CHALLENGES YOU WILL FACE WHEN PARSING PDFS WITH PYTHON – HOW TO PARSE PDFS WITH PYTHON

Challenges You Will Face When Parsing PDFs With Python – How To Parse PDFs With Python

research@theseattledataguy.com November 19, 2024 data engineering 0

Scraping data from PDFs is a right of passage if you work in data. Someone somewhere always needs help getting invoices parsed, contracts read through, or dozens of other use cases. Most of us will turn to Python and our trusty list of Python libraries and start plugging away.

parse accurately, even with OCR. Now, in a prior article, we discussed how you can parse PDFs, and in this article, I wanted to discuss some of the challenges you'll face when parsing PDFs. This ranges from the issues when developing custom pipelines to other challenges.

So, let's talk about the challenges you'll face when parsing PDFs with Python.

Challenges When Parsing PDFs

Here are some of the key issues we've run into when parsing PDFs.

Time Consuming

The initial challenge most developers might run into is that it is time-consuming to build a proper system to scrape PDFs. Don't get me wrong. Writing the initial Python scripts to parse PDFs is pretty easy.

But as you develop a more robust system, one that likely can handle large amounts of data, you'll likely run into various issues. This ranges from data quality issues to performance issues, etc. So, building out this proper system becomes very time-consuming. You're generally building a data pipeline, meaning it needs to be managed like one and not just a bunch of Python notebooks duct-taped together.

This also means that the engineering time can really add up. As soon as you feel like you've created a method to handle parsing the PDFs, you run into a gotcha. I recall going through several Facebook pipelines that were thousands of lines to properly parse, map and transform data from PDFs. So it can get quite expensive.

Unstructured and Non-Tabular Data

Perhaps the most obvious issue is the fact that the data is unstructured. Unlike data stored in spreadsheets or databases, PDFs are designed for display rather than structured data storage. This lack of standard formatting makes it difficult to extract meaningful data, especially from text-heavy PDFs where there's no clear separation of fields or records. Not to mention that PDFs often combine various types of content, like headers, footers, tables, images, and multi-paragraph text, without a consistent structure. This means that data points might appear anywhere on the page, often requiring manual identification and extraction rules.

Layout Complexity

One less thought of the issue is the layout. Now, some of this has to do with a point similar to the above. Sometimes, the chart you're looking to parse or the contract details you want to extract are on a different page or in a different section.

Don't believe me, you've likely had schema issues with structured files. You don't think it'll be worse with unstructured PDFs? So here are some of those issues you'll face in terms of layout.

- **Multi-Column Layouts:** PDFs with multi-column layouts, common in research papers and newspapers, are especially difficult to parse because the columns need to be identified and extracted in the correct order. Parsing tools often read text line-by-line, which doesn't naturally account for multiple columns.
- **Nested Tables and Nested Elements:** Documents with complex tables, such as nested tables (tables within tables), require extra effort to parse accurately. Standard table-extraction tools may fail to recognize nested structures, leading to incomplete or incorrect data extraction.
- **Dynamic Page Elements**: Many PDFs include dynamic elements, such as header and footer repetition across pages. Parsing tools often don't differentiate between main content and repetitive page elements, so headers, footers, and page numbers can interfere with the extraction of actual content. Separating these elements requires either page-specific handling or custom filtering.

Image-Embedded Text (OCR Requirement)

PDFs created from scanned images contain no raw text data. Instead, all text is embedded within images, making Optical Character Recognition (OCR) essential for text extraction. This can introduce further complications, as OCR often struggles with low-resolution images, non-standard fonts, and complex layouts. Let's be clear: even with OCR tools like Tesseract, accuracy isn't guaranteed, especially when dealing with complex fonts or low-quality scans. It's not a magic bullet. It'll do its best to try to decipher the text. Sometimes, it won't get it right. You want to make sure you build this into your system and have data quality checks that can detect some of these issues.

Accurate PDF Parsing

Another challenges you'll likely have to tackle is the accurate parsing of lengthy PDFs that contain similar information presented in multiple formats. For instance, in SEC filings, you

and actionable.

