# Automating Table Extraction From Academic Papers Using Python

**Muhammed Abed**

A report submitted in partial fulfilment of the

requirements for the award of the degree of

BACHELOR OF COMPUTER SCIENCE

CHARLES DARWIN UNIVERSITY

COLLEGE OF ENGINEERING, INFORMATION TECHNOLOGY AND ENVIRONMENT

October 2024

# DECLARATION

I hereby declare that the work herein, now submitted as a report for the degree of Bachelor of Computer Science at Charles Darwin University, is the result of my own investigations, and all references to ideas and work of other researchers have been specifically acknowledged. I hereby certify that the work embodied in this report has not already been accepted in substance for any degree, and is not being currently submitted in candidature for any other degree.

Signature: [illegible]

Date: 20 October 2024

# TABLE OF CONTENTS

# LIST OF FIGURES

# List of Tables

## List of Abbreviations

| | |
|---|---|
| API: | Application Programming Interface |
| CSIRO: | Commonwealth Scientific and Industrial Research Organisation |

# Automating Table Extraction From Academic Papers Using Python

***Abstract*—This paper explores the use of automated tools for extracting tabular data from academic papers, specifically focusing on groundwater studies. The research investigates multiple methods, including OpenAI API, Tabula, and Camelot, to streamline the process of extracting data from complex table formats commonly found in double-column academic papers. After extensive testing, Camelot was selected for deeper analysis due to its superior performance. The study involved the extraction of tables from 29 academic papers, addressing challenges such as noise, missing cells, and inconsistent table structures. Three extraction methods were tested: (1) basic Camelot extraction, (2) enhanced Camelot extraction with data cleaning, and (3) conversion to single-column formats before extraction by Camelot, followed by data cleaning. The results highlight the strengths and limitations of each approach, demonstrating how automated extraction can reduce manual labour but still requires further refinement for optimal results. The findings emphasise the importance of data cleaning and preprocessing to improve accuracy in extracting tabular groundwater data, with potential applications in streamlining research workflows and data collation for groundwater studies. Future work will focus on refining and integrating these methods into broader data management systems.**

***Keywords*— Camelot, double-column PDF, table-extraction, data-noise**

## I. Introduction

*Background*

Groundwater and spring analyses often use isotopes to inform the age of water, which is the time since they 'recharged' an aquifer. Understanding the age of water can inform rates of groundwater recharge, the velocity of groundwater and, therefore, the time between a potential contaminant event and the water moving elsewhere, etc. Groundwater data can be expensive, with some isotope analyses exceeding AUD700 per sample, and time-consuming, with some lab analyses taking 3 to 6 months. These samples are often not reused after they project the first project they are used in. This makes it critical to use this data efficiently. Although there are online resources in Australia that make data from state governments and CSIRO available, they do not include data from academic papers.

*Aim of Research*

The aim was to set up a system that could be used to extract tables from multiple papers with minimal manual intervention. If this can be achieved, the aim would be to collate data from different papers into one table. This will help enable data reuse, cutting costs and making them more accessible.

*Structure of Paper*

This paper first gives an overview of the tools tested for tabular data extraction, including OpenAI API, Tabula, and Camelot. Following this, the experiments section covers the performance of the different tools, and the difficulties encountered during the extraction process. It compares these tools, explaining the rationale behind selecting Camelot for deeper analysis. Different methods created to improve Camelot results were discussed. In the analysis and discussion of results, the paper evaluates the performance of the table extraction methods, focusing on aspects like noise reduction, data accuracy, and how well the tools handle complex table structures. This section compares the different extraction techniques and their effectiveness in addressing the challenges posed by the diverse table formats in the papers. Finally, the conclusion summarises the findings and proposes directions for future research, bringing together the insights gained from analysing groundwater isotopic data through automated table extraction.

## II. Approach

The academic papers selected for this study were downloaded as PDF files sourced through a Scopus search. The search criteria to find academic papers that contained groundwater isotopic data were:

- Search field entry: ("groundwater" OR "hydrogeology" OR "hydrology") AND ("tritium" OR "carbon-14" OR "CFC" OR "SF6")
- Country was limited to Australia
- Published from 1990 to present

The search targeted papers containing groundwater and isotope data in Australia. Papers were then selected based on the following conditions:

- Paper contains tables with isotopic data.
- PDF file is text-based.

The tables in these papers had complex structures that made their extraction a challenging process. There are different tools that are free to use and easily accessible in the academic field to extract tables from academic papers effectively.

The planned approach was to assess various tools to identify tables of varying sizes and levels of complexity within the papers and determine whether this could capture all cells accurately. From the set of 29 papers, 5 papers were randomly selected for the initial testing of these tools. The best-performing tool was then utilised to process all 29 papers together. Additional steps will be added to the extraction process to improve the output if required. This approach was taken when an initial check of the HTML pages that hosted each paper showed that these websites blocked web scraping tools. The tools that were tested are summarised below.

*OpenAI API*

The first approach was to use OpenAI API. This was first tested with the extraction feature in the web version of OpenAI's ChatGPT (OpenAI, 2023). This tool was chosen for its AI-driven capabilities.

*Tabula*

My next approach was to explore Tabula (Ariga, 2016), commonly used for table extraction from PDF files. Tabula has two different methods, or 'flavours', to extract tables, Lattice and Stream. The Stream flavour was used because the tables in the selected academic papers do not have visible lines separating the cells. Stream works by detecting tables based on the whitespace between cells.(Mehta, 2021). Moreover, Tabula's output options (CSV, TSV, JSON) aligned well with the project's goal to obtain a standardised, analysable format.

*Camelot*

Finally, I used Camelot, a Python library designed to extract tables from PDF files. Like Tabula, Camelot uses the flavours Stream and Lattice (Mehta, 2021), and I used Stream here, too. Camelot had a wide range of export options (CSV, JSON, HTML, Markdown, Excel and SQLite), and customizability made it a strong candidate for the automated extraction of complex tables (Bansal, 2023).

## III. Experiments / Testing / Measurements / Data

*Collection of Academic Papers*

A total of 29 papers were selected, with a total of 85 tables, ensuring a range of table orientations and placements, such as tables embedded within columns, landscape tables in portrait-oriented documents, and tables spanning multiple pages. These papers were manually saved as PDF files to be tested with the mentioned tools, as they could not be automatically extracted from the HTML websites.

*Evaluating Tools for Table Extraction*

*HTML extraction*

Web scraping was attempted using Beautiful Soup(Richardson, 2024), a Python library designed to scrape. Unfortunately, the output seemed to be encrypted text. This was because these websites do not allow web scraping tools (ADv0rnik, 2022).

*OpenAI API*

I first tested the extraction of the PDF files using the web version ChatGPT. This approach was met with significant problems:

- While some tables were detected, ChatGPT often failed to return all the tables in the PDF file.
- When attempting to extract larger tables, ChatGPT returned only partial data (e.g., 10 out of 20 rows).
- Cells were often misplaced, with data being shifted to completely different positions than their original location, resulting in corrupted table data.

Although this approach showed promise for smaller and less complex tables, extracting larger tables and multiple tables from the same paper faced difficulties. Besides that, larger tables had their data corrupted when extracted. The corruption of tables can significantly impact the use of tables for research or other purposes. This method ultimately proved unreliable when handling more complex tables.

*Tabula*

In the next approach, after investigating the outputs, the following were observed:

- Columns were often merged into a single cell, distorting the table's structure.
- Many tables were not detected, mostly those embedded within text columns.
- There were many cases where the surrounding text from the page was recorded as part of the table. These texts were stored in the same cell as the table data.
- Portions of tables, especially tables that were larger, were left undetected.
- The contents of tables frequently shifted, resulting in misalignment of data
- Tables can be saved as CSV, TSV or JSON files.

Issues such as merging columns, misplacing data, and overlapping surrounding text with table data can result in more time cleaning and rearranging the data, increasing manual intervention.

*Camelot*

Finally, I used Camelot, a Python library designed to extract tables from PDF files. Camelot allows users to modify various parameters according to their needs to suit the extraction process. Camelot was tested in its default settings and also with modified parameters. Camelot had its own limitations but showed the most promise in handling the complex layouts of the table:

- Camelot's Stream flavour often misinterprets the whitespace between the two columns of text and other whitespaces as part of the table structure, leading to incorrect table detections.
- Camelot seldom returned tables with merged columns or rows.
- Table contents remained in their position for most of the tables.
- In some instances where tables had surrounding text, Camelot returned this text along with the table. However, it was able to keep the text

separate from the table content, preventing any data corruption.

