EPiC Series in Computing

Volume 91, 2023, Pages 76–85

Proceedings of 38th International Conference on Computers and Their Applications

Crawling Based Data Collection and Data Pre-processing

Alexander Morales and Jiang Guo*
California State University, Los Angeles, USA
amora97@calstatela.edu, jguo@calstatela.edu

Abstract

This paper focuses on the design and architecture of an application that will programmatically pre-process data. This application aims to extract and provide clean airport terminal passenger throughput data within the United States. A different application will then use this data to forecast passenger throughput models via a friendly user experience webpage. The purpose of forecasting passenger throughput throughout the U.S. airport terminals is to improve the Transportation Security Administration (TSA) checkpoint operations. Such as increasing TSA personnel in security checkpoints when the forecast expects a high volume of passengers during the holiday season. On the other hand, decreasing the personnel workforce in other airport terminal checkpoints where they do not forecast a high passenger throughput. TSA seeks to improve its personnel scheduling using this forecasting model. In addition, the forecasting model will improve passenger satisfaction with non-excessive wait times at security checkpoints and does not jeopardize the safety of passengers with adequate security protocols.

1 Introduction

How can you provide a forecasting model of passenger throughput in each airport terminal within the United States without direct access to the data? Fortunately, the Transportation Security Administration, better known as "TSA" for frequent travelers, publishes passenger throughput data for all U.S. airports on this electronic reading room webpage. The category of interest in this reading room is the Airport category, and they upload new Portable Document Format (PDF) files weekly under the Freedom Act. As shown in Figure 1, it would be laborious to manually download every file available on the website and costly for third-party services to convert PDF files into Excel files. Currently, there is no free service that provides file conversions at no cost, with no limitations on the number of files that can be converted. Besides, what guarantees these converters won't impute or delete data during the conversion without notice? Why not build a convertor that will ensure that no data is corrupted?

* Contact author: Jiang Guo jguo@calstatela.edu

The proposal of an application that can be executed at any time and automatically download recently uploaded to the reading room webpage. Once the files are downloaded to a local directory of the application, it will attempt to extract the data from the file. The data will then be cleansed to ensure all incomplete data is imputed or removed depending on the severity. The application will export an Excel file that will provide the ability to cross-validate and visualize the pre-processing data results. Finally, it will persist the clean data into a MySQL database table for the forecasting application to periodically strengthen the forecasting models with new training data.

