Automating Audit Reconciliations: A Use Case

Jeroen Bellinga
PricewaterhouseCoopers Accountants N.V.
jeroen.bellinga@pwc.com

Tjibbe Bosman*
University of Amsterdam, Foundation for Auditing Research
t.bosman@uva.nl

Seyit Hocuk
Stichting CentERdata, Tilburg University
s.hocuk@tilburguniversity.edu

Wim H.P. Janssen
University of Amsterdam, Foundation for Auditing Research
w.h.p.janssen@uva.nl

Alaa Khzam
PricewaterhouseCoopers Accountants N.V.
alaa.khzam@pwc.com

*Corresponding Author

We thank, Olof Bik, Arjan Brouwer, and Joris Roosen for their support, comments, and encouragement for this project and paper, all remaining errors are our own. The authors thank the Foundation for Auditing Research (FAR) for its research grant 2019F02. The views expressed in this document are those of the authors and not necessarily those of the FAR.

Jeroen Bellinga, PricewaterhouseCoopers Accountants N.V., Groningen, The Netherlands, Tjibbe Bosman, University of Amsterdam, Amsterdam Business School, Accounting Section, The Netherlands, Foundation for Auditing Research, Breukelen, The Netherlands, Seyit Höcük, Stichting CentERdata, Tilburg, The Netherlands, Tilburg University, Tilburg, The Netherlands, Wim Janssen, University of Amsterdam, Amsterdam Business School, Accounting Section, The Netherlands, Foundation for Auditing Research, Breukelen, The Netherlands, Alaa Khzam, PricewaterhouseCoopers Accountants N.V., Groningen, The Netherlands.

Automating Audit Reconciliations: A Use Case

ABSTRACT

A key audit procedure is the reconciliation of audit evidence to the audit subject matter, which is often in a Portable Document Format (PDF). These reconciliations are a recurring task for every new version of the audit subject matter. Audit firms tend to “offshore” simple and repetitive audit tasks to shared service centers. Offshoring however comes at the expense of coordination costs, delays in the process, and challenges regarding the liability risk to the auditor. This paper presents an open-source algorithm to extract data from (draft) annual reports (PDF files) using Python to automate, rather than outsource, these repetitive financial statement reconciliations. The use case we present resulted in significant time saving for the audit of Dutch investment funds. The same technique can potentially be applied by academics to automate their data gathering activities from filed financial statements in PDF documents.

Keywords: Automation Audit Tasks; Data Extraction; Python; Reconciliation; Investment Funds; Academic Practitioner Collaboration

JEL Classifications: M42, G23, G29

Data Availability: The algorithm this paper discussed is open-source and publicly available.

I. INTRODUCTION

Audit subject matters, including financial statements, are generally prepared as a multipage PDF documents. Ensuring that what is in the PDF reconciles to the audit evidence is an important but somewhat tedious task for the audit team. Audit firms tend to offshore simple and repetitive audit tasks to shared service centers (Daugherty, Dickins, and Fennema 2012). Outsourcing however comes at the expense of coordination costs, delays in the process (Hanes 2013), and challenges regarding the liability risk to the auditor (Lyubimov, Arnold, and Sutton 2013). This is particularly the case for the audits of clients that issue large sets of financial statements that follow a similar template or structure, for example asset management companies that manage many investment funds. Annual reports in the asset management industry are generally issued per investment fund, which require a separate reperformable documented reconciliation between the financial statements and the supporting audit evidence that is the basis for the audit opinion. This procedure needs to be repeated for every new version of the audit subject matter. Typically, the financial statements go through multiple revisions and required reconciliations before the audit subject matter is finalized. Following this, the audit team initiates the offshored standardized financial statement quality check procedures for every version of the financial statements, where a remote team checks the arithmetic and internal consistency within the final draft of the PDF document, as well as a tie-out procedure which reconciled the figures in the PDF document to the underlying audit evidence.

By applying several simple techniques in Python this process can be automated and accelerated, which allows the audit team to save (substantial) time and budget. Concerns with applying data science techniques in the auditing process usually encompass the apprehension of (unintended online) sharing of non-disclosed unaudited data, the assurance of operations and calculations performed by automated models, and whether the technique is reperformable. Finally, a lack of experience and resistance to new techniques at the auditee or audit team might be a hurdle. We will discuss how we coped with these issues.

The outline of this paper is as follows, first we give a brief introduction of our academic practitioner collaboration followed by a short description of the used programming language Python. Second, we outline the challenges that audit practitioners generally face when applying new techniques in the audit. Third, we discuss several possible approaches and explain our choices. Finally, we provide the algorithm that we developed, so that audit practitioners and academics can apply it and adjust it to their own situation and needs.

II. OUR ACADEMIC AND PRACTITIONER COLLABORTATION

In 2015 the Dutch auditing profession founded the Foundation for Auditing Research (FAR). The aim of FAR is to enhance the knowledge base by conducting academic research into the key drivers of what makes a good audit today and academically inform audit practices and their continuous improvement effort. For this purpose, FAR actively collaborates with the largest auditing firms in the Netherlands, to obtain archival data, invite participants to complete surveys and promote research informed policy making. Proprietary data points are collected and anonymized by the audit firms, where publicly available data is prepared by research associates from FAR.

