CURRENT ISSUES IN AUDITING
Vol. 16, No. 1
Spring 2022
pp. A1–A8

American Accounting Association
DOI: 10.2308/CIIA-2020-043

Robotic Process Automation for the Extraction of Audit Information: A Use Case

Jeroen Bellinga
PricewaterhouseCoopers Accountants N.V.

Tjibbe Bosman
University of Amsterdam
Foundation for Auditing Research

Seyit Höcük
Stichting CentERdata
Tilburg University

Wim H. P. Janssen
University of Amsterdam
Foundation for Auditing Research

Alaa Khzam
PricewaterhouseCoopers Accountants N.V.

SUMMARY: The reconciliation of audit evidence to the audit subject matter is a key and recurring audit procedure. Before reconciling information, data needs to be extracted from the audit subject matter, which is often in a Portable Document Format (PDF). Reconciliations are a recurring task for every new version of the audit subject matter.

We thank Denise Dickins (editor), two anonymous reviewers, 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, Amsterdam, The Netherlands, Foundation for Auditing Research, Breukelen, The Netherlands; Seyit Höcük, Stichting CentERdata, Tilburg, The Netherlands, Tilburg University, Tilburg, The Netherlands; Wim H. P. Janssen, University of Amsterdam, Amsterdam Business School, Accounting Section, Amsterdam, The Netherlands, Foundation for Auditing Research, Breukelen, The Netherlands; Alaa Khzam, PricewaterhouseCoopers Accountants N.V., Groningen, The Netherlands.

Supplemental material can be accessed by clicking the link in Appendix A.

Editor's note: Accepted by Denise Dickins.

Submitted: January 2021
Accepted: April 2021
Published Online: May 2021

Large audit firms typically "offshore" simple and repetitive audit tasks such as reconciliations 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, the data extraction for reconciliations. The algorithm resulted in a significant time saving for the audit of a large Dutch asset management firm. Researchers apply the algorithm to minimize hand-collection of financial statement data.

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

JEL Classifications: M42; G23; G29.

Keywords: automation audit tasks; data extraction; Python; reconciliation; investment funds; academic practitioner collaboration.

I. INTRODUCTION

Audit subject matters, including financial statements, are generally prepared as multipage PDF documents. Ensuring what is in the PDF reconciles to the audit evidence is an important but somewhat tedious task for the audit team. Larger 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 multiple 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 require reconciliations before the audit subject matter is finalized.

By applying several techniques in Python, the data extraction part of this process can be automated and accelerated through Robotic Process Automation (RPA), which allows the audit team to save (substantial) time and budget. RPA is one of the technologies from which large innovations are expected in the audit profession (Austin, Carpenter, Christ, and Nielson 2020). Yet the adoption of RPA is lagging in audit compared to the other lines of services (Cooper, Holderness, Sorensen, and Wood 2019). 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.

The outline of this paper is as follows: In Section II we give a brief introduction of our academic practitioner collaboration followed by Section III's short description of the Python programming language. In Section IV, we outline the challenges that audit practitioners generally face when applying new techniques such as RPA in the audit. In Section V, we discuss several possible

techniques applied and explain our choices. We provide the algorithm that we developed, so audit practitioners can customize it to their own situation and needs.

II. OUR ACADEMIC AND PRACTITIONER COLLABORATION

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 are prepared by research associates from FAR.

The manual data collection process is a complex, time-intensive, and expensive process. To use our resources carefully, we decided to standardize and automate this process where possible. We collaborated with the data-science department of Stichting CentERdata to apply data-science techniques in the data collection process and consult us with the implementation.[1] By applying new data techniques, researchers can efficiently obtain data-points that are not available in regular academic databases such as detailed information over impairment disclosures, segment reporting, and other items that can be gathered from the financial statements. Following several discussions with audit practitioners we adjusted the data collections techniques we developed for gathering research data to the needs of audit practitioners and made our algorithm publicly available.[2]

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 particularly 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 offers the opportunity to make code more readable, cluster the code by cells, and add clear notes and documentation directly to the code.[3] Jupyter Notebook Files can provide the auditor with a reperformance audit trail of the procedures performed within the code. Python comes with many packages that can be installed and applied when needed.[4] The packages we incorporate in this work include *Fitz*, *re*, *csv* and *os*. We encourage readers to obtain and install Python from the open-source Anaconda distribution at https://anaconda.org/ and follow the instructions we prepared on GitHub.[5]

[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 asset management audit client for which we adjusted the data collection algorithm.