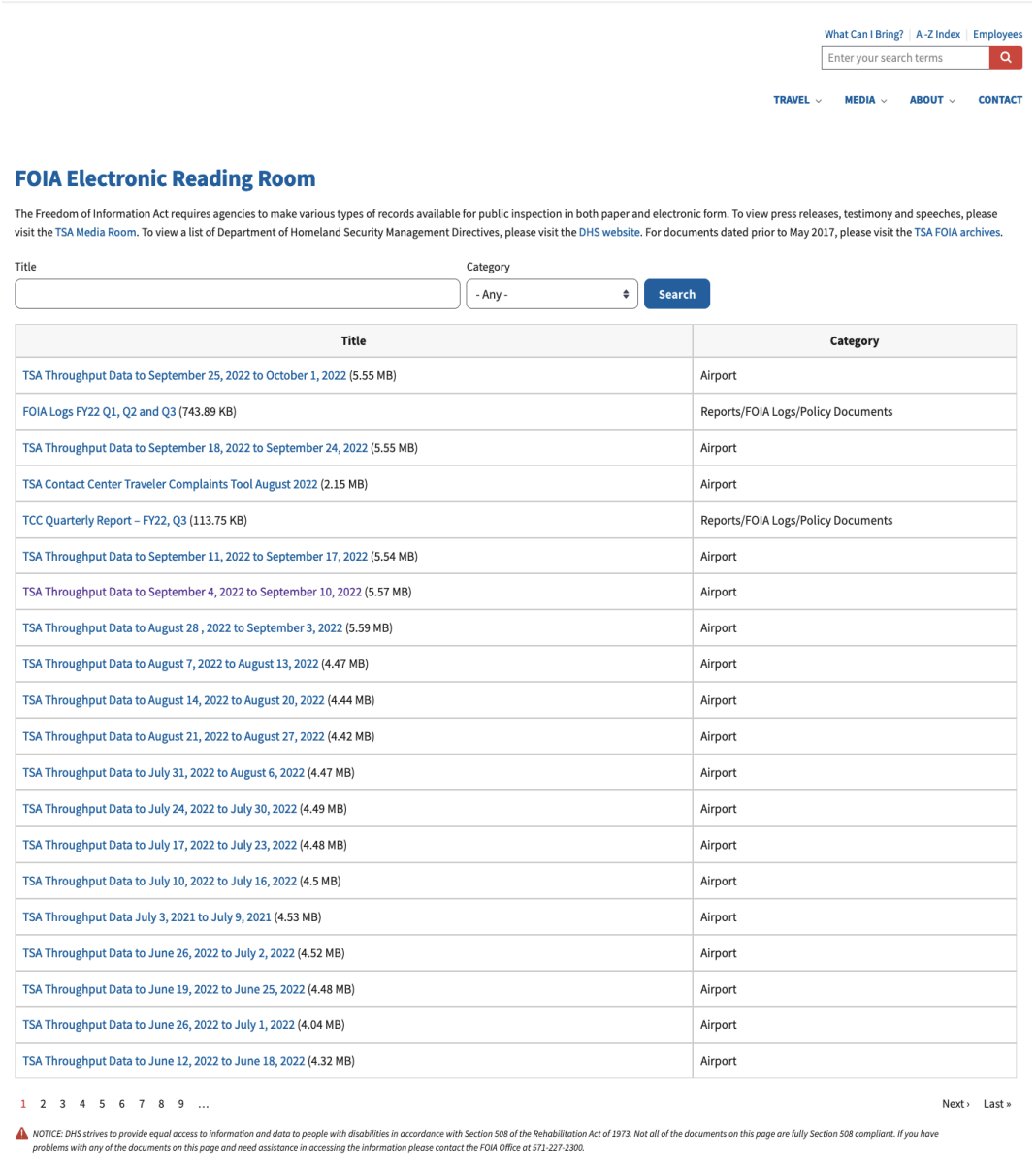

Figure 1. FOIA Electronic Reading Room

The requirements gathering phase in the software development lifecycle is the most critical aspect of a project. The project's features and capabilities need to be thoroughly discussed with stakeholders.

With clear communication of the project's deliverables, the application design must meet every stakeholder's criterion.

The deliverables are to provide the forecasting application with clean pre-processed airport terminal data extracted from the PDF files. The application will utilize web scraping and data science libraries to cleanse the airport terminal data from dirty data picked up during extraction. Then it will save the data into a MySQL database table for the upstream forecasting application to consume the data to improve the forecasting models.

The finalized requirements for the application were the following:

- Programmatically download and extract data from PDF files.
- Ensure missing data is imputed and dirty data is deleted from the dataset.
- Save cleansed data into the database and export data into an Excel file.

Data Analysis provides insight into the significance of the data columns the dataset will deliver to the forecasting application's models. Dirty data can be removed after the extraction providing the upstream application to only deal with clean, relevant data using data cleansing techniques. Figure 2 is an example of a PDF file from the Transportation Security Administration's reading room webpage. The file format will be the same for all files downloaded and read into the proposed application. It contains a table that can be created through Microsoft Excel or any other spreadsheet application. The data is written within the defined table, which contains merged cells in different columns. This can cause implications when using open-source libraries to read files into an application. All airport-relevant files found on the TSA reading room website are consistent with the table formatting.

With the table formatting implications identified during the data analysis. There will be specific capabilities a library will need to have to identify merged cells in a table. Otherwise, this will lead to missing data after the complete data extraction phase. There cannot be any missing data as the PDF does not contain missing data. If any data was to be removed from the dataset due to missing data values. It would dimmish the application's existence, as it needs to guarantee complete dataset delivery to the upstream application without data loss. Therefore, if a reliable open-sourced library can identify merged cells and organize the data into the proper column accordingly, it will be a contender for the application to use.