For example, imagine an SEC filing for a large multinational corporation. You might find three separate tables showing revenue figures for adjusted revenue, segment revenue and consolidated revenue. While these tables look similar, each serves a different analytical purpose. Parsing the PDF accurately involves recognizing which table is which, interpreting the adjustments made, and ensuring that the final analysis reflects the intended financial perspective.

Large Files and Performance Issues

Parsing large PDFs can be memory-intensive, especially when handling multi-page documents or high-resolution scans. Standard libraries may struggle to load large files into memory, leading to performance bottlenecks or even crashes. In turn, this leads to slow processing, not to mention needing to manage large chunks of data in memory for multi-page PDFs. This means you'll need to create more robust systems that can actually manage the larger data sets and perhaps parallelize some of the workloads.

Tools And Libraries Made To Make Parsing PDFs Easier

Now, we've already talked about several Python Libraries, but let's dig into those a little more deeply and offer a few tools you can use to parse PDFs.

Pytesseract – is an OCR (Optical Character Recognition) tool, which means it's used to extract text from images or scanned documents. If you have a document or image that is not in a text-based format, such as a PNG, JPEG or scanned PDF, Pytesseract will likely be the right tool.

PyPDF2 – is designed to handle text extraction, manipulation, and merging/splitting of text-based PDF files, not scanned images. If the PDF is not a scanned document but was generated electronically, PyPDF2 can directly extract this text.

pdfplumber – is designed for extracting structured data from PDFs, including text, tables, and positional information. It excels at handling PDFs with complex layouts, making it ideal for extracting tabular data and analyzing precise document structures. Its ability to extract tables and text accurately makes it a go-to tool for processing financial statements, invoices, and reports.

Roe.ai – If you're not comfortable with Python or you just want to be able to run large queries over your PDFs, you can use tools like Roe AI. Roe has built-in data connectors to

Tips for Successful PDF Parsing

If you are looking to parse PDFs with Python, then here are a few quick scripts and tips you can use to get started.

Use Multiple Libraries Together

No single library can handle all aspects of PDF parsing well, so combining libraries can help tackle different challenges. For example: Use pdfplumber for tables and PyPDF2 for text extraction if a document contains a mix of structured and unstructured content.

For scanned PDFs, combine `pdf2image` and `Tesseract OCR` to handle image-based text extraction.

Example: Extract text and table data from a PDF with both text and tables.

```python
import pdfplumber
import PyPDF2

# Open the PDF file with PyPDF2

pdf_file = 'example.pdf'
pdf_reader = PyPDF2.PdfReader(pdf_file)

# Extract text with PyPDF2
full_text = ""

for page in pdf_reader.pages:
    full_text += page.extract_text()  # Extract text from each page

print("Extracted Text:", full_text)

# Extract tables with pdfplumber
with pdfplumber.open(pdf_file) as pdf:
    for page in pdf.pages:
        tables = page.extract_tables()  # Extract tables
        for table in tables:
            print("Extracted Table:", table)  # Print each table
```

