PharmaSUG China 2024 - Paper CC-10008

Automated PDF Data Extraction Tool - An Interactive Platform with Python Dash

You Lyu, Johnson & Johnson

ABSTRACT

PDF files are widely used for sharing information in clinical trials, including protocols, clinical study reports, etc. However, extracting data from those can be challenging due to their varying layouts and complex formatting. In this paper, we present an automated tool for extracting structured data from PDF documents. It streamlines the extraction process, enhances productivity, and enables organizations to leverage the wealth of structured data embedded within PDF documents for further analysis and decision-making. Furthermore, this tool has its user interface established in Dash, a powerful and intuitive framework for building interactive web applications in Python. With the increasing demand for data-driven and user-friendly web interfaces, Dash provides a seamless solution for developing robust and dynamic applications without the need for extensive front-end development skills.

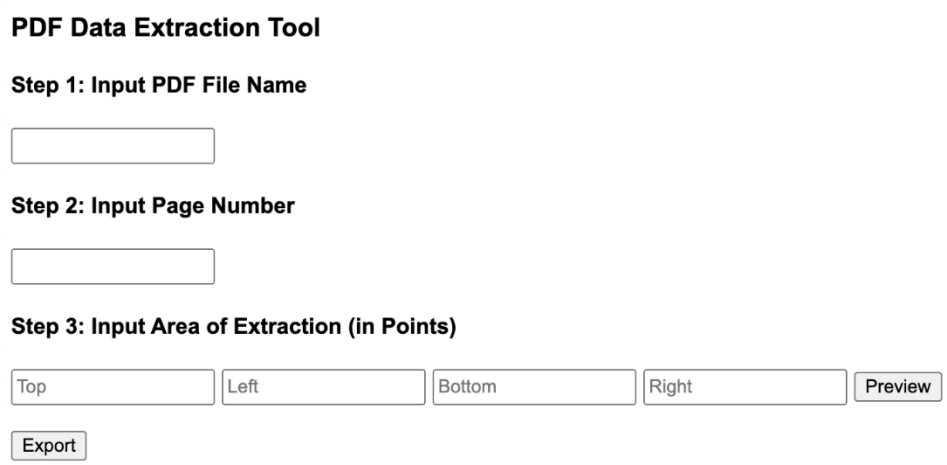
INTRODUCTION

It is inevitable that PDF files have become a ubiquitous format for storing and sharing information in clinical trials. And we constantly face the need of extracting tabular data from PDF files but also encounter difficulty since the nature of the tabular data is unstructured and has varying layouts and complex formatting. One intuitive and straightforward method is to extract data manually. However, it is highly time consuming and not efficient enough to be a good practice in routine work. Therefore, the need of developing an automated PDF data extraction tool to reduce the time and effort required while maintaining data integrity is identified. Meanwhile, I would like to utilize the rich resource of data exploration and manipulation techniques in Python libraries.

As we know, Python has not been predominantly adopted in pharmaceutical industry and remains an innovative exploratory programming language for most programmers. It is equally important to develop a user-friendly interface that users do not need extensive front-end development skills to access. Dash integrates seamlessly with various data manipulation and analysis Python libraries, such as Pandas and NumPy, offering extensive possibilities for data processing, visualization, and real-time updates. The ability to connect to databases, APIs, and other data sources enhances the application's dynamic capabilities, allowing for real-time data streaming and interactive insights. Therefore, this paper will not only focus on how to automate the tabular data retrieval process from PDF files but also how to develop an intuitive and accessible user interface for this tool.

LAYOUT OF DASH APPLICATION

The layout design of Dash application is quite intuitive (Display 1). Users need to follow three steps to provide inputs: the PDF file name of extraction (including suffix), a single page number (multi-page extraction is not supported for now), and input area of extraction (in points). After the input boxes have been filled in, users need to click the ‘Preview’ button to execute the back-end code. If inputting properly, users can review the extracted data on the web application. Moreover, there is another option that users can download a csv file of extracted data by clicking ‘Export’ button.



Display 1. Layout of Dash Application

The input boxes are necessary in helping the algorithm to locate the file, the page and most importantly, the area of the tabular data. It is more accurate for the algorithm to identify the table as user narrows the area of extraction. This 'Preview' tab is designed for helping user to identify whether the targeted table is extracted. If not, user needs to check if those input parameters are properly entered before extracted data is used for further analysis. Additionally, the 'Export' tab helps to export data into a csv format since csv file can be easily read in commonly used software in industry, including SAS, R, Python and can be used for further analysis.

COMPONENTS OF DASH APPLICATION

The Dash application consists of two component parts from designer perspective. The first part is Dash core components (abbreviated as dcc), including the input boxes in our application. It is shown as dcc.Input() in the code. Users need to fill in six input boxes: the PDF file name of extraction (including suffix), a single page number (multi-page extraction is not supported for now), and input area of extraction (in points). Then the type of input boxes needs to be specified. For example, text is expected for pdf file name input box; therefore, users specify text as its input type. Moreover, there are a lot of other functional core components, including dropdown menu, checklist, etc. Those components are currently not applied in current PDF data extraction application but are beneficial in different application design scenarios.