The data analysis was concluded with multiple libraries being used to extract data from the PDF files, such as tabula-py, camelot-py, and pdfplumber. Two of these libraries were promising at the beginning of the data analysis phase. However, over time tabula-py and camelot-py became inconsistent when extracting the data. This leads to overhead data manipulation to get the data into the proper columns in the Excel file. At last, pdfplumber was the most reliant library to use when it came to data extraction with merged cells. The only drawback was that the data had missing data values at random.

Date	Hour of Day	Airport		City	State	Checkpoint	Metrics	PMIS - Total Customer Throughput (Unadjusted)
4/28/2019	00:00	ANC	Ted Stevens Anchorage International	Anchorage	AK	South Checkpoint		154
		ATL	Hartsfield Atlanta International	Atlanta,	GA	Main Checkpoint		16
		BOS	Logan International	East Boston	MA	E2		95
		BQN	Rafael Hernandez	Aguadilla	PR	Rafael Hernandez Air		200
		BWI	Baltimore-Washington International	BWI Airport	MD	Pier C		5
						Pier D/E		0
		DEN	Denver International	Denver	CO	South		132
		DTW	Detroit Metro Wayne County	Detroit	MI	Blue-2		11
						Red 3		19
		EWR	Newark International	Newark	NJ	Terminal B/2		81
						Terminal C/1		25
		FAI	Fairbanks International	Fairbanks	AK	ASAA-FAI		0
		FAT	Fresno Air Terminal	Fresno	CA	FAT 01		18
		GSN	Saipan	Saipan	MP	GSN01		283
		GUM	Antonio B. Won Pat International	Tamuning	GU	GUM01		363
		HNL	Daniel K. Inouye International Airport	Honolulu	HI	HNL03		37
		IAD	Washington-Dulles International	Washington, DC	VA	Passenger Screening Area: PreCheck		158
		IAG	Niagara Falls International	Niagara Falls	NY	IAG-01		33
		IND	Indianapolis International	Indianapolis	IN	Checkpoint B		1
		JFK	John F. Kennedy International	Jamaica	NY	Terminal 1		328
						Terminal 2		4
						Terminal 4 Main		398
						Terminal 5		9
						Terminal 7		3
						Terminal 8		40
		LAS	McCarran International	Las Vegas	NV	Term 1 - AB		144
						Term 1 - D		69
						Terminal 3 - E Upper		198
		LAX	Los Angeles International	Los Angeles	CA	Suites		7
						TBIT Main Checkpoint		204
						Terminal 4 - Passenger		1
						Terminal 4A - Passenger		81
						Terminal 6 - Passenger		3
						Terminal 7 - Passenger		23
		MEM	Memphis International	Memphis	TN	Int'l Arrivals		0
		MIA	Miami International	Miami	FL	North 2		44
						South-J1		193
		MSP	Minneapolis-St. Paul International	St. Paul	MN	North CP		18
		ORD	Chicago-OHare International	Chicago	IL	10		140
						4B		21
						6A		34
		PBG	Plattsburg International	Plattsburgh	NY	Checkpoint 1		28

Figure 2. Initial data in PDF document

2 System Design

2.1 System Structure

Figure 3 shows a TSA personnel interacting with an application on a computer to visualize the forecasting model's prediction of passenger throughput. The left of the diagram shows the different components the data traversed through to be presentable to the user. Typically, these components used for data pre-processing are not client-facing. However, these components do the heavy lifting by providing clean data to enhance the forecasting models the user interacts with.

