Spreadsheet Data Extractor (SDE): A Performance-Optimized, User-Centric Tool for Transforming Semi-Structured Excel Spreadsheets into Relational Data

ANONYMOUS AUTHOR(S)
SUBMISSION ID:

Organizations across various sectors frequently struggle to analyze and utilize semi-structured data derived from spreadsheets due to the lack of defined structure. This paper introduces the Spreadsheet Data Extractor (SDE), an open-source tool designed to convert semi-structured spreadsheet data into structured formats without requiring programming knowledge. Building upon previous work, we have enhanced the SDE with incremental loading of worksheets, accurate rendering of cell dimensions by parsing XML data, and performance optimizations to handle large datasets efficiently. We compare our tool with existing solutions and demonstrate its effectiveness through performance evaluations, highlighting its potential to facilitate efficient and reliable data extraction from diverse spreadsheet formats.

CCS Concepts: • **Applied computing** → **Spreadsheets**; • **Information systems** → **Data cleaning**; • **Software and its engineering** → **Extensible Markup Language (XML)**; • **Human-centered computing** → *Graphical user interfaces*; • **Theory of computation** → Data compression.

ACM Reference Format:
Anonymous Author(s). 2025. Spreadsheet Data Extractor (SDE): A Performance-Optimized, User-Centric Tool for Transforming Semi-Structured Excel Spreadsheets into Relational Data. In *Proceedings of 2025 International Conference on Management of Data (ACM PODS '25).* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/XXXXXXX.XXXXXXX

1 INTRODUCTION

Spreadsheets are ubiquitous tools used across various domains, including healthcare [3], nonprofit organizations [10, 12], finance, commerce, academia, and government [6]. Despite their widespread use, analyzing and utilizing data stored in spreadsheets poses significant challenges due to their semi-structured nature. Data in spreadsheets are often formatted for human readability, employing layouts with empty cells, merged cells, hierarchical headers, and multiple tables, which hinder machine readability and automated data processing.

While unstructured data in spreadsheets have advantages—such as an easily comprehensible hierarchy of metadata for humans—the same features complicate automated data extraction.

The objective of this work is to transform semi-structured spreadsheet data into machine-readable formats. Existing approaches often rely on heuristics, machine learning, or programming-by-example techniques, which may introduce errors or require significant Aufwand um diese Fehler zu finden und zu reparieren. To address these challenges, we present the Spreadsheet Data Extractor (SDE), a tool that enables users to define data hierarchies through cell selection without any programming knowledge.

ACM PODS '25, June 22–27, 2025, Berlin, Germany

ACM ISBN …$15.00
https://doi.org/10.1145/XXXXXXX.XXXXXXX

1.1 Contributions

Our contributions are as follows:

(1) We release the SDE under the open-source GNU General Public License v3.0, promoting community access and collaboration.
(2) We implement incremental loading of worksheets to enhance performance, allowing the tool to handle large Excel files efficiently.
(3) We accurately render row heights and column widths by parsing XML data, ensuring that the spreadsheet's visual representation closely matches that of Excel.
(4) We optimize the rendering engine to draw only the visible cells, significantly improving performance when dealing with large datasets.
(5) We integrate the selection hierarchy, worksheet view, and output preview into a unified interface, streamlining the user experience.

2 INTRODUCTION

Previous work by Alexander Aue, Norbert Röder, and Andrea Ackermann introduced a tool that facilitates data extraction from Excel files. [1] While effective, this solution faced performance issues and inaccuracies in rendering cell dimensions, limiting its usability with large and complex datasets. In this paper, we build upon their foundational work by releasing the software under the open-source GNU Public License Version 3 (GPLv3) and introducing several key enhancements. These improvements include implementing incremental loading of worksheets to enhance performance, accurately rendering row heights and column widths by parsing XML data, and optimizing the rendering engine to handle large datasets efficiently. Additionally, we have integrated the selection hierarchy and worksheet view into a unified interface to improve user experience. These contributions collectively address the limitations of the existing solution, making data extraction from complex spreadsheets more efficient and user-friendly.