The other Dash component is HTML component where users specify buttons or headers information, etc. One advantage of Dash application is that it can be highly customized for any text font, color, location of input boxes, etc. It allows users to design visually appealing interfaces, choose specific color schemes, and add branding elements to match organization's style and requirements.

```
app.layout = html.Div([
html.H3('PDF Data Extraction Tool', style={'font-family':'Arial'}),
html.H4('Step 1: Input PDF File Name'),
dcc.Input(id='filename',type='text'),
html.H4('Step 2: Input Page Number'),
dcc.Input(id='page_num',type='number'),
html.H4('Step 3: Input Page Number'),
dcc.Input(id='top',type='number',placeholder='Top'),
dcc.Input(id='left',type='number',placeholder='Left'),
dcc.Input(id='bottom',type='number',placeholder='Bottom'),
dcc.Input(id='right',type='number',placeholder='Right'),
```

```
html.Button('Preview',id='apply-button',n_clicks=0),
html.Br(),
pdf_table
])
```

TECHNIQUES OF DATA EXTRACTION

Tabula-py is a Python library that provides an interface to Tabula, a Java-based tool for extracting tables from PDF documents. The library includes functions that used to extract tabular data from PDF files.

- PDF Parsing: Tabula-py uses a PDF parsing engine to analyze the PDF file's structure, identify tables, and extract tabular data. It analyzes the page layouts, text positioning, and other metadata within the PDF file to identify table boundaries.
- Area Extraction: Once the tables are detected, Tabula-py extracts the specific regions within the PDF that contain the tabular data. It uses the identified table boundaries to define the extraction areas and isolate the relevant content.
- Data Extraction: Tabula-py performs OCR (Optical Character Recognition) on the extracted table regions to recognize and extract the textual data contained within them. This involves applying image processing techniques to convert the scanned or image-based text in the PDF into machine-readable data.
- Post-processing: After the data extraction, Tabula-py applies additional post-processing techniques to improve the accuracy and cleanliness of the extracted data. It handles issues such as merged cells, incomplete or misaligned rows, and other common table irregularities.

Overall, Tabula-py provides an accessible and convenient way to extract tabular data from PDF files. By combining PDF parsing, table detection, area extraction, OCR, and post-processing techniques, Tabula-py enables users to easily convert PDF tables into structured data that can be further analyzed or integrated into other applications or workflows.

CHALLENGES OF INTEGRATING DATA EXTRACTION AND DASH APPLICATION

Ensuring seamless integration of data extraction tool with other components of a Dash application can present specific challenges. One challenge involves synchronizing the data extraction process with the overall workflow of the application. Timing the extraction to occur at the right moment on user request, in our case, requires careful coordination with other components and events within the application.

Additionally, integrating the extracted data seamlessly with other components of the Dash application, such as data visualization, may require proper data formatting, transformation, or integration with other data sources. Ensuring data integrity, consistency, and compatibility with downstream processes can be a complex task that requires careful design and consideration. Successfully addressing these challenges necessitates a thorough understanding of the data extraction tool, coordination with other components, and comprehensive testing to validate the integration and ensure a seamless user experience throughout the data extraction process within the Dash application.

CONCLUSION

In this paper, we have presented an automated PDF data extraction tool with Python Dash, providing a solution for efficiently extracting structured data from PDF documents. Leveraging the power of Dash, our tool offers a user-friendly interface and customizable features that streamline the extraction process, enhance productivity, and enable organizations to unlock valuable insights hidden within PDF data.

By seamlessly integrating with Python's data manipulation libraries, the tool enables efficient data processing, analysis, and visualization. The ability to connect to databases, APIs, and other data sources ensures real-time updates and dynamic data streaming within Dash applications. The customizable user interface and interactivity features enhance user experiences and enable interactive exploration of extracted data.

Extensive testing and evaluation have demonstrated the effectiveness and efficiency of our automated PDF data extraction tool. Compared to manual extraction methods, our tool significantly reduces extraction time, minimizes errors, and improves overall data accuracy. It empowers users to extract valuable insights from PDF data rapidly and reliably.

REFERENCES

Hossain, S., Calloway, C., Lippa, D., Niederhut, D., & Shupe, D. 2019, July. Visualization of Bioinformatics Data with Dash Bio. In *SciPy*, 126-133.

Chawla, A., Gupta, A., & Shushrutha, K. S. 2022. Intelligent Information Retrieval: Techniques for Character Recognition and Structured Data Extraction.

DASH Documentation & User Guide | Plotly. https://dash.plotly.com/.

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to my colleagues in C&SP for their continued support. I would also like to thank Tina Zhai for identifying the need of developing this application and providing testing documents.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

You Lyu
Johnson & Johnson
ylv13@its.jnj.com