- The edge tolerance parameter of the Camelot library was modified to avoid data noise. Although it helped reduce data noise in some cases, other tables in the same paper would remain undetected.
- Tables can be saved as CSV, JSON, HTML, SQLite, Excel, or markdown files.
- The ‘flag_size’ argument was set to True. This places superscripts and subscripts between the tags ‘<s></s>’. This is done as these texts cannot be extracted in their original form and are converted to normal-sized text.

Camelot faced a notable challenge when using the Stream method. The academic papers chosen for this study are double-columned, with whitespace separating the columns. Camelot’s Stream flavour often misinterprets the whitespace between the two columns of text and other whitespaces as part of a table structure, leading to incorrect table detections. However, its ability to consistently detect tables while maintaining data integrity makes it an excellent choice for further testing. Camelot proved to be the most reliable approach for extracting accurate tables with complex layouts. These points minimise the need for manual intervention, reducing the time to prepare the data table. Camelot’s extensive range of parameters allows users to improve the extraction process. Moreover, using Camelot, tables can be exported in multiple formats, such as CSV, JSON, Excel, etc. This allows data to be processed and analysed further with minimal manual intervention (Bansal, 2023).

To proceed with the research, Camelot was chosen over the other tools primarily due to its ability to maintain table integrity, even with complex structures. Unlike OpenAI’s ChatGPT, which frequently returned incomplete and corrupted tables, and Tabula, which struggled with merging columns and was misaligning data, Camelot provided the most reliable extractions with minimal data shifting or merging issues. Compared to Tabula, Camelot’s Stream flavour allowed for more accurate table structure detection and separation of surrounding text, reducing the manual intervention needed to clean and organise the extracted table. Additionally, Camelot’s range of export options and customizability makes it an optimal choice for handling the intricate layouts common in academic PDF files.

*Enhanced Camelot-Based Methods*

Since changing Camelot’s parameters resulted in undetected tables, and the tool with default parameters resulted in significant data noise, two additional programs were made that used Camelot, with its default configuration, as the primary tool for the table extraction shown in Fig. 1. These methods aimed to improve the final table output. The outputs were then analysed to compare, which would result in more reliable, cleaner outputs. The three different programs are:

- Camelot_scrape: This program directly applies Camelot to the original PDF files. Camelot attempts to detect tables using Stream flavour.

- Clean_camelot: This program introduces additional data-cleaning steps after Camelot's table detection.

  The steps include:
  1. Remove table cells containing 30 or more characters.
  2. Delete tables with less than or equal to 150 characters.
  3. Delete tables where less than 7% of the characters are numeric.

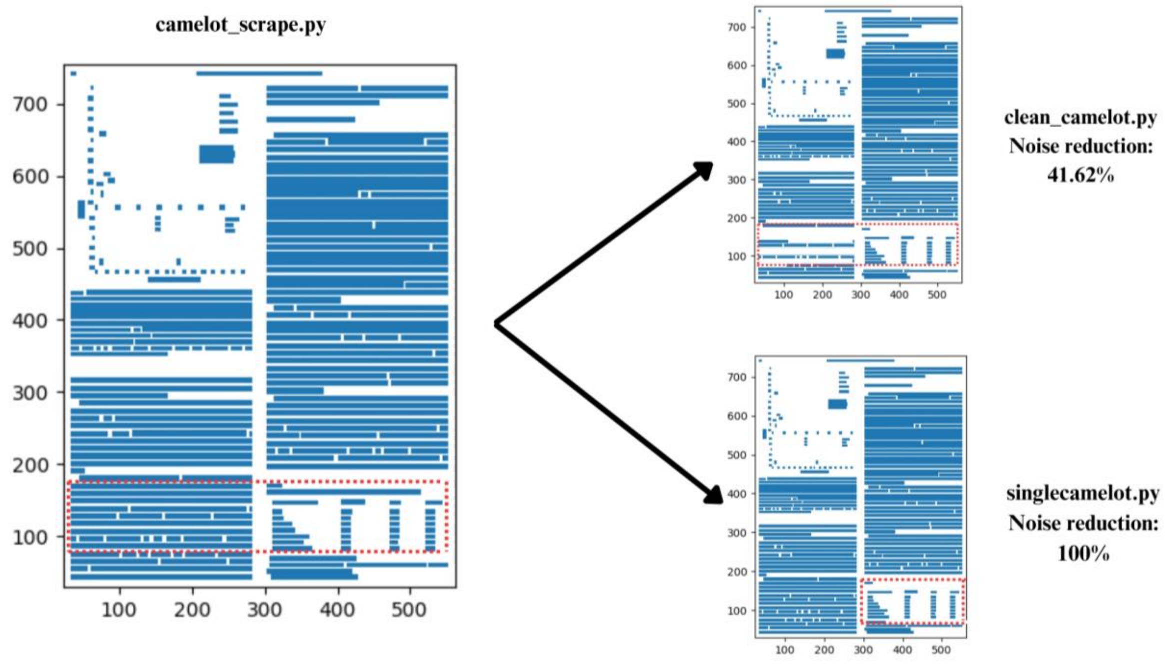


**Fig. 1** Sample showing output of different table extraction methods

4. Delete rows and columns with zero entries.
5. Delete tables with two or fewer columns.
6. If, after the cleaning, no numeric characters can be found in the table, delete the table.

This method aims to reduce noise and improve table extraction accuracy by focusing on the actual table structure.

- Singlecamelot: This program converts the double-column PDF files into single-column Word documents. Then, they are converted to single-column PDF files before running the same table extraction and cleaning processes as clean_camelot. This method uses the pywin32 (Hammond, 2023) Python library, as it automates this conversion with no input required from the user. The whitespace between two columns has often been an obstacle for Camelot in distinguishing between texts and tables. Converting PDF files to single-column format aims to eliminate the problematic whitespace between columns, which often confuses Camelot's extraction process.

The three methods of table extraction were applied to the complete set of papers. Each method processed all the PDF files together. The processing time for each program was recorded. The outputs from each program were saved as CSV files and analysed for accuracy and noise levels. Fig. 1 shows an example of the outputs from the different methods of extracting the same column-embedded table, where blue bars represent the text detected, and the red-dashed boxes show the area detected as a table. For all the papers, camelot_scrape returned CSV files with no real table, in other words, false positives. When no tables were present on a page, it would misinterpret the two columns of text as a single table. Often, if a page had a table with text above or below, it would return two CSV files: one with the actual table and another with just the text. To test if the cleaning methods could minimise false positives, the papers in which they occurred were recorded. Additionally, comments were made on the false positives detected to understand which parts of the paper trigger false positives. Moreover, I investigated each method's output for all the tables and recorded the properties and errors given in Table 1.

For each table type, the percentage of time these errors occurred was recorded, along with the mean noise reduction for the cleaning methods. The data collected was categorised to help identify the efficiency of each extraction method in handling specific table structures and layout types and to understand the performance of these methods with different table layouts (see Table 4 in Appendix D). Only table types with more than ten samples were analysed to make the analysis more reliable. Undetected tables were considered to be tables with missing data.

**Table 1** Properties and errors recorded for each output.

| **Property/Error** | **Description** |
|---|---|
| Table type | Table layout. |
| Program | The program used for extraction. |
| Detected | Recorded as 'Y' or N'.<br>If table contents are completely mixed with surrounding text and cannot be differentiated, the table is considered undetected. |
| Columns/rows merged | Recorded as 'Y' or N'. |
| Content shifted | Recorded as 'Y' or N'. |
| Missing data | Recorded as 'Y' or N'. If the Table number, title or footnote are missing, they are not considered missing data. |
| Additional Comments | Comment on additional observations. |
| Table cells detected | Number of table cells detected. Table number, title and footnote are counted as table cells. |
| Non-table cells detected | Number of non-table cells detected. |
| Total cells detected | Total number of cells detected. |
| Noise percentage | (Number of non-table cells detected / Total number of cells detected) * 100. |
| Noise reduction (cleaning methods only) | Difference between the noise detected with camelot_scrape and the cleaning method / the noise detected with camelot_scrape. |

## IV. Analysis and Discussion of Results

### *Analysis of Noise Reduction*

Fig. 3 in Appendix C shows that camelot_scrape detected the most noise, around 33%, extracting column-embedded tables, while landscape tables and those below text were extracted with less than 1% noise. The cleaning methods excelled in removing data noise from column-embedded tables, where clean_camelot and singlecamelot had a mean noise reduction of 65% and 90%, respectively. For the tables above text, both methods achieved cleaning more than 30% of the noise; however, they were only able to achieve less than 6% noise reduction for the other table types. For every paper, camelot_scrape returned CSV files with no actual table data but other texts that were detected as columns. Using clean_camelot resulted in these false positives falling by a significant 90%. Chart labels accounted for all the false positives detected by clean_camelot. Although singlecamelot reduced these false positives by 69%, it did not perform as well as clean_camelot. Besides chart labels, singlecamelot also detected some reference page references as tables, accounting for around 75% of these false positives (See Fig. 5 in Appendix C).