3 RELATED WORK

The extraction of relational data from semi-structured documents, particularly spreadsheets, has garnered significant attention due to their ubiquitous use across domains such as business, government, and scientific research. Several frameworks and tools have been developed to address the challenges of converting flexible spreadsheet formats into normalized relational forms suitable for data analysis and integration. Notable among these are **DeExcelerator**, **XLIndy**, **FLASHRELATE**, **Senbazuru**, **TableSense** und den Ansatz von Aue et al. auf dessen arbeit wir aufbauen.

3.1 Aue et al.'s Converter

Aue et al. [1] developed a tool aimed at facilitating data extraction from Excel spreadsheets by utilizing the Dart 'excel' package to open '.xlsx' files. Users can select cells containing data and metadata to define the data hierarchy. However, this method encounters performance bottlenecks as the package requires loading the entire '.xlsx' file into memory before processing, leading to slow response times, especially with large files. Die Lösung nutzte das Dart-Package excel, um die .xlsx-Dateien zu öffnen. Dies war jedoch sehr langsam, da das Paket die gesamte .xlsx-Datei zunächst vollständig einliest. Wir haben daher eine eigene Funktionalität in Dart implementiert, die die Excel-Arbeitsblätter inkrementell lädt. Additionally, their solution calculates row heights and column widths based solely on cell content, disregarding the actual dimensions specified in the Excel file. This results in discrepancies between the tool's rendering and the original spreadsheet. The tool also renders all cells regardless of their visibility within the viewport, causing significant performance degradation when handling worksheets with numerous cells.

3.2 DeExcelerator

Eberius et al. [7] introduced **DeExcelerator**, a framework that transforms partially structured spreadsheets into first normal form relational tables using heuristic-based extraction phases. It addresses challenges such as table detection, metadata extraction, and layout normalization. While effective in automating normalization, its reliance on predefined heuristics limits adaptability to heterogeneous or unconventional spreadsheet formats, highlighting the need for more flexible approaches.

3.3 XLIndy

Koci et al. [8] developed **XLIndy**, an interactive Excel add-in with a Python-based machine learning backend. Unlike DeExcelerator's fully automated heuristic approach, XLIndy integrates machine learning techniques for layout inference and table recognition, enabling a more adaptable and accurate extraction process. XLIndy's interactive interface allows users to visually inspect extraction results, adjust configurations, and compare different extraction runs, facilitating iterative fine-tuning. Additionally, users can manually revise predicted layouts and tables, saving these revisions as annotations to improve classifier performance through (re-)training. This user-centric approach enhances the tool's flexibility, allowing it to accommodate diverse spreadsheet formats and user-specific requirements more effectively than purely heuristic-based systems.

3.4 FLASHRELATE

Barowy et al. [2] presented **FLASHRELATE**, an approach that empowers users to extract structured relational data from semi-structured spreadsheets without requiring programming expertise. FLASHRELATE introduces a domain-specific language, **FLARE**, which extends traditional regular expressions with spatial constraints to capture the geometric relationships inherent in spreadsheet layouts. Additionally, FLASHRELATE employs an algorithm that synthesizes FLARE programs from a small number of user-provided positive and negative examples, significantly simplifying the automated data extraction process.

FLASHRELATE distinguishes itself from both DeExcelerator and XLIndy by leveraging programming-by-example (PBE) techniques. While DeExcelerator relies on predefined heuristic rules and XLIndy incorporates machine learning models requiring user interaction for fine-tuning, FLASHRELATE allows non-expert users to define extraction patterns through intuitive examples. This approach lowers the barrier to entry for extracting relational data from complex spreadsheet encodings, making the tool accessible to a broader range of users.

3.5 Senbazuru

Chen et al. [4] introduced **Senbazuru**, a prototype Spreadsheet Database Management System (SSDBMS) designed to extract relational information from a large corpus of spreadsheets. Senbazuru addresses the critical issue of integrating data across multiple spreadsheets, which often lack explicit relational metadata, thereby hindering the use of traditional relational tools for data integration and analysis.