[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*.

[4] Python packages are basically standard pieces of programming code.

[5] For researchers we can recommend the one-week Python course of Ties de Kok. Resources including instructional videos, slides, and coding problems are freely available under: https://github.com/TiesdeKok/limperg_python

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

Unaudited and unpublished data are subject to client confidentiality and therefore cannot be shared by the auditor with third parties unless the client consents. We therefore developed our script 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 the documentation of the packages we used to ensure that no data are shared outside of the local machine where the algorithm runs 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 for which the audit has been completed, we were able to compare the output of the algorithm 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 (digital associate).

V. TECHNIQUES APPLIED

Our goal is to make structured and reliable machine-readable exports of all the financial information in a batch of draft financial statements (PDF audit subject matter). 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. In this section we discuss the available standard software for PDF data extraction first, followed by a discussion as to whether to apply Optical Character Recognition (OCR), Machine Learning (ML), and Robotic Process Automation (RPA) in this context. In addition, we discuss 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 would provide us with reliable and useful results.[6] 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 opportunity for batch processing and automating the reconciliation steps after loading the data in a spreadsheet format. Batch processing is particularly useful when an audit team audits several sets of financial statements at the same time in for example a group audit setting or the audit of several funds. Researchers are mostly interested in batch processing to collect large and reliable datasets.

Optical Character Recognition (OCR)

Scans of original documents are often non-machine-readable. They can be made machine-readable by the application of 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

[6] We considered the software packages Adobe Acrobat Pro, Wondershare PDF Element Pro, or Nitro PDF Converter.

disadvantage of OCR is that it needs much computing power and therefore is relatively slow.[7] 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. For researchers OCR is sometimes necessary if the original filings are only available in a non-machine-readable format. We leave OCR out of scope for the rest of this paper.

Machine Learning (ML) or Robotic Process Automation (RPA)

Several large audit firms offer ML solutions for data extractions and anomaly detection (Dickey, Blanke, and Seaton 2019). These solutions are generally developed by Big 4 audit firms together with large organizations such as IBM (KPMG 2017) or the *Fraunhofer Institut* (Fraunhofer 2016; PwC 2017). The solutions usually require substantial consulting and license fees and are therefore less accessible for smaller audit practices. Furthermore, these ML applications generally rely upon supervised machine-learning. A manual (supervision) check is therefore still necessary whereas our goal is to reliably automate the manual checks where possible. In addition, machine learning models can lack transparency and the transformation from input to output is not necessarily reperformable, which might make machine-learning problematic under the current auditing standards.[8] We therefore chose to go with an RPA approach where every step is reperformable and the set-up relatively user-friendly.

Python Packages

There are several Python packages available that are useful in extracting data from (machine-readable) PDF files (see Table 1). Some of these packages are specifically designed to extract entire tables from PDF files and give as output the tables in spreadsheet format. Other packages are mainly used for extracting all the content from PDF files, such that the information from tables is not kept in spreadsheet format. For our purposes we are not limited to the first set of packages, if the structure of the tables is somehow kept intact by the package. We tested several packages for performance and convenience and chose to use *PyMuPDF*. This package is not specifically designed for table extraction, but, at least in our case, the information from tables is stored in such a way that the financial statement line item is followed by the amounts. That is, we know which amounts belong to which financial statement line items. This is important, as otherwise we will not be able to extract the right amounts that belong to a specific financial statement line item.

Our Setting

The accounting standards under IFRS require the presentation of financial statements in a consistent mostly prescribed format (IAS 1). Under local Dutch GAAP there are several model financial statements that are mandated by Dutch civil law. Both IFRS and Dutch GAAP models are mandatory, where deviation from the models 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. The potential of audit efficiencies

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

[8] The machine-learning algorithm might have learned something new and therefore produce a different outcome at a different point in time.