*Analysis of Table Structure and Data Loss*

Camelot_scrape detected all landscape tables, leaving around 15% and 5% of tables undetected, which were below or above the text, respectively. Column-embedded tables had the highest percentage of tables undetected, at around 25% (See Fig. 6 in Appendix C). Singlecamelot had a high failure rate detecting landscape tables. Upon further investigation, I found that landscape tables were often converted to images when converting the double-column PDF file to a single-column. Since Camelot is designed only to extract text-based tables, the converted tables remain undetected. In some cases, clean_camelot had slightly higher failure rates in table detection than camelot_scrape. This usually happened with tables that had cells containing more than 30 characters. Some tables had their headings removed as they contained superscripts and subscripts. These texts were flagged for their size, exceeding the cells' 30-character limit. This led to the cells' removal, which may, in turn, result in the cells, columns, or rows meeting other cleaning conditions that would lead to the deletion of the table. Upon further investigation, it was found that the tables undetected by clean_camelot did not contain numeric data. Since the aim is to collect isotopic readings, losing this data would not be a disadvantage to this method.

Fig. 7 Appendix C shows that camelot_scrape outperforms all the other methods in capturing table cells regardless of the table type. Clean_camelot misses data more often than camelot_scrape for all table types. As previously mentioned, clean_camelot's cleaning steps deleted table contents, which was the reason for the higher data loss in all table types. Fig. 8 in Appendix C shows a table extracted by camelot_scrape and clean_camelot. Characters flagged for size can be seen in between tags. Notice that the second and last rows were removed by clean_camelot. This is because the cell contents in these rows exceeded the character limit, resulting in their deletion. This made the rows empty, which were then removed during a cleaning step. Singlecamelot performed poorly regarding table completeness. Except for column-embedded tables, singlecamelot had the highest percentage of missing data when extracting other table types. The differences recorded between the second highest and singlecamelot (highest) for tables above and below text are about 50% and 55%, respectively. There are multiple reasons for this:

- During the conversion to single-column PDF files, the row(s) containing column headings would be converted to an image, making it impossible for Camelot to detect.
- Some tables are split across two pages, where one part of the table would be extracted and the other would not.
- Singlecamelot uses the same cleaning process as clean_camelot. Hence, it deletes table cells that meet the criteria to be removed.

Sometimes, table cells would shift one column left or right or one row up or down. Regarding content shifting, camelot_scrape and clean_camelot had the same percentage of tables where cells were shifted for all table types. There is a drastic increase in this percentage for all table types extracted using singlecamelot.py. Although camelot_scrape and clean_camelot did not face this problem when extracting column-embedded tables, a small percentage of the tables were returned with misplaced data for other table types (Fig. 9 in Appendix C).

In other cases, the row or column would completely merge with an adjacent row or column. Fig. 11 in Appendix C shows an instance where singlecamelot merged two columns into one. In the fourth row from the bottom, we can also see a reading in the 'Distance from coast (km)' column, moved to the column to its left. Camelot_scrape and clean_camelot always had an equal percentage of tables with merged rows or columns (Fig. 10 in Appendix C). No merged rows or columns were found when extracting landscape tables and those below text. However, with column-embedded tables and those above text, around 8% and 4%, respectively, faced this problem. On the other hand, singlecamelot did not merge any rows or columns with column-embedded tables. In contrast, with other table types, it returned the most tables facing this problem, the highest being 10.7% extracting tables above text.

*Other Observations*

The PDF files from which tables were extracted had different encodings, which caused some characters to be misinterpreted or changed to some set of characters. The CSV files had to be imported using the 'Get Data' function to view the outputs properly on Microsoft Excel. This solved the problem with almost all tables. Another problem faced was, for some papers, the symbols were replaced with letters; for example, 'δ' was replaced with 'd'. In addition, there were a few occurrences where spaces between words were not detected, detecting the words as a single word. The three processes took only a few minutes to execute 29 papers. Camelot_scrape and clean_camelot took less than 5 minutes, while singlecamelot took around 14 minutes (Table 2).

**Table 2** Processing time for each program

| Program | Time taken / s |
|---|---|
| camelot_scrape.py | 484.09 |
| clean_camelot.py | 531.82 |
| single_camelot.py | 840.05 |

Reviewing the academic papers, I encountered various types of groundwater data, including chemical compositions, isotope ratios, and hydrological measurements such as water depth, pH levels, and temperature. These datasets were presented in different formats, ranging from structured tables to unstructured text, making it challenging to create a unified metadata set. Moreover, the units of measurement and data structures varied across studies, further complicating the process. This highlights the need for specialised algorithms to standardise these diverse formats into a cohesive and usable dataset.

*Discussion*

*Method 1: camelot_scrape*

While it successfully identified tables, it also detected text as tables. This issue arises because Camelot uses whitespace to simulate table structures, leading to noisy outputs where text between columns or paragraphs is misinterpreted as table data. Although the method reliably detects isolated tables, those within columns produce CSV files containing additional text, requiring manual cleaning. However, table structures were maintained accurately in over 95% of the cases.

*Method 2: clean_camelot*

Because this uses the same approach to extracting tables as the first method, the table structure maintained by clean_camelot was equally good. The cleaning steps effectively reduced noise in most cases while significantly reducing the occurrence of false positives. The process ensured that the method returned tables containing numeric and isotopic data and discarded tables containing mostly alphabetic characters. However, the filtering process was not foolproof; in a few cases where surrounding texts did not meet the cleaning rules, they remained in the extracted table output. Overall, clean_camelot effectively extracts tables while maintaining their structure and minimising data loss.

*Method 3: singlecamelot*

This conversion was beneficial for extracting tables embedded within columns, reducing the chance of text being misclassified as part of the table. However, the conversion process led to new issues. Landscape tables were often converted into images, making them unreadable by Camelot. Additionally, during the conversion, cells from different columns were sometimes merged, or table content was misplaced, resulting in entries from one column appearing in another. In cases where tables spanned multiple pages, this method would sometimes detect the table on one page but not on the other. Despite proving to be the most effective way to extract data from column-embedded tables, its struggles to maintain table structure and accurately extract data makes it unreliable for use on a variety of papers

## V. Conclusions

Table extraction from research papers is crucial, especially when gathering unique data that may not be available in public online resources. While state governments provide certain datasets, academic papers often contain valuable, specialised data that must be extracted manually, which can be time-consuming and tedious. Since data collection, particularly in fields like groundwater studies, can be expensive and labour-intensive, improving table extraction methods would significantly enhance data accessibility, potentially reducing costs.

In this research, I initially attempted to extract tabular data by web scraping; however, many websites restrict access to web scrapers, rendering this method ineffective. As a result, I shifted to working with downloaded PDF files, using tools like OpenAI, Tabula, and Camelot to extract tables. While Camelot performed the best among the three, it produced noisy outputs that required further refinement.

To address these issues, I developed two programs using Camelot as the primary tool. The singlecamelot program excelled with column-embedded tables but struggled with other table structures, often failing to maintain the correct format and missing data. In contrast, clean_camelot outperformed singlecamelot in all other table categories, particularly in improving data clarity and reducing noise. The choice of program would come down to which table types the user is mostly dealing with. Singlecamelot would be the best option when extracting column-embedded tables, while clean_camelot would be the better option in other cases. The programs' parameters can be modified according to the user's needs. For example, if the user wants to extract tables with cells containing more than 30 characters, they can increase the limit accordingly to avoid losing data. Extracting data is the first step in data collation, and this method makes the extraction process much quicker and easier than doing it manually.