Senbazuru comprises three primary functional components:

(1) **Search**: Utilizing a textual search-and-rank interface, Senbazuru enables users to quickly locate relevant spreadsheets within a vast corpus. The search component indexes spreadsheets using Apache Lucene, allowing for efficient retrieval based on relevance to user queries.
(2) **Extract**: The extraction pipeline in Senbazuru consists of several stages:

- **Frame Finder**: Identifies data frame structures within spreadsheets using Conditional Random Fields (CRFs) to assign semantic labels to non-empty rows, effectively detecting rectangular value regions and associated attribute regions.
- **Hierarchy Extractor**: Recovers attribute hierarchies for both left and top attribute regions. This stage also incorporates a user-interactive repair interface, allowing users to manually correct extraction errors, which the system then generalizes to similar instances using probabilistic methods.
- **Tuple Builder and Relation Constructor**: Generates relational tuples from the extracted data frames and assembles these tuples into coherent relational tables by clustering attributes and recovering column labels using external schema repositories like Freebase and YAGO.

(3) **Query**: Supports basic relational operations such as selection and join on the extracted relational tables, enabling users to perform complex data analysis tasks without needing to write SQL queries.

Senbazuru's ability to handle hierarchical spreadsheets, where attributes may span multiple rows or columns without explicit labeling, sets it apart from earlier systems like DeExcelerator and XLIndy. By employing machine learning techniques and providing user-friendly repair interfaces, Senbazuru ensures high-quality extraction and facilitates the integration of spreadsheet data into relational databases.

3.6 TableSense

Dong et al. [5] developed **TableSense**, an end-to-end framework for spreadsheet table detection using Convolutional Neural Networks (CNNs). TableSense addresses the diversity of table structures and layouts by introducing a comprehensive cell featurization scheme, a Precise Bounding Box Regression (PBR) module for accurate boundary detection, and an active learning framework to efficiently build a robust training dataset.

While **DeExcelerator**, **XLIndy**, **FLASHRELATE**, and **Senbazuru** focus primarily on transforming spreadsheet data into relational forms through heuristic, machine learning, and programming-by-example approaches, **TableSense** specifically targets the accurate detection of table boundaries within spreadsheets using deep learning techniques. Unlike region-growth-based methods employed in commodity spreadsheet tools, which often fail on complex table layouts, TableSense achieves superior precision and recall by leveraging CNNs tailored for the unique characteristics of spreadsheet data. However, TableSense focuses on table detection and visualization, allowing users to generate diagrams from the detected tables but does not provide functionality for exporting the extracted data for further analysis.

3.7 Comparison and Positioning

While **DeExcelerator**, **XLIndy**, **FLASHRELATE**, **Senbazuru**, and **TableSense** each offer unique approaches to spreadsheet data extraction, they share certain limitations. Many of these tools are not readily accessible: **FLASHRELATE** and **TableSense** are proprietary, and **Senbazuru**, **XLIndy**, and **DeExcelerator** are discontinued projects with limited or no source code availability. In contrast, we contribute our spreadsheet data extractor under the GNU General Public License v3.0, allowing the community to access, use, and improve the tool freely.

Moreover, unlike the aforementioned tools that rely on heuristics, machine learning, or AI techniques—which can introduce errors requiring users to identify and correct—we adopt a user-centric approach that gives users full control over data selection and metadata hierarchy definition. While this requires more manual input, it eliminates the uncertainty and potential inaccuracies

associated with automated methods. To streamline the process and enhance efficiency, our tool includes user-friendly features such as the ability to duplicate hierarchies of columns and tables, and to move them over similar structures for reuse, reducing the need for repetitive configurations.

By combining the strengths of manual control with enhanced user interface features and performance optimizations, our tool offers a robust and accessible solution for extracting relational data from complex and visually intricate spreadsheets. These enhancements not only improve performance and accuracy but also elevate the overall user experience, making our tool a valuable asset for efficient and reliable data extraction from diverse spreadsheet formats.

