origination Data":
        #  checkpoint_name = "my_checkpoint_type2"
        #  results = context.run_checkpoint(checkpoint_name=checkpoint_name, batches=[{"batch_kwargs": {"dataset": ge_df}}])

         profile = ProfileReport(df, explorative=True)
         if st.button("Generate Report: Pandas Profile"):
            report = profile.to_file("report.html")
            st.success("Report generated successfully!")


         if "report.html" in locals():
             st.download_button("Download Report", "report.html")

     else:
        #  checkpoint_name = "my_checkpoint_type2"
        #  results = context.run_checkpoint(checkpoint_name=checkpoint_name, batches=[{"batch_kwargs": {"dataset": ge_df}}])

         profile = ProfileReport(df, explorative=True)
         if st.button("Generate Report: Pandas Profile"):
                report = profile.to_file("report.html")
                st.success("Report generated successfully!")

         if "report.html" in locals():
             st.download_button("Download Report", "report.html")
```

4. **Error Handling:** If the uploaded file does not conform to the expected column patterns for either data type, a warning message is displayed, prompting users to review and possibly correct their selection.
5. **Great Expectations Validation:** Upon clicking the "Run Expectation" button, the application uses Great Expectations to validate the uploaded data against predefined expectations

Guide on how to initiate it/ Tutorial:

- Once the github repo has been cloned, we need to go to the following file Assignment_01 > PartB > main.py
- Open the integrated terminal for the same
    - Create the required virtual environment by writing
        - python -m venv .reportenv
    - Activate the Virtual Environment
        - source .reportenv/bin/activate
    - Then import the required modules from requirements.txt
        - pip install -r requirements.txt
    - On the terminal run the streamlit app with the following code:
        - streamlit run main.py