## VI. Future Work

Future work could also include collaborating with experts in computer science, hydrology, and information science who could help develop more specialised extraction algorithms tailored to the unique needs of groundwater data. Beyond extraction, the creation of a centralised database for academic groundwater data could be considered, potentially mandated by government agencies, ensuring researchers contribute to this database that could save costs and time and enhance research outcomes. The proposed methods could extract data from academic papers and create a repository where common properties, such as location coordinates and depth, can relate to the new and existing data.

# APPENDIX A: LITERATURE REVIEW

## Introduction of Literature Review

With the increase in number of academic papers published, it is becoming challenging for researchers looking to efficiently access and analyse the information contained within the document. This ever-growing body of literature necessitates the development of effective methods for extracting tables from academic papers. These tables hold crucial insights for scientific reviews and research projects (Mirdamadi et al., 2018). These tables encapsulate quantitative data, statistical analyses, comparisons, and summaries, offering a concise and organised representation of research findings (Huang et al., 2024). Unfortunately, the inherent complexity of extracting tabular data from academic papers, often in the form of PDF documents, presents significant obstacles for researchers (Corrêa & Zander, 2017; Ghosh Chowdhury et al., 2022; Jinghong et al., 2023; Vine et al., 2019). This literature review is organised thematically and focuses on three main areas: (1) the importance of tabular data extraction, (2) challenges in extracting tabular data, and (3) existing methods and techniques. This literature review explores current developments, identifies challenges, and highlights promising future directions in this field.

## Review of Literature on Tabular Data Extraction

*Importance of Tabular Data Extraction*

Automated table extraction allows researchers to easily access and analyse large amounts of data, aiding knowledge discovery and scientific reviews. This information can be used for trend analysis, generating insights and decision-making in many areas of research (Huang et al., 2024; Mirdamadi et al., 2018). In the field of biomedical research, for example, extracting data from clinical trial tables can provide valuable information about patient demographics, treatment outcomes, and safety profiles (Milosevic et al., 2016). Besides that, tabular data extraction is essential for the creation of knowledge graphs which is a structured interconnected representation of knowledge. This will allow researchers to mine tables for data and fill knowledge graphs with factual information making them more complete and correct. This extracted data can also be used to enhance current knowledge bases, allowing researchers to examine relationships and trends between various subjects (Huang et al., 2024).

*Challenges in Extracting Tabular Data from Academic Papers*

There are many difficulties that are encountered when trying to obtain table data from research papers, especially those in PDF format, and these difficulties stem from the nature of these papers themselves. A primary problem is the variety of table layouts and structures. Academic papers can be different in their formatting styles, with tables of varying arrangements of rows, columns, and cells. The presence of non-tabular elements, such as images, figures, and text blocks, can further complicate the extraction process. Also, tables may span across multiple pages, hence, advanced methods must be employed to properly and precisely recreate the entire table structure (Corrêa & Zander, 2017; Ghosh Chowdhury et al., 2022; Jinghong et al., 2023; Vine et al., 2019).

Accurately identifying table boundaries and recognising internal table structures is another significant challenge. Tables may not always follow a standard rectangular format, with some exhibiting irregular shapes or open boundaries. The presence of merged cells makes it much more difficult to determine the boundaries of the table and to recreate the internal structure. In some cases, cells may be missing or have unclear data, making it difficult to accurately extract and interpret the information in a table. (Corrêa & Zander, 2017; Ghosh Chowdhury et al., 2022; Jinghong et al., 2023; Vine et al., 2019). The complex nature of these challenges necessitates the development of robust and flexible extraction methods that can effectively handle the diverse characteristics of academic papers.

*Methods and Techniques for Tabular Data Extraction*

The extraction of tabular data from academic papers has been tackled using a variety of methods and techniques, each with its strengths and limitations. Rule-based approaches, relying on predefined rules and patterns to identify and extract tabular data, have been widely employed. These approaches often leverage heuristics based on the visual characteristics of tables, such as line intersections and cell boundaries, to identify and extract table content. Although rule-based methods perform well for a defined table structure, they fail when it comes to changes in layout and formatting; thus, they are not as flexible with academic papers (Corrêa & Zander, 2017; Ghosh Chowdhury et al., 2022; Vine et al., 2019).

Machine learning techniques have emerged as an alternative to rule-based approaches, offering greater flexibility and adaptability to diverse table structures. These techniques leverage algorithms that learn from training data, enabling them to identify and extract tabular data from a wider range of documents. Machine learning methods like supervised learning can be trained to recognise table boundaries and extract content by being trained on annotated datasets. On the downside, annotated datasets are sometimes very tedious and costly to produce, and this makes supervised learning, inapplicable in some cases (Corrêa & Zander, 2017; Ghosh Chowdhury et al., 2022; Jinghong et al., 2023; Vine et al., 2019).

Self-supervised learning, a relatively new approach, has shown promise in reducing the dependence on annotated data. This technique leverages unlabelled data to learn representations of tables, enabling the development of models that can perform effective extraction without requiring extensive manual annotation (Ghosh Chowdhury et al., 2022).

Hybrid methods, combining rule-based and machine-learning approaches, have been proposed to leverage the strengths of both techniques. (Jinghong Li, 2023) These methods utilise rule-based approaches for initial table detection and then employ machine learning algorithms to refine the extracted information. (Jinghong Li, 2023) This combination allows for more accurate and robust extraction, addressing the limitations of individual approaches. (Jinghong Li, 2023)

*Evaluation of Existing Tools and Frameworks*

Many tools and frameworks have been created to ease the process of tabular data extraction from research papers. These tools use many different methods and theories from rule based to machine learning algorithms. It is important to assess the strengths and weaknesses of these tools in order to determine how well they work and what aspects could be improved (Corrêa & Zander, 2017; Milosevic et al., 2016; Mirdamadi et al., 2018; Wiechork & Charão, 2021).

Mirdamadi et al. (2018) evaluated the performance of three table extraction tools - PDFGenie, PDF2Table, and PDF2HTML - using a manually created ground truth and a framework based on the Cranfield approach. This approach involved defining a set of questions and evaluating the answers provided by the search engine after indexing the extracted tables. The evaluation results revealed variations in the accuracy and performance of these tools, highlighting the importance of selecting the appropriate tool for specific table structures and extraction tasks. Similarly, Wiechork & Charão, (2021) evaluated the performance of several tools for extracting data from PDF documents, including Excalibur and Tabula for tabular data extraction, CyberPDF and PDFMiner for textual content extraction, and Aletheia and ExamClipper for region-of-interest extraction. The results highlighted the limitations of these tools in handling diverse document layouts and the need for further development to address these challenges.

*Emerging Trends and Future Directions*

The field of table data extraction from research papers is one that is constantly changing, with new trends and breakthroughs constantly emerging as a result of the creation of new technologies and methods. One of the most promising approaches to improving table extraction accuracy and robustness is through the use of deep learning techniques and, more specifically, convolutional neural networks (CNNs). CNNs can learn complex features from images of tables and, therefore, are much better able to identify table boundaries and extract content (Ghosh Chowdhury et al., 2022; Jinghong et al., 2023; Vine et al., 2019). Generative adversarial networks (GANs), a type of deep learning model, have also been explored for table extraction, particularly for generating standardised table skeletons from table images. This approach allows for more accurate and consistent table structure recognition, improving the overall extraction process (Vine et al., 2019).

Furthermore, integrating tabular data extraction with other information retrieval and knowledge management tasks is an emerging trend. Extracted tabular data can be integrated with semantic analysis techniques to enhance the understanding of table content. This integration can lead to the development of more sophisticated search systems that can retrieve information based on the content of tables. The extracted data can also be used to populate knowledge graphs, enriching these repositories with factual information and enabling researchers to explore connections and patterns across different domains (Huang et al., 2024; Mirdamadi et al., 2018)

.

Despite these advancements, several challenges remain in the field of tabular data extraction. The development of more robust and versatile tools that can handle diverse table structures and document layouts is a crucial area for future research. The need for larger and more diverse datasets for training and evaluating extraction models is also critical. Furthermore, integrating domain-specific knowledge into extraction methods can significantly enhance the accuracy and relevance of the extracted data (Ghosh Chowdhury et al., 2022; Huang et al., 2024; Jinghong et al., 2023; Mirdamadi et al., 2018; Vine et al., 2019).