4 METHODOLOGY

In this section, we detail the design and implementation of the Spreadsheet Data Extractor (SDE), emphasizing its user-centric approach and performance optimizations. The SDE enables users to transform semi-structured spreadsheet data into structured, machine-readable formats without requiring programming expertise. We achieve this through an intuitive interface that allows for cell selection and hierarchy definition, incremental loading of worksheets, accurate rendering of cell dimensions, and optimized performance for handling large datasets by incrementally loading of worksheets and dadurch dass wir nur solche Zellen rndern, die in der aktuellen ansicht auch sichtbar sind.

4.1 User-Centric Data Extraction

The core functionality of the SDE revolves around allowing users to select cells containing data and metadata to define a data hierarchy. This process is facilitated through a graphical interface that displays the spreadsheet and allows for intuitive selection and manipulation of the selection hierarchy.

4.1.1 Hierarchy Definition. Users can select individual cells or ranges of cells by clicking and pressing using shift-click for multi-selection. These selections represent either data or metadata.

The selected cells are organized into a hierarchical tree structure, where each node represents a data element, and child nodes represent nested data or metadata. This hierarchy defines how the data will be transformed into a structured format.

4.1.2 Reusability and Efficiency. To optimize the extraction process and reduce repetitive tasks, the SDE allows users to duplicate previously defined hierarchies and apply them to similar regions within the spreadsheet. This feature is particularly useful for spreadsheets with repeating structures, such as multiple tables with the same format.

4.2 Example Workflow

Consider a spreadsheet containing statistical forecasts of future nursing staff availability in Germany [11]. The SDE interface consists of three main components (Figure 1):

Hierarchy Panel (Top Left): Displays the hierarchy of cell selections, initially empty.

Spreadsheet View (Top Right): Shows the currently opened Excel worksheet for cell selection.

Output Preview (Bottom): Provides immediate feedback on the data extraction based on current selections.

4.2.1 Step-by-Step Extraction: The user adds a node to the hierarchy and selects the cell containing the metadata "Nursing Staff" (Figure **??**). Diese Zelle steht für eine Metainformation, welche die alle Zellen in diesem Arbeitsblatt gemeinsam haben. Daher sollte sie als erstes ausgewählt werden and should appear at the beginning of each row in the output CSV file.

Define Subtables:

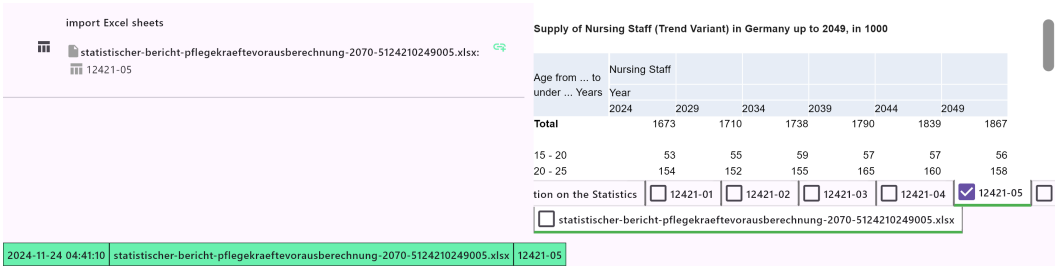


Fig. 1. The SDE Interface Overview.