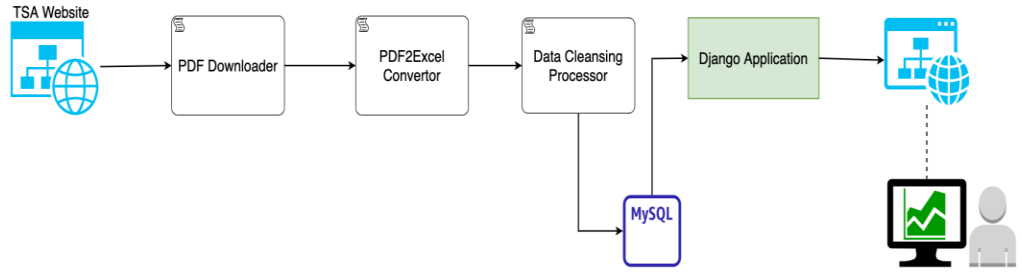

Figure 3. High level application architecture

2.2 Process Flowchart

With the high level of the application architecture drafted, you want to put together a process flowchart on how your application will handle different scenarios based on the requirements given initially. Figure 4 below shows the first version of the process flow chart. The end user must understand your application process flow, so they know the big picture of how they're getting their data. From an engineering perspective, the process flow will allow you to visualize any design inconsistencies or improvements that can eliminate unnecessary steps. This is key; unnecessary steps can impact the performance of the application. A second version of the process flow chart was drafted to address issues with the first draft. These changes did not affect the project deliverables but improved the reliability of the application and utilized the infrastructure, such as the MySQL database, to store any metadata related to the files.

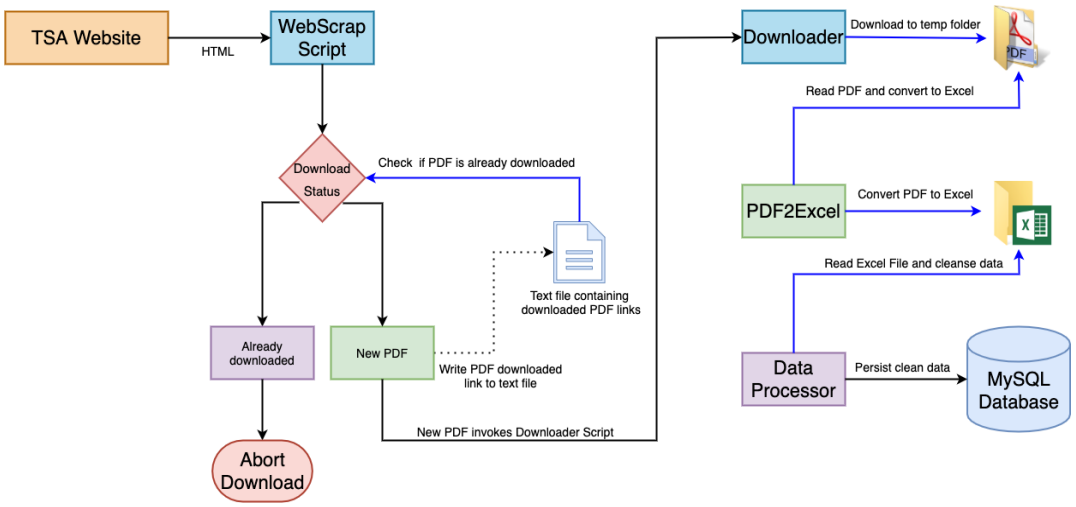

Figure 4. Application process flowchart version

3 Implementation

This section will guide you through the implementation of each component and the functionalities they provide to the application. Screenshots containing code snippets will be included throughout this section to clarify the application's implementation.

3.1 Database Operations

This class handles any communication between the application and MySQL database. It inherits a class that contains the credentials and attempts to establish a connection to the database, as shown in Figures 5 and 6 below. It provides a generic select method that allows any table within the connected database to be queried based on the SQL statement given as an argument. Some proprietary methods exist for inserting, searching, and updating records based on the database table. The purpose of this class is to condense the database access logic from the application into a single class.

```
class DatabaseConnection(object):
    @classmethod
    def db_connection(cls) -> mysql.connector.connection_cext.CMySQLConnection:
        kwargs = {
            'user': 'root',
            'password': '*********',
            'host': '127.0.0.1',
            'database': 'app_schema'
        }
        try:
            return mysql.connector.connect(**kwargs)
        except mysql.connector.Error as err:
            print("Error occurred when attempting to connect to database")
            print(err.msg)
```

Figure 5. Code snippet of database connection class

```
class DatabaseOperations(DatabaseConnection):

    def __init__(self):
        super().__init__()

    def select(self, query: str) -> List[dict]:
        if not query:
            return []

        connection = self.db_connection()
        cursor = connection.cursor(dictionary=True)
        cursor.execute(query)

        result = cursor.fetchall()

        cursor.close()
        connection.close()

        return result
```