## Conclusions

This literature review has explored the landscape of extracting tabular data from academic papers, highlighting the significance of this task for various research and analysis purposes. While advancements in rule-based, machine learning, and hybrid approaches have shown promise, challenges remain in handling the diversity of table structures, document layouts, and the presence of non-tabular elements. The integration of tabular data extraction with other information retrieval and knowledge management tasks presents a promising avenue for future research. However, further research and development is essential to allow researchers to effectively use and study the vast amount of knowledge that is contained within academic papers and thus, further the growth of knowledge and innovation.

## Appendix B: Data and Programs Used for this Report

### Programs

- *Camelot_scrape.py*: Default Camelot table extraction
- *Clean_camelot.py*: Cleaning steps applied after default Camelot table extraction.
- *Singlecamelot.py*: PDF file is converted to single column, then default Camelot extraction performed, followed by cleaning steps of clean_camelot.py
- *Tabula.py*: Tabula table extraction.
- *Web_scrape.py*: HTML page extraction using Beautiful Soup.
- *Graphs.pptx*: Used to create graphs for the paper.

### Data

- *Results.xlsx:* Recorded data from outputs

## Appendix C: Figures

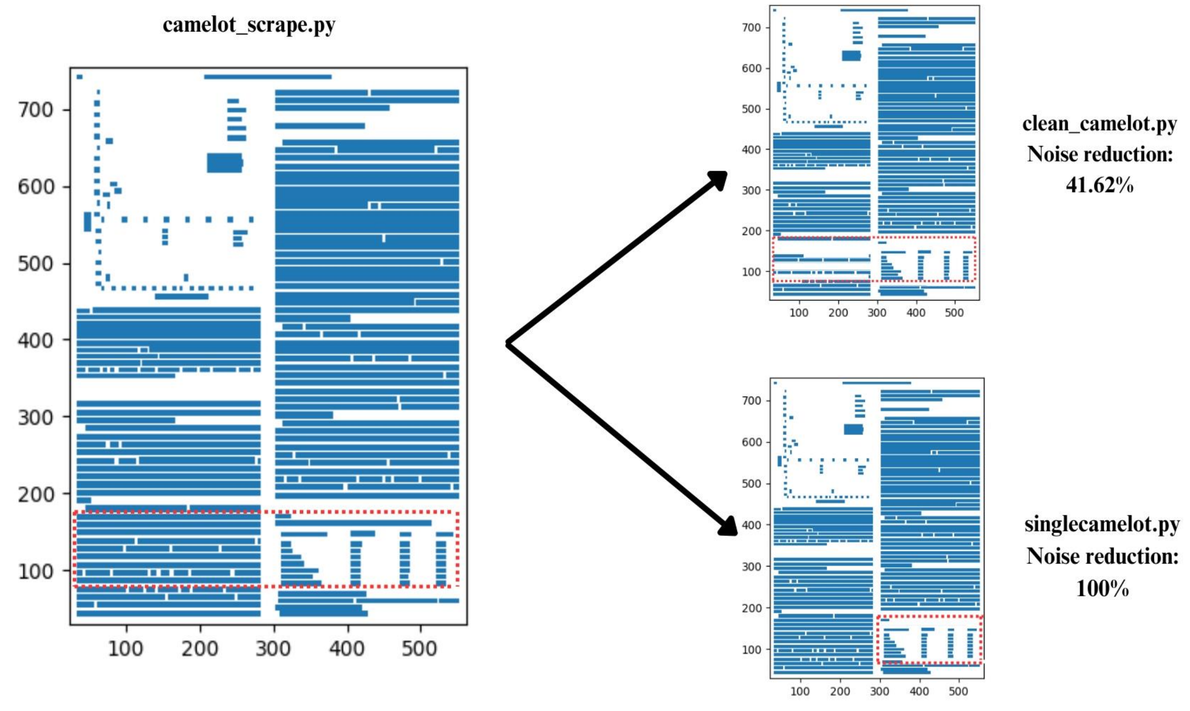


**Fig. 2** Sample showing output of different table extraction methods

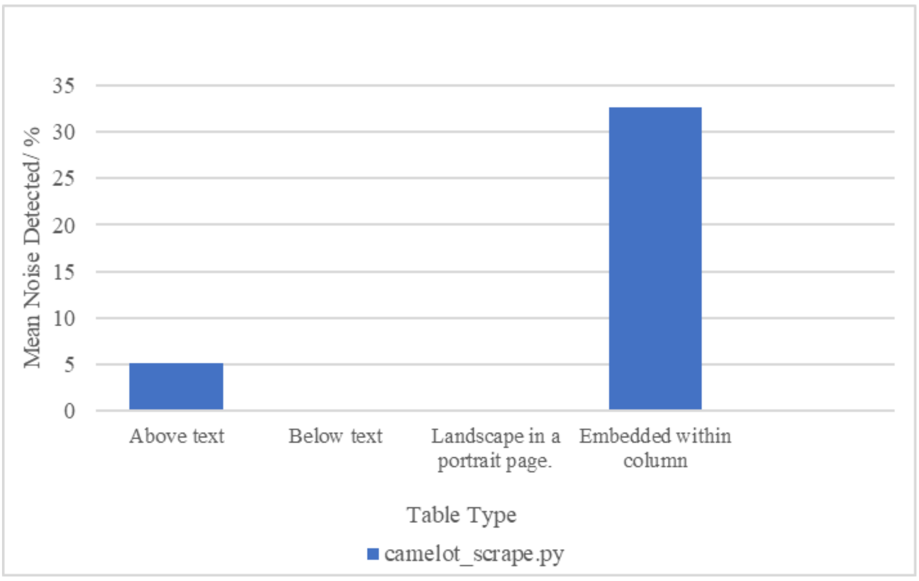