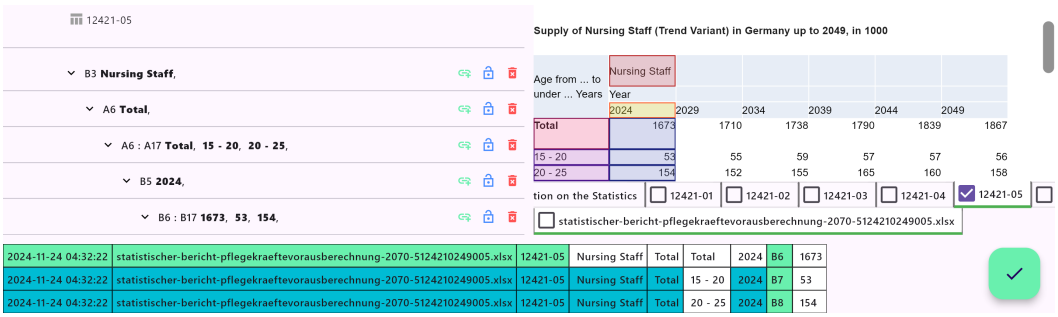


Fig. 2

Within this node, the user adds a child node and selects the cell "Total" which serves both as a table header and a row label. This selection represents the first subtable.

Select Row Labels:

The user adds another child node and selects the range of cells containing row labels ("Total," "15-20," "20-25") by clicking the first cell and shift-clicking the last cell.

Select Column Data:

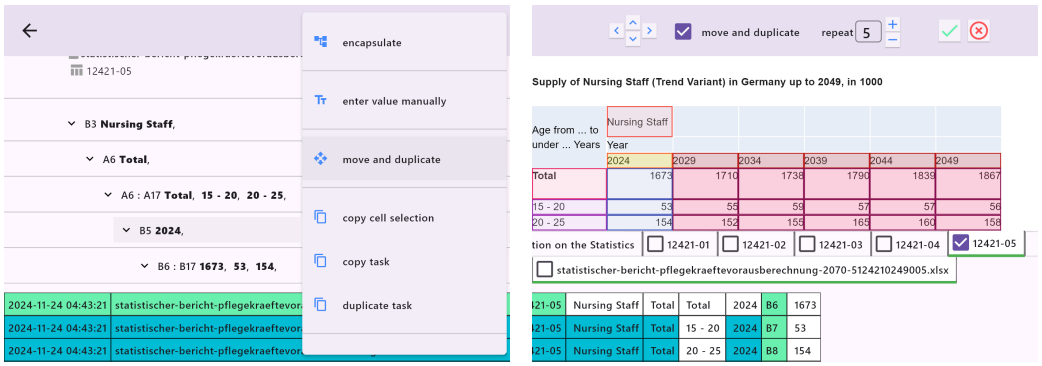
A child node is added under the row labels node, and the user selects the year "2024" and after that another child node is added under the year node and the user selects the corresponding data cells (e.g., "1673," "53," "154").

Die Hierachie ist nun gefüllt mit 5 Knoten von denen jedes bis auf den letzten einen eingebetteten Kind-Knoten haben. Im bereich oben rechts werden die selektierten Zellen mit unterschiedlichen Farben für jeden Knoten angezeigt.

Im unteren Bereich wird die Ausgabe angezeigt. Für jeden Knoten der als Kind eingebettet wird, erscheint in der ausgabe eine weitere Spalte. Sind in der Selektion nicht nur einzelnne Zellen angegeben, so erscheinen die werte der Liste auch in der Ausgabe als Werte in einer neuen Zeile.

Duplicate and Adjust Hierarchies:

To avoid repetitive manual entry for additional years, the user duplicates the hierarchy for "2024" and adjusts the cell selections to include data for subsequent years (e.g., "2025," "2026") using the "Move and Duplicate" feature. Dazu wird der Knoten der ersten Spalte "2024" selektiert und darauf rechtsgegklickt. Ein Popup öffnet sich in dem die Aktion "move and duplicate" auftaucht, die nun geklickt werden sollte, wie in Abbildung 3a zu sehen ist. This feature allows the user to shift cell selections horizontally or vertically and specify the number of repetitions, streamlining the process of capturing similar data structures.