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 high 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 users 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.

by automatization, and subsequent reduction of audit fees, might form a powerful incentive for auditees to comply with the standard model financial statements.

We wrote the RPA algorithm for both IFRS and Dutch GAAP, as the asset management firm on which we applied the algorithm has funds reporting under both accounting standards. The standardized format from the model financial statements (IAS 1 and Dutch Accounting Standard 615) allows us to apply the regular expressions technique. The regular expressions technique matches strings of characters from a file. Regular expressions can be accessed via the Python *re* module. Useful free resources to check the performance of regular expressions on pieces of text (strings) are https://pythex.org/ and https://regex101.com/. We wrote regular expressions for all relevant financial statement line items for both IFRS and Dutch GAAP. Readers can download the algorithm and adjust the RPA programming code to their needs by adjusting the regular expressions to other names of financial statement line items (see Appendix A for the link to the downloadable file).

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 by using regular expressions.
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 FSLI 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 or Google Sheets. This also significantly decreases processing time of auditing subsequent versions of the subject matter, as differences can be easily identified. Finally, the algorithm can be extended with automated reasonableness checks such as matching the sum of individual FSLIs to the subtotals (totals) in the financial statements, assets/liabilities matching, and other checks that were previously performed by humans in offshoring centers. Any exceptions generated are then easily traceable to FSLIs that must be added to the code, or the fact that the audit subject matter contains errors that must be corrected by the auditee. Our Python code on including a manual, documentation, and comments can be found on GitHub and in the supplemental material available, see the link in Appendix A.

REFERENCES

Austin, A. A., T. D. Carpenter, M. H. Christ, and C. Nielson. 2021. The data analytics journey: Interactions among auditors, managers, regulation, and technology. 38 (3): 1888–1924. https://doi.org/10.1111/1911-3846.12680

Cooper, L. A., D. K. Holderness, Jr., T. L. Sorensen, and D. A. Wood. 2019. Robotic process automation in public accounting. *Accounting Horizons* 33 (4): 15–35. https://doi.org/10.2308/acch-52466

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

Dickey, G., S. Blanke, and S. Seaton. 2019. Machine learning in auditing current and future applications. *The CPA Journal* 89 (6): 16–21.

Eastwood, B. 2020. *The 10 most popular programming languages to learn in 2020. (June 18)*. Available at: https://www.northeastern.edu/graduate/blog/most-popular-programming-languages/

Fraunhofer IAIS. 2016. *Innovation Campus von Fraunhofer IAIS und PwC erfolgreich gestartet. (July 27)*. Available at: https://www.iais.fraunhofer.de/de/presse/presseinformationen/presseinformationen-2016/presseinformation-160727.html

Hanes, D. R. 2013. Geographically distributed audit work: Theoretical considerations and future directions. *Journal of Accounting Literature* 32 (1): 1–29. https://doi.org/10.1016/j.acclit.2013.09.001

KPMG. 2017. *Automated Contract Data Extraction Within the IFRS 16 Lease Framework*. Amstelveen, The Netherlands: KPMG. Available at: https://assets.kpmg/content/dam/kpmg/de/pdf/Themen/2018/produktflyer-contract-abstraction-tool-info-en-sec-bf.pdf

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

PwC. 2017. *Data Sieve™ application helps simplify and accelerate data extraction for leases*. Available at: https://www.pwc.com/us/en/services/audit-assurance/accounting-advisory/data-sieve.html

Python Software Foundation. 2020. *The Python tutorial*. Available at: https://docs.python.org/3/tutorial/index.html#tutorial-index

Stack Overflow. 2020. *Stack overflow trends*. Available at: https://insights.stackoverflow.com/trends?tags=r%2Cpython%2Cc%2Cc%2B%2B

van Rossum, G. 1995. Python reference manual. In *Centrum voor Wiskunde en Informatica—Computer Science/Department of Algorithmics and Architecture*. Amsterdam, The Netherlands: CWI. Available at: https://ir.cwi.nl/pub/5008/05008D.pdf

APPENDIX A

CIIA-2020-043_audit_reconciliations_Ano: http://dx.doi.org/10.2308/CIIA-2020-043.s01