**Fig. 3** Mean percentage of noise detected by camelot_scrape

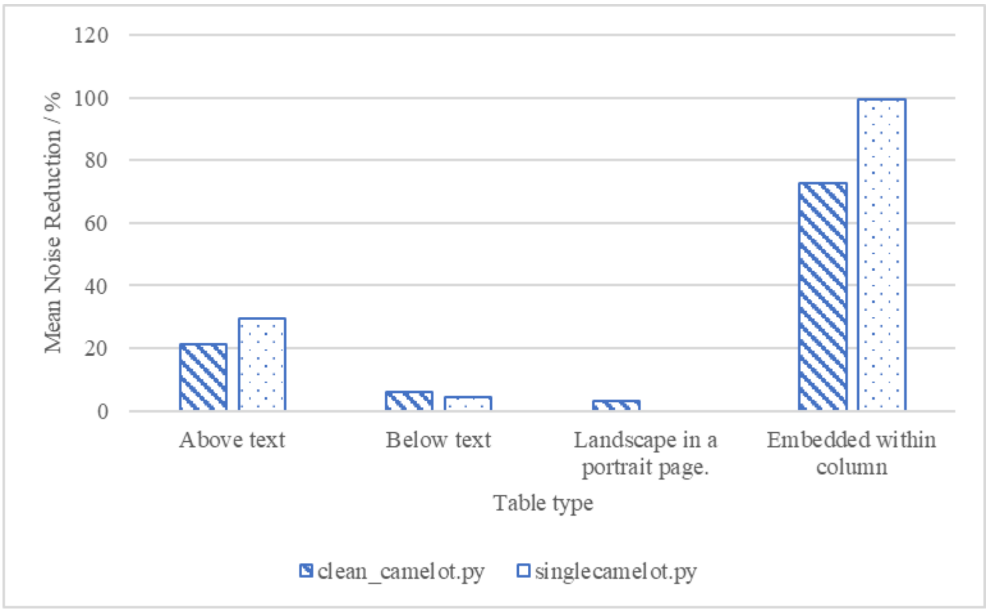


**Fig. 4** Mean noise reduction

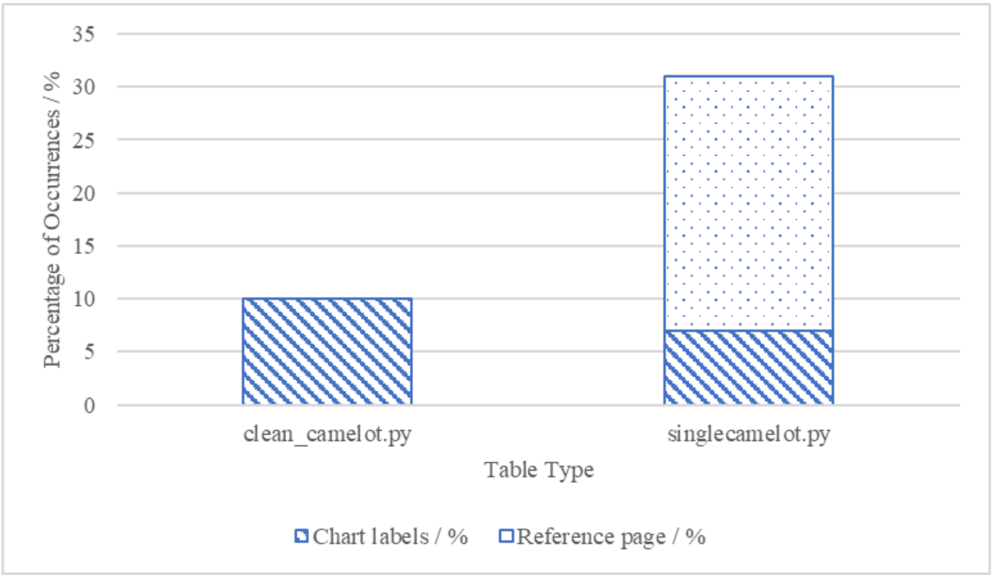


**Fig. 5** Percentage of papers with false positives

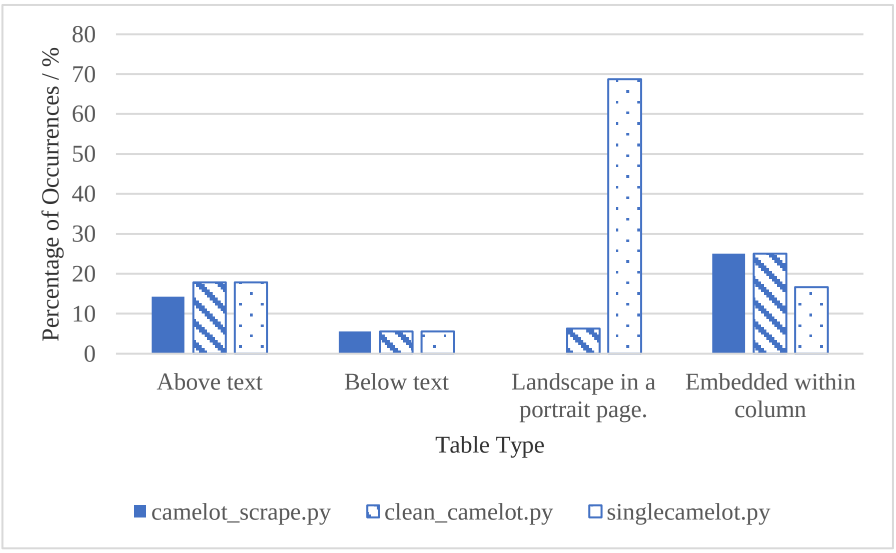


**Fig. 6** Percentage of tables undetected

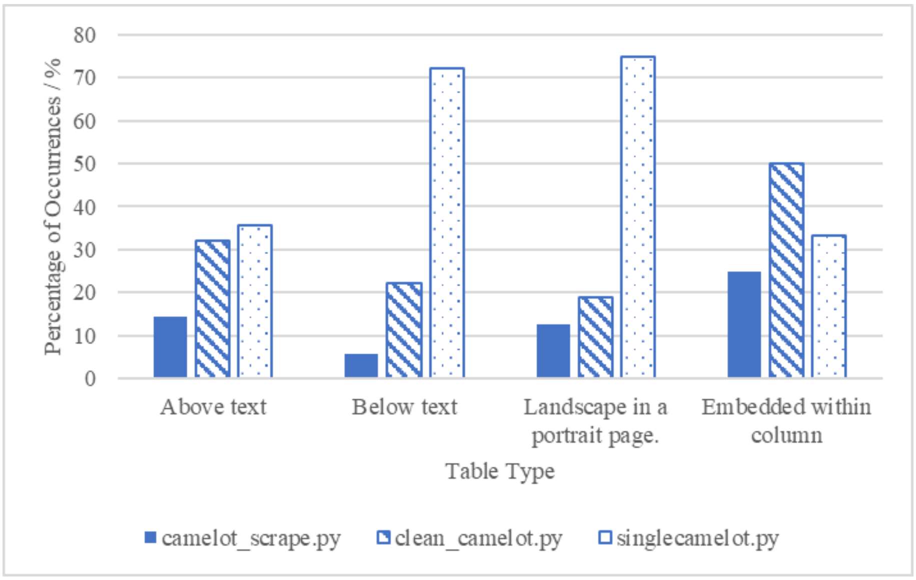
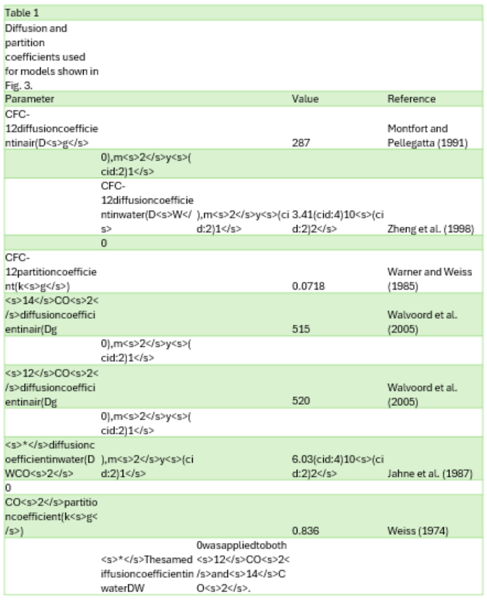
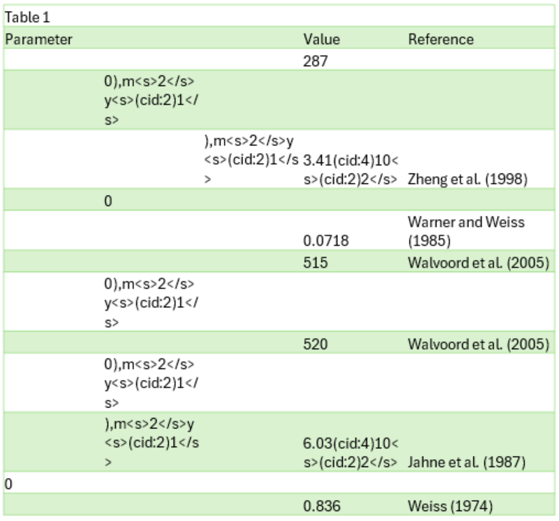


**Fig. 7** Percentage of tables with missing data



| Table 1 | | | | |
|---|---|---|---|---|
| Diffusion and partition coefficients used for models shown in Fig. 3. | | | | |
| Parameter | | | Value | Reference |
| CFC-12diffusioncoefficientinair(D<s>g</s> | | | 287 | Montfort and Pellegatta (1991) |
| | 0),m<s>2</s>y<s>(cid:2)1</s> | | | |
| | CFC-12diffusioncoefficientinwater(D<s>W</s> | ),m<s>2</s>y<s>(cid:2)1</s> | 3.41(cid:4)10<s>(cid:2)2</s> | Zheng et al. (1998) |
| | 0 | | | |
| CFC-12partitioncoefficient(k<s>g</s>) | | | 0.0718 | Warner and Weiss (1985) |
| <s>14</s>CO<s>2</s>diffusioncoefficientinair(Dg | | | 515 | Walvoord et al. (2005) |
| | 0),m<s>2</s>y<s>(cid:2)1</s> | | | |
| <s>12</s>CO<s>2</s>diffusioncoefficientinair(Dg | | | 520 | Walvoord et al. (2005) |
| | 0),m<s>2</s>y<s>(cid:2)1</s> | | | |
| <s>*</s>diffusioncoefficientinwater(DWCO<s>2</s> | ),m<s>2</s>y<s>(cid:2)1</s> | | 6.03(cid:4)10<s>(cid:2)2</s> | Jahne et al. (1987) |
| 0 | | | | |
| CO<s>2</s>partitioncoefficient(k<s>g</s>) | | | 0.836 | Weiss (1974) |
| | <s>*</s>Thesamediffusioncoefficientin waterDW | 0wasappliedtoboth <s>12</s>CO<s>2</s>and<s>14</s>CO<s>2</s>. | | |