(a) (b)

Fig. 3

Daraufhin öffnet sich in der App Bar oben rechts eine Reihe von schaltflächen, die erlauben die Zellenselektionen des Knotens sowie aller Kinderknoten zu verschieben wie in Abbildung 3b zu sehen ist. Durch das drücken auf die Schaltfläche zum verschieben der Selektion um eine Einheit nach rechts, ist die nächste Spalte ausgewählt, jedoch wird würde damit auch die Selektion der ersten Spalte wieder aufgehoben, da die Selektion verschoben wurde. Damit auch die erste Spalte erhalten bleibt, kann die Checkbox "move and duplicate" aktiviert werden. Dadurch wird die verschobene Selektion zusätzlich zur ursprungsselektion erstellt. Die Änderungen werden allerdings erst übernommen wenn auf den Akzeptieren button geklickt wird. Die nächsten SPalten könnten auf die gleich Art und Weise ebenfalls selektiert werden. Doch das geht auch schneller, denn statt nur einmal die selektion zu verschieben und gleichzeitig zu duplizieren, kann auch das eingabefeld "repeat" mit so vielen Wiederholungen gefüllt werden, wie es Spalten gibt. Mit der Eingabe der Nummer 5 wird damit die Selektion der ersten Spalte 5 mal um eine Einheit nach rechts verschoben und dabei bei jeden Schritt dupliziert.

Finalize Selection:

The user reviews the selections in the spreadsheet view, where each selection is highlighted in a different color corresponding to its node in the hierarchy. Erst nachdem der Nutzer die verschobenen und duplizierten selektionen in der Worksheet ansieht überprüft hat und den Akzeptieren button geklickt hat, werden die Knoten in der Hierachie wie gewünscht angelegt. Das Ergebnis dieser Operation ist in Abbildung 4 zu sehen.

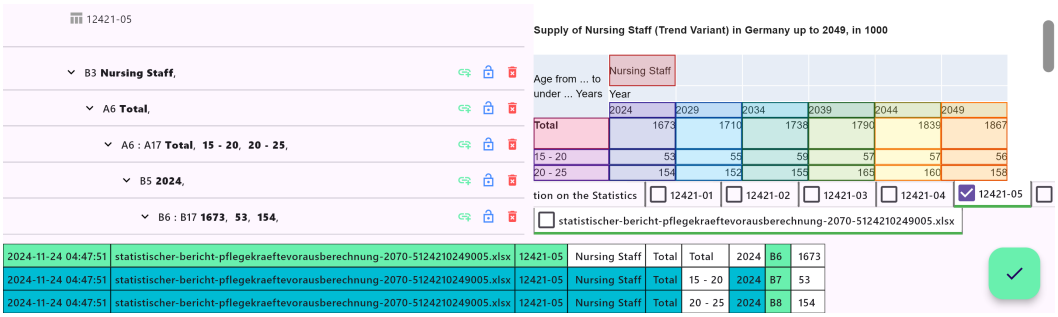


Fig. 4

Cross Product Transformation:

Damit ist die erste Tabelle vollständig selektiert. Der daraus resultierende Graph ist in Abbildung 5 zu sehen. Zur vereinfachung ist das Beispiel auf die ersten 3 Spalten und die ersten 3 Zeilen der Tabelle beschränkt.

Once the hierarchy is defined, the SDE applies a cross product operation to generate a relational format from the selection graph. This involves:

Node Duplication: Nodes with multiple edges (e.g., the row labels "Total," "15-20," "20-25") are duplicated so that each path in the graph represents a unique data row.

Value Replication: Nodes with single values (e.g., the year "2024") are replicated to match the number values of the associated node , ensuring alignment in the output.

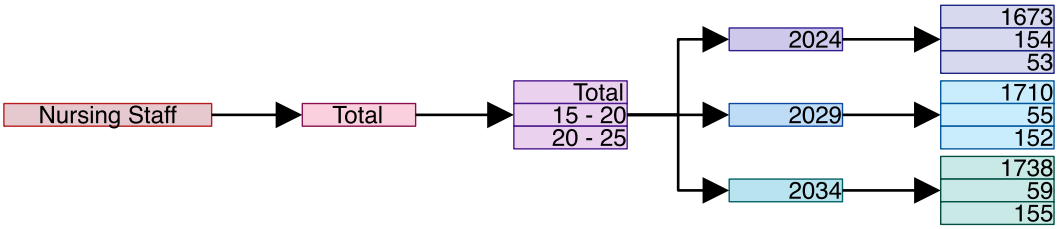


Fig. 5. Illustration of the cross product transformation.