extract_table_example.py hosted with ❤ by GitHub view raw

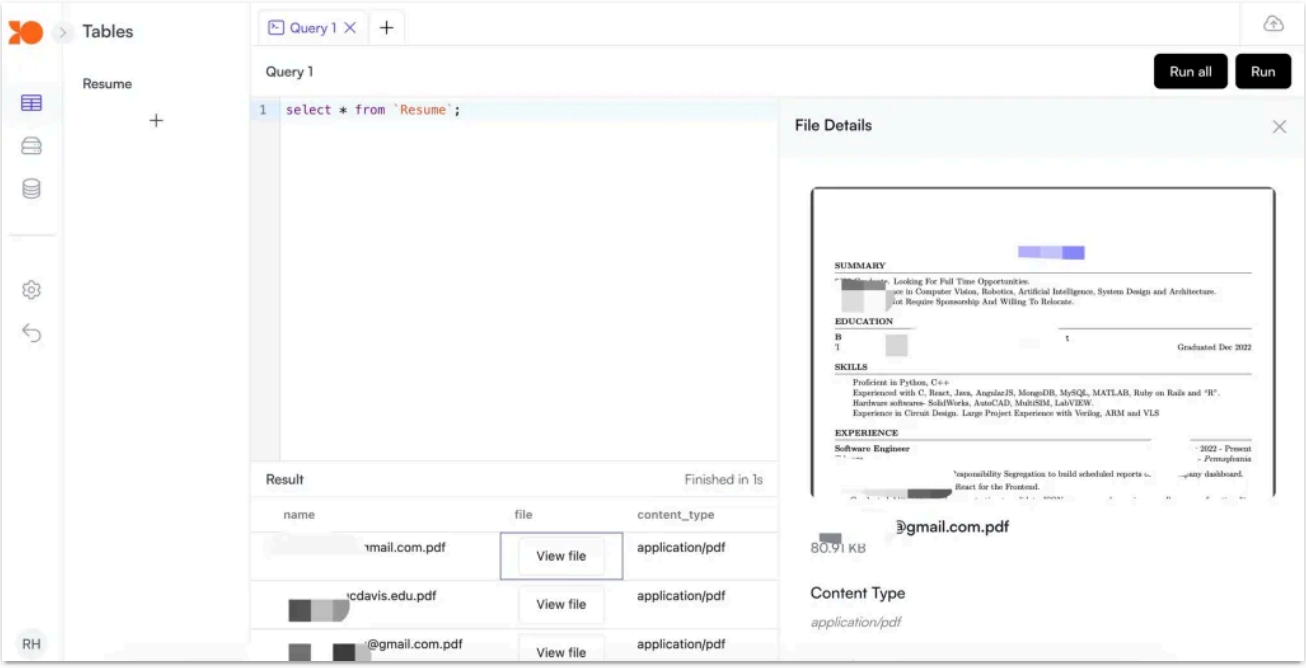
It's hard to ignore that LLMs are playing a role in making many tedious tasks easier. Whether you're using ChatGPT, Claude or another LLM. Many of them provide some option for parsing PDFs.

With vision-enabled LLMs like ChatGPT with image understanding, you can directly analyze PDFs without needing external tools to extract content. This capability allows the model to "see" the structure and layout of the document, enabling tasks such as direct parsing, contextual summarization and even better accuracy. But they aren't the only option.