camelot_scrape



| Table 1 | | | | |
|---|---|---|---|---|
| Parameter | | | Value | Reference |
| | | | 287 | |
| | 0),m<s>2</s>y<s>(cid:2)1</s> | | | |
| | | ),m<s>2</s>y<s>(cid:2)1</s> | 3.41(cid:4)10<s>(cid:2)2</s> | Zheng et al. (1998) |
| | 0 | | | |
| | | | 0.0718 | Warner and Weiss (1985) |
| | | | 515 | Walvoord et al. (2005) |
| | 0),m<s>2</s>y<s>(cid:2)1</s> | | | |
| | | | 520 | Walvoord et al. (2005) |
| | 0),m<s>2</s>y<s>(cid:2)1</s> | | | |
| | ),m<s>2</s>y<s>(cid:2)1</s> | | 6.03(cid:4)10<s>(cid:2)2</s> | Jahne et al. (1987) |
| 0 | | | | |
| | | | 0.836 | Weiss (1974) |

clean_camelot

**Fig. 8** Data loss after cleaning steps applied on table (Wood et al., 2014) in clean_camelot(right)

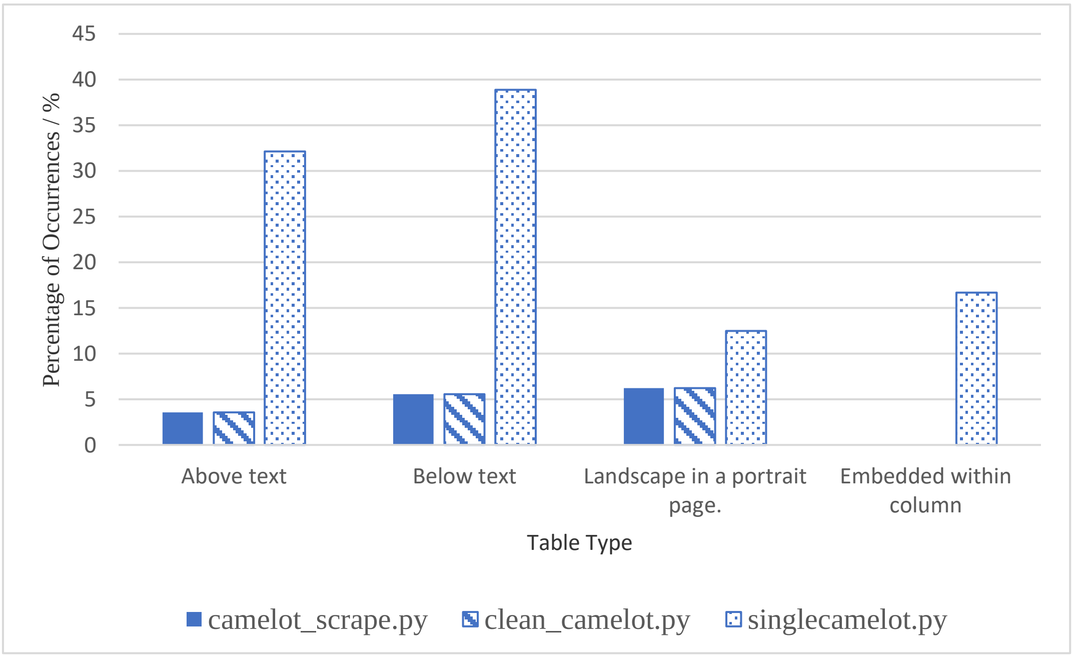


**Fig. 9** Percentage of tables where content shifted

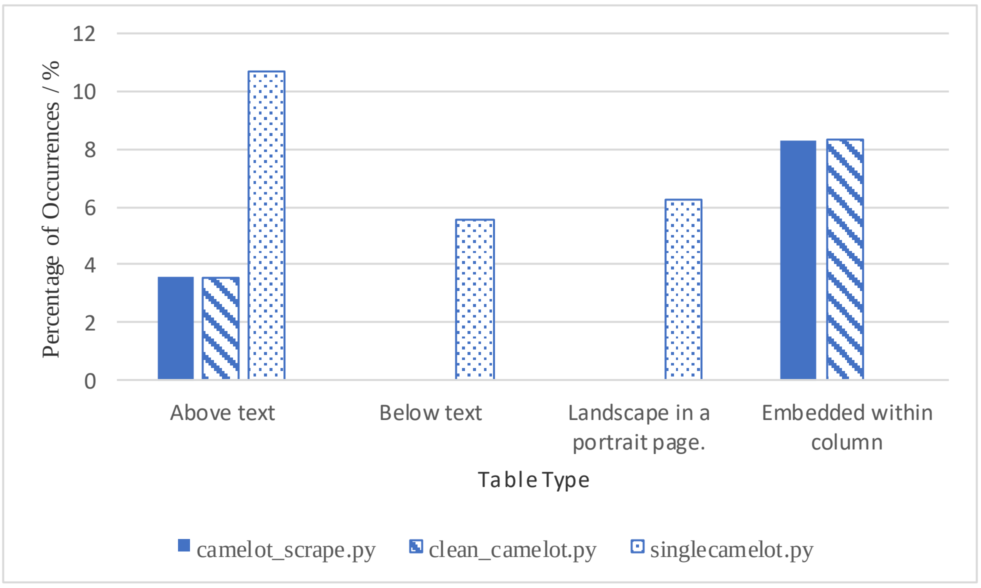
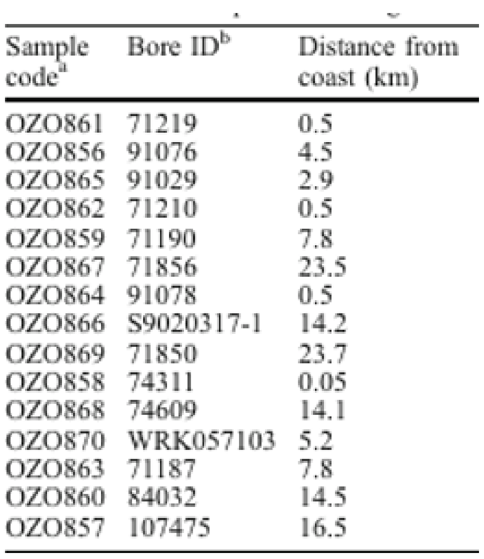
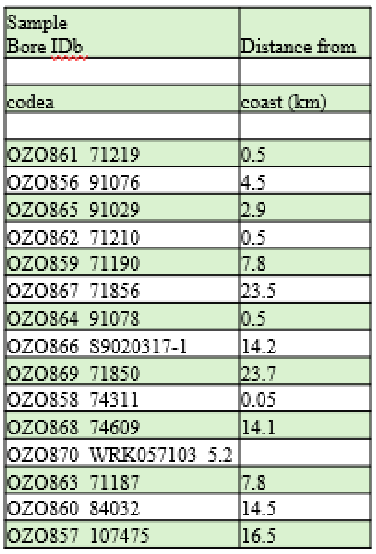


**Fig. 10** Percentage of tables with merged rows or columns



| Sample code[a] | Bore ID[b] | Distance from coast (km) |
|---|---|---|
| OZO861 | 71219 | 0.5 |
| OZO856 | 91076 | 4.5 |
| OZO865 | 91029 | 2.9 |
| OZO862 | 71210 | 0.5 |
| OZO859 | 71190 | 7.8 |
| OZO867 | 71856 | 23.5 |
| OZO864 | 91078 | 0.5 |
| OZO866 | S9020317-1 | 14.2 |
| OZO869 | 71850 | 23.7 |
| OZO858 | 74311 | 0.05 |
| OZO868 | 74609 | 14.1 |
| OZO870 | WRK057103 | 5.2 |
| OZO863 | 71187 | 7.8 |
| OZO860 | 84032 | 14.5 |
| OZO857 | 107475 | 16.5 |



| Sample Bore IDb | Distance from |
|---|---|
| | |
| codea | coast (km) |
| | |
| OZO861 71219 | 0.5 |
| OZO856 91076 | 4.5 |
| OZO865 91029 | 2.9 |
| OZO862 71210 | 0.5 |
| OZO859 71190 | 7.8 |
| OZO867 71856 | 23.5 |
| OZO864 91078 | 0.5 |
| OZO866 S9020317-1 | 14.2 |
| OZO869 71850 | 23.7 |
| OZO858 74311 | 0.05 |
| OZO868 74609 | 14.1 |
| OZO870 WRK057103 5.2 | |
| OZO863 71187 | 7.8 |
| OZO860 84032 | 14.5 |
| OZO857 107475 | 16.5 |