The manual data collection process for research purposes is a complex, time-intensive, and expensive process, which we started to standardize and automate where possible. To apply data-science techniques for the FAR data collection efforts, and to consult us on the matter at hand, we engaged with the data-science department of Stichting CentERdata[1]. Out of these automatization efforts and several conversations with the audit firms[2], we considered the application of similar techniques from the data collection process to audits engagements in daily practice. We arrived at the use case for the audits of investment funds through discussion of the techniques with experienced auditors from a Big 4 firm that participate in the FAR collaboration.

III. WHAT IS PYTHON?

Python is an open-source general-purpose programming language (van Rossum, 1995). Due to its design the programming code is relatively easy to read, learn, and understand. This makes Python especially useful for auditing practitioners without prior programming or data science experience. Python is a widely accepted programming language and is regarded as very stable. According to many sources (Stack Overflow 2020, Eastwood 2020), Python is the most popular programming language currently in use, and still growing. There are many (free) resources to learn how to use Python (Python Software Foundation, 2020). The application of Python within a web-based interactive computational environment such as Jupyter Notebooks[3] offers the opportunity to make code more readable, cluster the code by cells, and add clear notes and documentation directly to the code. Jupyter Notebook Files (ipynb-extension) can provide the auditor with a reperformance audit trail of the procedures performed. Python comes with many packages, which are basically standard pieces of code that can be installed and applied when needed. The packages we incorporate in this work include *Fitz, re, csv* and *os*.

IV. CHALLENGES WE EXPERIENCED WHEN APPLYING NEW TECHNIQUES IN THE AUDIT PROCESS

Unaudited and unpublished data is subject to client confidentiality and therefore cannot be shared with third parties by the auditor unless the client agrees on this. We therefore only programmed based on published and audited versions of the financial statements, these were already publicly available and therefore posed no risk. Furthermore, we reviewed the code and

[1] Stichting CentERdata is a research institute associated with Tilburg University, which provides a wide range of research support services

[2] All authors are subject to non-disclosure agreements relating to the specific audit client for which we first developed this script.

[3] There are also many other integrated development environments (IDEs) for Python, the most popular ones include PyCharm, Atom, Spyder, IDLE, and Visual Studio Code.

the documentation of the packages we used to ensure that no data is shared outside of the local machine where the script is performed on. Since we only use standard packages, we rely on the Python documentation and release notes relating to the effectiveness of the operations.

Another issue is the evaluation of the performance of automated tools. Because we applied our script to published financial statements of which the audit has been completed, we were able to compare the acquired results to the figures in the finalized, error-free version of the financial statements. Moreover, we had the benefit of having team members with prior coding experience from working as researchers at various universities, now working for the audit firms. This made the adaptation to applying new techniques in auditing much better. Performance evaluation was an integral part of the development process.

V. TECHNIQUES APPLIED

Our goal is to make a reliable and structured machine-readable export of all the financial information in of the draft financial statements (PDF audit subject matter) to automate reconciliations. This export is most useful for the audit team in the form of a spreadsheet, as the outcome of most accounting and consolidation systems is also provided in spreadsheet format. Spreadsheets allow for easy and direct interactions with the data. In this section we discuss available standard software first, followed by Optical Character Recognition (OCR), the Python packages we considered, an explanation of our setting and finally the application of our algorithm.

Standard Software Solutions

There are several techniques to export data from a PDF. The first consideration we made was whether standardized software such as Adobe Acrobat Pro, Wondershare PDF Element Pro or Nitro PDF Converter would provide us with reliable and useful results. We tested these software packages for several financial statements, but the result was a large quantity of columns and a lot of redundant mark-up information. Another disadvantage is that these software programs require a paid license and provide little opportunities for batch processing and automating the reconciliation steps after loading the data in a spread sheet format.

Optical Character Recognition (OCR)

If a PDF is non-machine-readable, when it is for example a scan of an original document, it can be made readable by applying OCR. With OCR the computer recognizes the characters from the document and computes a machine-readable file out of the non-machine-readable PDF. A disadvantage of OCR is that it needs much computing power and therefore is relatively slow[4]. In the audit process OCR is generally not needed for the audit subject matter, as the auditor directly interacts with the auditee and can request the PDF in machine-readable format.

Python Packages

There are several Python packages available that are useful in extracting data from (machine-readable) PDF files. We listed the most common of these packages in Table 1 with a short description. We tested all four of these packages and chose to apply *PyMuPDF*. *PyMuPDF* has the advantage that it is not sensitive to the number of pages of the table and can be used in extracting numerical data. Tabula-py performs best when it knows the page number where the data is located, since the page number of tables can differ from financial statements to financial statements, we decided not to apply the Tabula-py package. *PDFMiner* and *PyPDF2* have a focus on text extraction rather than numbers, which is our focus. Another argument in favor of *PyMuPDF* is its relatively simple application in defining upper and lower limits of each table, which in the case of financial information is often the table title.