Figure 6. Code snippet database operations class select method

3.2 WebScraper

When the executable is executed, an instance WebScraper will be instantiated and create an HTTP GET request to the TSA's reading room website. Once a response is received from the server hosting the webpage on the internet. The response content is passed into the BeautifulSoup4 library with the feature set to "html.parser". Figure 7 below contains a code snippet of the HTML that the library will receive in the response content. This was tedious during the development phase as the HTML file can be very large, and tags tend to be nested within containers. After some analysis of which tags contain the data that needs to be extracted, you will utilize the find method provided by the library. The "anchor" HTML tag has the download link, as shown in the figure 7 below on line 4, set for the HTML attribute "href." The links will then be filtered out if they do not contain specific keywords that correlate the links relevant to the airport throughput files. Finally, once the links are filtered, the domain will be appended for the Downloader class to download the files.

```
1  <td class="priority-low views-field views-field-title views-align-left" headers="view-title-table-column">
2      <div class="foia-reading-title">
3          <p><a class="foia-reading-link"
4                  href="/sites/default/files/foia-readingroom/tsa-throughput-may-9-2021-to-may-15-2021.pdf">TSA
5                  Throughput Data May 9, 2021 to May 15, 2021</a><span class="foia-reading-span"> <span
6                      class="foia-reading-filesize">(5.44 MB)</span></span></p>
7      </div>
8  </td>
9  <td class="foia-reading-category priority-low views-field views-field-field-foia-category views-align-left"
10     headers="view-field-foia-category-table-column">
11     <p>
12         Airport
13     </p>
14 </td>
15 <td class="priority-low views-field views-field-title views-align-left" headers="view-title-table-column">
16     <div class="foia-reading-title">
17         <p><a class="foia-reading-link"
18                 href="/sites/default/files/foia-readingroom/tsa-throughput-may-2-2021-to-may-8-2021.pdf">TSA
19                 Throughput Data May 2, 2021 to May 8, 2021</a><span class="foia-reading-span"> <span
20                     class="foia-reading-filesize">(5.39 MB)</span></span></p>
21     </div>
22 </td>
23 <td class="foia-reading-category priority-low views-field views-field-field-foia-category views-align-left"
24     headers="view-field-foia-category-table-column">
25     <p>
26         Airport
27     </p>
28 </td>
29 <td class="priority-low views-field views-field-title views-align-left" headers="view-title-table-column">
30     <div class="foia-reading-title">
31         <p><a class="foia-reading-link"
32                 href="/sites/default/files/foia-readingroom/tsa-contact-center-traveler-complaints-report-april.pdf">TSA
33                 Contact Center Traveler Complaints Tool April 2021</a><span class="foia-reading-span"> <span
34                     class="foia-reading-filesize">(15.3 MB)</span></span></p>
35     </div>
36 </td>
```

Figure 7. TSA reading room webpage HTML code snippet

3.3 Downloader

The Downloader class allows the application to download PDF files using the URL constructed by the WebScraper class discussed in the sub-section above. The class will call the endpoint using a get request method. Based on the response status code, the class will update the file status with either failed or success. If the file status is updated to success, the file is fetched. Now the class will download the file in chunks of 8 KB. The method download_pdf_files is the only public method the main program

will call. Its responsibility is to iterate through the list provided by the main program and execute the __downloader class method for each file's URL. Also, it checks if the target folder passed in as an argument exists. If not, it is creating the directory. Once the __downloader method completes execution, it will return a dictionary containing the status of the file. This dictionary will be appended to a list that will be returned to the main program.