**Fig. 11** Original table (Currell et al., 2013) columns(left) and vs columns in singlecamelot's output

## APPENDIX D: TABLES

**Table 3** Properties and errors recorded for each output.

| **Property/Error** | **Description** |
|---|---|
| Table type | Table layout. |
| Program | The program used for extraction. |
| Detected | Recorded as ‘Y’ or N’.<br>If table contents are completely mixed with surrounding text and cannot be differentiated, the table is considered undetected. |
| Columns/rows merged | Recorded as ‘Y’ or N’. |
| Content shifted | Recorded as ‘Y’ or N’. |
| Missing data | Recorded as ‘Y’ or N’. If the Table number, title or footnote are missing, they are not considered missing data. |
| Additional Comments | Comment on additional observations. |
| Table cells detected | Number of table cells detected. Table number, title and footnote are counted as table cells. |
| Non-table cells detected | Number of non-table cells detected. |
| Total cells detected | Total number of cells detected. |
| Noise percentage | (Number of non-table cells detected / Total number of cells detected) * 100. |
| Noise reduction (cleaning methods only) | Difference between the noise detected with camelot_scrape and the cleaning method / the noise detected with camelot_scrape. |

**Table 4** Error rates and noise reduction across different table types and extraction methods.

| **Table type** | **Sample size** | **Program** | **Undetected**<br>**%** | **Columns/rows merged**<br>**%** | **Contents shifted**<br>**%** | **Missing data**<br>**%** | **Noise detected**<br>**%** | **Mean noise reduction %** |
|---|---|---|---|---|---|---|---|---|
| Centred, spanning full page width, above text. | 30 | camelot_scrape.py | 13.33 | 3.33 | 3.33 | 13.33 | 4.56 | |
| | | clean_camelot.py | 16.67 | 3.33 | 3.33 | 30.00 | 2.89 | 13.56 |
| | | single_camelot.py | 20.00 | 10.00 | 30.00 | 43.33 | 2.32 | 16.69 |
| Centred, spanning full page width, below text. | 17 | camelot_scrape.py | 5.88 | 0.00 | 5.88 | 5.88 | 0.13 | |
| | | clean_camelot.py | 5.88 | 0.00 | 5.88 | 23.53 | 0.00 | 5.88 |
| | | single_camelot.py | 5.88 | 5.88 | 41.18 | 70.59 | 0.99 | 5.24 |
| Centred, uses the full-page width, no surrounding text. The table is landscape in a portrait page. | 16 | camelot_scrape.py | 0.00 | 0.00 | 6.25 | 12.50 | 0.34 | |
| | | clean_camelot.py | 6.25 | 0.00 | 6.25 | 18.75 | 0.23 | 6.44 |
| | | single_camelot.py | 68.75 | 0.00 | 6.25 | 75.00 | 0.00 | 6.25 |
| Embedded within column | 13 | camelot_scrape.py | 23.08 | 7.69 | 0.00 | 23.08 | 28.93 | |
| | | clean_camelot.py | 23.08 | 0.00 | 0.00 | 46.15 | 11.49 | 65.43 |
| | | single_camelot.py | 15.38 | 7.69 | 23.08 | 30.77 | 0.06 | 88.52 |

**Table 5** Processing time for each program

| **Program** | **Time taken / s** |
|---|---|
| camelot_scrape.py | 484.09 |
| clean_camelot.py | 531.82 |
| single_camelot.py | 840.05 |

## REFERENCES

ADv0rnik. (2022, October 2). *Answer to 'How to scrape data from sciencedirect'* [Online post]. Stack Overflow. https://stackoverflow.com/a/73928459

Ariga, A. (2016). *tabula-py: Simple wrapper for tabula-java, read tables from PDF into DataFrame* (Version 2.10.0) [Python]. https://github.com/chezou/tabula-py

Bansal, P. (2023). A Comparison of python libraries for PDF Data Extraction for text, images and tables. *Medium*. https://pradeepundefned.medium.com/a-comparison-of-python-libraries-for-pdf-data-extraction-for-text-images-and-tables-c75e5dbcfef8

Corrêa, A. S., & Zander, P.-O. (2017). Unleashing Tabular Content to Open Data: A Survey on PDF Table Extraction Methods and Tools. *Proceedings of the 18th Annual International Conference on Digital Government Research*, 54–63. https://doi.org/10.1145/3085228.3085278

Currell, M., Cendón, D. I., & Cheng, X. (2013). Analysis of environmental isotopes in groundwater to understand the response of a vulnerable coastal aquifer to pumping: Western Port Basin, south-eastern Australia. *Hydrogeology Journal*, *21*(7), 1413–1427. https://doi.org/10.1007/s10040-013-1017-9

Ghosh Chowdhury, A., Ben Ahmed, M., & Atzmueller, M. (2022). Towards Tabular Data Extraction From Richly-Structured Documents Using Supervised and Weakly-Supervised Learning. *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–4. https://doi.org/10.1109/ETFA52439.2022.9921455

Hammond, M. (2023). *pywin32: Python for Window Extensions* (Version 306) [Python; Microsoft :: Windows]. https://github.com/mhammond/pywin32

Huang, J., Chen, H., Yu, F., & Lu, W. (2024). From Detection to Application: Recent Advances in Understanding Scientific Tables and Figures. *ACM Computing Surveys*, *56*(10), 1–39. https://doi.org/10.1145/3657285

Jinghong, L., Koichi, O., Wen, G., & Shinobu, H. (2023). A Text Block Refinement Framework For Text Classification and Object Recognition From Academic Articles. *2023 International Conference on Innovations in Intelligent Systems and Applications (INISTA)*, 1–6. https://doi.org/10.1109/INISTA59065.2023.10310320

Mehta, V. (2021). *Camelot: PDF Table Extraction for Humans—Camelot 0.11.0 documentation*. https://camelot-py.readthedocs.io/en/master/user/how-it-works.html

Milosevic, N., Gregson, C., Hernandez, R., & Nenadic, G. (2016). Extracting Patient Data from Tables in Clinical Literature—Case Study on Extraction of BMI, Weight and Number of Patients: *Proceedings of the 9th International Joint Conference on Biomedical Engineering Systems and Technologies*, 223–228. https://doi.org/10.5220/0005660102230228

Mirdamadi, A., Hanbury, A., & Piroi, F. (2018). *Table Extraction for Information Retrieval* [Vienna University of Technology]. https://api.semanticscholar.org/CorpusID:252047450

OpenAI. (2023). *ChatGPT* (Version May 13, 2024) [Computer software]. OpenAI. https://chat.openai.com

Richardson, L. (2024). *beautifulsoup4: Screen-scraping library* (Version 4.12.3) [Python]. https://www.crummy.com/software/BeautifulSoup/bs4/

Vine, N. L., Zeigenfuse, M., & Rowan, M. (2019). Extracting Tables from Documents using Conditional Generative Adversarial Networks and Genetic Algorithms. *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–8. https://doi.org/10.1109/IJCNN.2019.8851886

Wiechork, K., & Charão, A. (2021). Automated Data Extraction from PDF Documents: Application to Large Sets of Educational Tests: *Proceedings of the 23rd International Conference on Enterprise Information Systems*, 359–366. https://doi.org/10.5220/0010524503590366

Wood, C., Cook, P. G., Harrington, G. A., Meredith, K., & Kipfer, R. (2014). Factors affecting carbon-14 activity of unsaturated zone CO2 and implications for groundwater dating. *Journal of Hydrology*, *519*, 465–475. https://doi.org/10.1016/j.jhydrol.2014.07.034