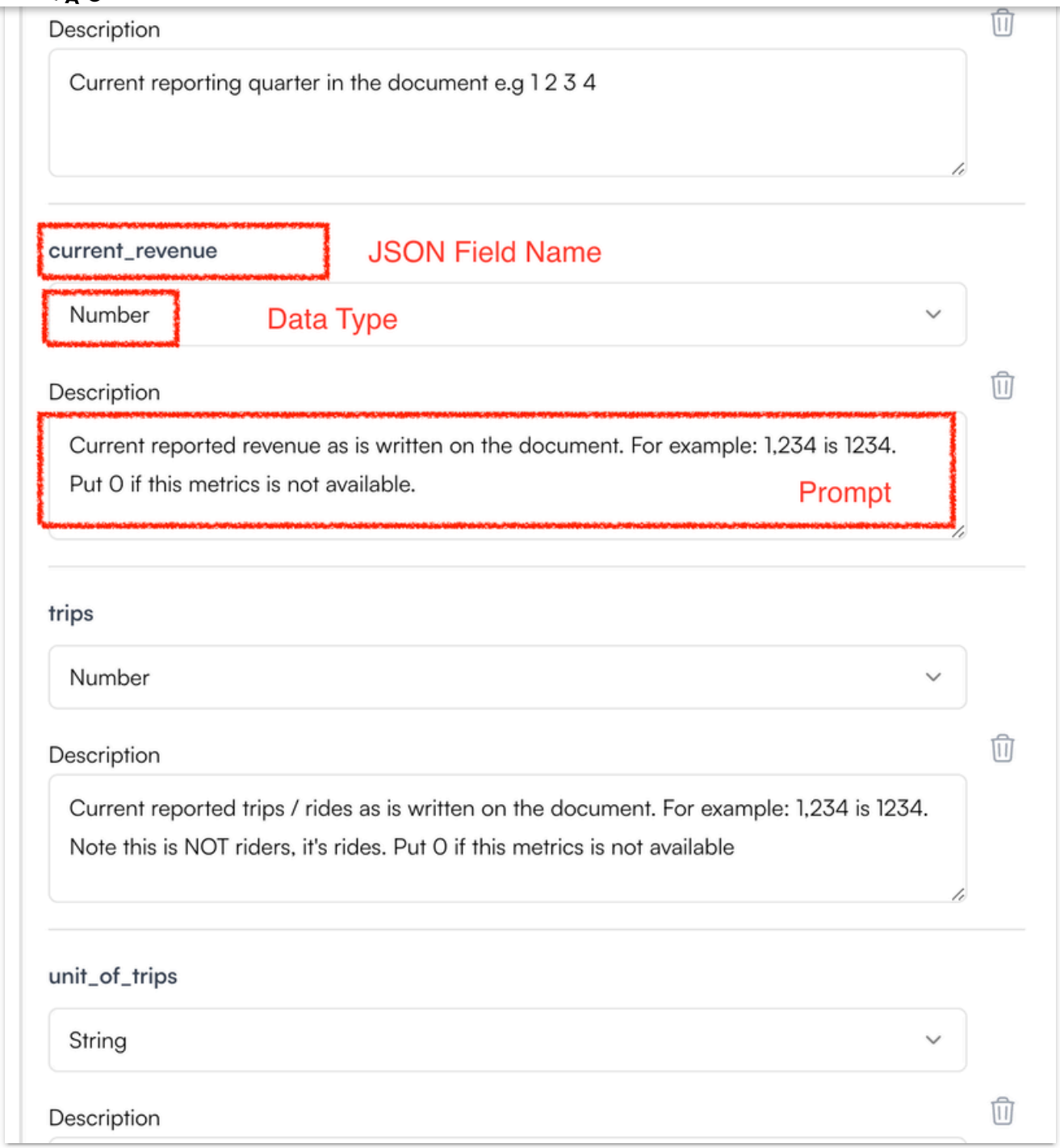
Use Tools Like Roe.AI To Help Reduce Time To Value

Besides writing custom code and using an LLM, you can use tools such as Roe.ai to make handling PDFs far simpler. For some teams, you might be too busy to build an entire PDF parser and still for others it might require more technical skill than you currently have. Still others might be looking for a way to more accurately extract data from their PDFs which Roe can help do.

For example, you could use Roe to help parse and analyze thousands of resumes or SEC files. This could be to perform some form of financial analysis or quickly going through resumes.



Instead of code, you can use agents like the ones in the image below to parse specific data points out of PDFs. Ensuring both faster and more accurate parsing.



This would allow your team to focus more on the analysis and less on the parsing of PDFs. Meaning you can drive value faster!

Making Parsing PDFs Even Easier

Parsing PDFs is a common need for large enterprises and start-ups. However, it continues to provide similar challenges, from having to build systems that can handle varying types

If you're looking to dig deeper into parsing PDFs for analytics, then I'd love to chat.

Disclosure: Seattle Data Guy does have a stake in Roe.AI

Also! Don't forget to check the articles below.

Common Pitfalls of Data Analytics Projects

How to cut exact scoring moments from Euro 2024 videos with SQL

9 Habits Of Effective Data Managers – Running A Data Team

The Data Engineer's Guide to ETL Alternatives

Build A Data Stack That Lasts – How To Ensure Your Data Infrastructure Is Maintainable

Explaining Data Lakes, Data Lake Houses, Table Formats and Catalogs

How To Modernize Your Data Strategy And Infrastructure For 2025

---

data engineering Data Science

« PREVIOUS: FROM IC TO DATA LEADER: KEY STRATEGIES FOR MANAGING AND GROWING DATA TEAMS

NEXT: PREPARING YOUR DATA INFRASTRUCTURE FOR 2025: LESSONS FROM THE PAST, STRATEGIES FOR THE FUTURE »

Learn About End-To-End Data Flows (Data Engineering, MLOps, and Data Science)

Over 107,000 subscribers

Type your | Subscribe

By subscribing you agree to Substack's Terms of Use, our Privacy Policy and our Information collection notice