Um aus dem Selektionsgraphen ein relationales Format zu generieren wird darauf das cross product angewendet. Was dabei genau passiert ist in Abbildung 6 zu sehen. Dort wo der Graph mehrere Kanten hat, was in dem Beispiel für den Knoten mit der Liste der Werte "Total", "15-20", "20-25" gilt, wird der Knoten so häufig dupliziert, wie es Kanten gibt, sodass jeder Knoten nur noch eine Kante zum nächsten Knoten hat. Darüber hinaus werden Knoten, die keine Liste von Werten enthalten, sondern nur einen Wert enthalten, in eine Liste umgewandelt und ihr Wert wird so häufig dupliziert, bis die anzahl dem Verknüpften Knoten entspricht. So wird in dem Beispiel der Knoten mit dem Wert "2024" in eine Liste mit den Werten "2024", "2024", "2024" umgewandelt, damit er die gleiche Anzahl an Werten hat wie der verknüpfte Knoten mit den Werten "1673", "154", "53".

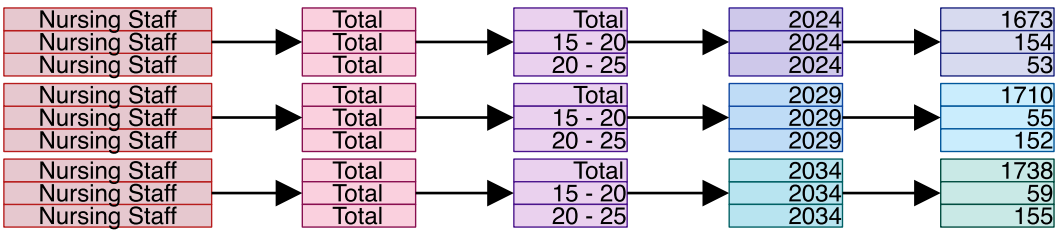


Fig. 6

Das gleiche was für die Spalten bereits so gut funktioniert hat, kann nun auch für die Untertabellen wiederholt werden, wie in Abbildung 7 zu sehen ist.

Durch das Selektieren des Knotens mit dem Wert "Total" und das drücken auf die Schaltfläche "move and duplicate" kann die Selektion der Untertabelle "Total" auf die anderen Untertabellen angewendet werden. Dazu muss die Tabelle um so viele Zeilen nach unten verschoben werden, bis sie mit der untertabelle überlappt Dabei gibt es nur ein kleines Problem. Denn zu den Unterknoten des Knotens "Total" gehören auch die Spaltenheader dazu. Würden diese Spaltenheader in der