Our Setting

In the Netherlands, several industries have model financial statements which are required under Dutch GAAP and mandated by Dutch civil law. These models are mandatory, where deviation from the model is only allowed in cases where deviation is necessary to give a true and fair view in the financial statements. Additions of more detailed line items, subtotals and limited adjustments of financial statement line-item names are allowed and are therefore always something that the auditor needs to consider. Dutch Accounting Standard 615 Investment Entities requires the application of a standard model financial statements and a minimum of financial statement line items to specify. Furthermore, the sequence of notes to the financial statements is fixed. The potential of audit efficiencies by automatization, and subsequent reduction of audit fees, might form a powerful incentive for auditees to comply with the standard model financial statements. The standardized format from the model financial statements allows us to apply the regular expressions technique. Regular expressions are a

[4] This still applies when multiprocessing techniques are applied.

technique that matches strings of characters from a certain file. Regular expressions can be accessed via the Python *re* module. Useful free resources to check the performance of regular expressions on several pieces of text are www.pythex.org and www.regex101.com. We wrote regular expressions for all relevant financial statement line items, notes, and disclosures.

Application of the Algorithm

After importing all (unaudited) audit subject matter (PDFs) of a similar structure (GAAP, Model Financial Statements etc.) in one folder, the Python algorithm performs the following steps:

1. Extraction of tables from PDFs.
2. Match the Financial Statement Line Items (FSLIs) of each table in the report by using regular expressions.
3. Extract the data comprising the monetary amounts of the balance sheet, profit-and-loss statement, cash flow statement and the notes for each FSLI.
4. Export the FSLIs names with the data in one CSV output file.

After obtaining the financial statement data in a standardized format, the auditor can then easily automate reconciliation of the CSV output file to the output of the accounting information systems (audit evidence), which is often produced in end-user computing solutions such as Microsoft Excel. This also significantly decreases processing time of auditing subsequent versions of the subject matter, as differences can be easily identified. Finally, the script can be extended with automated reasonableness checks such as matching the sum of individual FSLI's (subtotals) to the subtotals (totals), assets/liabilities matching and other checks that were previously performed by humans in offshoring centers. Any exceptions generated are then easily traceable to FSLI's that must be added to the script, or the fact that the audit subject matter contains errors that must be corrected by the auditee. Our Python code including explanation, documentation and comments can be found in the online Appendix.

REFERENCES

Daugherty, B. E., Dickins, D., and Fennema, M. G. (2012). Offshoring tax and audit procedures: Implications for U.S.-based employee education. *Issues in Accounting Education*, *27*(3), 733–742. doi: 10.2308/iace-50141.

Eastwood, B. (2020). The 10 Most Popular Programming Languages to Learn in 2020.

Available at: https://www.northeastern.edu/graduate/blog/most-popular-programming-languages/ (last accessed December 18, 2020).

Hanes, D. R. (2013). Geographically distributed audit work: Theoretical considerations and future directions. *Journal of Accounting Literature, 32*(1), 1–29.

Lyubimov, A., Arnold, V., and Sutton, S. G. (2013). An examination of the legal liability associated with outsourcing and offshoring audit procedures. *Auditing: A Journal of Practice & Theory*, *32*(2), 97–118. doi: 10.2308/ajpt-50354

Python Software Foundation. (2020). The Python Tutorial. Available at: https://docs.python.org/3/tutorial/index.html#tutorial-index (last accessed December 18, 2020).

Stack Overflow. (2020) Stack Overflow Trends. Available at: https://insights.stackoverflow.com/trends?tags=r%2Cpython%2Cc%2Cc%2B%2B (last accessed December 18, 2020).

van Rossum, G. (1995). Python Reference Manual. In *Centrum voor Wiskunde en Informatica - Computer Science /Department of Algorithmics and Architecture*. Available at: https://ir.cwi.nl/pub/5008/05008D.pdf (last accessed December 18, 2020).

TABLE 1 Python PDF Packages for Data Extraction

Python Package	Description of the Package
PyMuPDF (Fitz)	Python bindings for MuPDF is a lightweight PDF and XPS viewer. The library can access files in PDF, XPS, OpenXPS, epub, comic and fiction book formats, and is known for its top performance and high rendering quality.
Tabula-py	It is a simple Python wrapper of tabula-java, which can read tables from PDFs and convert them into Pandas DataFrames (comparable to Spreadsheets). It also enables you to convert a PDF file into a CSV-, TSV-, or JSON-file.
PDFMiner	Package for information extraction from PDF documents. Unlike other PDF-related tools, PDFMiner focuses mainly on getting and analyzing text data. PDFMiner allows to obtain the exact location of text in a page, as well as other information such as fonts or lines. PDFMiner includes a PDF converter that can transform PDF files into other text formats (such as HTML). It has an extensible PDF parser that can be used for other purposes than text analysis.
PyPDF2	Python library to extract document information and content, split documents page-by-page, merge documents, crop pages, and add watermarks. PyPDF2 supports both unencrypted and encrypted documents.