4 PDF Data Extractor

This class exposes only a single public class method called extract_text. This method will be executed within main.py once the PDF file path is validated that it exists within the file system locally using the utility function Figure 8. The file path is then passed as an argument in the method. First, the method will create an empty instance of a data frame, which will accumulate all the data extracted from each page in the file. With the help of the pdfplumber library, the file is opened using the file_path argument. A check for an optional argument debugger_page_count is used for debugging purposes only. This is to limit the number of pages to be processed. If the debugger is not enabled, it will create a visual progress bar in the command line to display the progress of the data extraction. As it iterates through each page of the PDF, it will pass each page into a protected class method called __page_cleanser that will process each page individually. Each page will get its data extracted separately and append its extracted data into the empty data frame page_pdf. Data is concatenated to final_df before iterating to the next page (see Figure 9).

```
def pdf_file_locator(file_name: str, source_folder: str = 'temp') -> str:
    path = os.getcwd()
    file_path = "{0}/{1}/{2}".format(path, source_folder, file_name)
    if os.path.exists(file_path):
        print('PDF File Path Location: {}  '.format(file_path))
        return file_path

    return ""
```

Figure 8. PDF locator utility function

```
@classmethod
def extract_text(cls, file_path: str, debugger_page_count: int = None) -> pd.DataFrame:
    final_df = pd.DataFrame()
    with pdfplumber.open(file_path) as pdf:
        if debugger_page_count is not None:
            pages = [pdf.pages[debugger_page_count]]
        else:
            pages = pdf.pages
        for page in progress_bar(iterable=pages, prefix="Extracting", suffix="Completed"):
            page_df = cls.__page_cleanser(page=page)
            final_df = pd.concat([final_df, page_df], ignore_index=True)
            time.sleep(0)
        pdf.close()
    return final_df
```

Figure 9. PDFDataExtractor class extract_text method code snippet

4.1 Future Work

The future work for this application would be configuring the application to run on a server with a scheduler to run weekly to download newly uploaded data. On the other hand, it would be beneficial to retrofit the application to run on AWS lambda functions and utilize AWS EventBridge to schedule the lambdas to run once every week. This would be the most cost-effective solution. It is achievable as the system components are not coupled. Hence, the main program logic would be removed, and each component could be executed in separate lambda functions.

Acknowledgment

This research was performed under an appointment to the U.S. Department of Homeland Security (DHS) Science & Technology (S&T) Directorate Office of University Programs Summer Research Team Program for Minority Serving Institutions, administered by the Oak Ridge Institute for Science and Education (ORISE) through an interagency agreement between the U.S. Department of Energy (DOE) and DHS. ORISE is managed by ORAU under DOE contract number DE-SC0014664. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DHS, DOE or ORAU/ORISE.

References

[1] AWS. What is an IDE? Retrieved from https://aws.amazon.com/what-is/ide/

[2] Chojecki Przemek. (2019). Data science in 7 easy steps. Retrieved from https://towardsdatascience.com/data-science-in-7-easy-steps-ca80d063f175

[3] Daisy. Is excel enough for data analysis? Retrieved from https://datasciencenerd.com/is-excel-enough-for-data-analysis/

[4] Gazoni Eric, C. C. (2022). Openpyxl - A python library to read/write excel 2010 xlsx/xlsm files. Retrieved from https://openpyxl.readthedocs.io/en/stable/

[5] Jet Brains. Scientific tools. Retrieved from https://www.jetbrains.com/pycharm/features/scientific_tools.html

[6] Mozilla Developer Network. (2022). HTML: HyperText markup language. Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTML

[7] Oracle Corporation. (2022a). Chapter 1 introduction to MySQL connector/python. Retrieved from https://dev.mysql.com/doc/connector-python/en/connector-pythonintroduction.html

[8] Oracle Corporation. (2022b). What is MySQL? Retrieved from https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

[9] Pandas package overview. (2022). Retrieved from https://pandas.pydata.org/docs/getting_started/overview.html

[10] Python Software Foundation. Creation of virtual environments. Retrieved from

https://docs.python.org/3/library/venv.html

[11] Reitz Kenneth. (2022). Requests: HTTP for humans. Retrieved from https://requests.readthedocs.io/en/latest/

[12] Richardson Leonard. (2022). Beautiful soup documentation. Retrieved from https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[13] Singer-Vine Jeremy. (2022). Pdfplumber. Retrieved from https://pypi.org/project/pdfplumber/

[14] Team Nuggets. (2018). Why data scientists love python. Retrieved from https://www.cbtnuggets.com/blog/technology/data/why-data-scientists-love-python