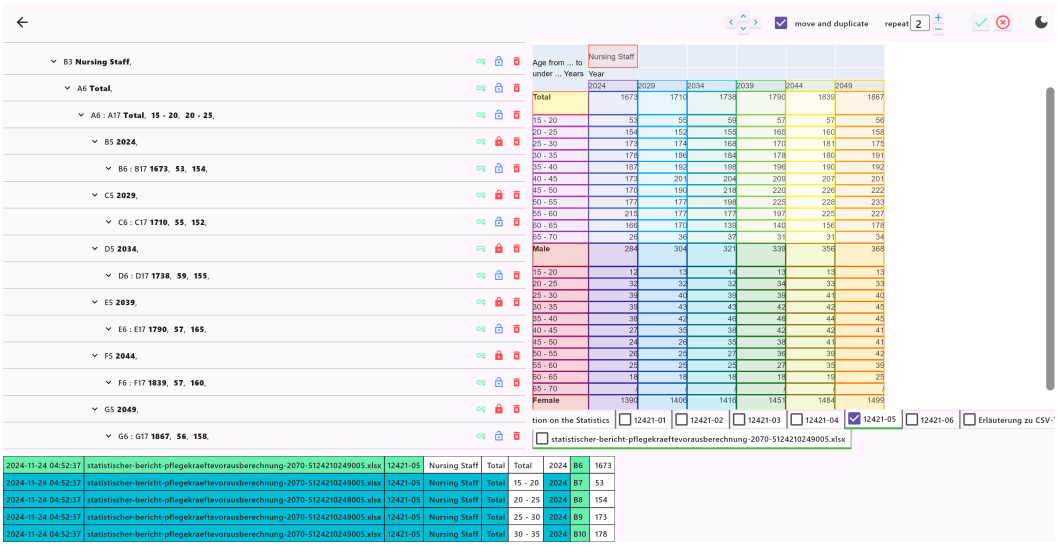


Fig. 7

Untertabelle darunter wiederholt werden, würde das verschieben der Selektionen nach unten unverändert funktionieren. Doch dadurch, dass diese Zellen in den untertabellen nicht wiederholt werden, , muss dafür gesorgt werden, dass sich die Zellen mit den Spaltenheadern daran gehindert werden sich durch die funkion zum verschieben und kopieren mit verschoben werden. Zu diesem Zweck können einzelne Knoten von der verschiebung ausgenommen werden, dazu muss lediglich das Vorhägeschlock button geklickt werden, wodurch die Zellselektion nicht mehr in der verschiebung enthalten ist. Sie bleibt starr an ihrem ursprungsort, egal ob die anderen Zellen verschoben werden. Desshalb werden nun die Knoten mit den SPaltenheadern, also die mit den Jahreszahlen 2024 bis 2049 identifiziert und selektiert und über den button mit dem Vorhängeschloss die Zellselektion die starr bleibt. Durch das verschieben nach unten und durch das damit verbundene duplizieren können nun sehr einfach die Zellselektionen nach unten verschoben und dupliziert werden. Durch das einsellen von 2 Wiederholungen werden alle untertabellen komplett ausgewählt.

4.3 Inkrementelles laden der Excel Dateien.

Wenn excel Dateien öffnet, dann lädt es zunächst die gesamte Datei mit allen Arbeitsblättern, bevor das erste Arbeitsblatt zur Darstellung gebracht wird. Bei Dateien, die sehr große Arbeitsblätter enthalten, kann dies einige Sekunden bis Minuten dauern. Den Benutzer, das extrahieren aus einer. Anzahl von Excel Dateien zu erleichtern, implementierten wir einen Mechanismus zum Inkrementellen Laden von Excel Arbeitsblättern

Eine exe Datei ist ein zip Archiv, welches einer Reihe an XML Dateien enthält, die die Workshits die Formatierungen und wiederkehrende Texte beschreiben. Unsere Lösung öffnet zunächst das Archiv und extrahiert nur die Metainformationen der einzelnen darin verpackten Dateien. Dieser Schritt dauert nur wenige millisekunden, die meiste Zeit geht stattdessen dabei drauf, die einzelnen eingebetteten Dateien wie die sharedStrings.xml Datei, die styles.xml Datei oder die einzelnen sheetX.xml Dateien zu, extrahieren. Da sowohl die Formatierungen aus der styles.xml Datei, als auch die wiederkehrenden String aus der sharedStrings.xml Datei im so gut wie jedem worksheet auftauchen, extrahieren wir diese auch gleich beim öffnen der Datei. Diese beiden Dateien sind im

vergleich zu den Worksheet Datei aber relativ klein und das entpacken ällt daher nicht stark ins gewicht. Die sheetX.xml Datei lassen wir aber zunächst unverpackt und speichern im Arbeitsspeicher lediglich die Verpackte Datei zwischen. Erst wenn eine beliebige Zelle in dem Worksheet von unserem Programm abgefragt wird, entweder weil der Benutzer den Worksheet durch einen Klick darauf zur anzeige gebracht hat, oder weil ein Unittest die Werte einer Excel Datei überprüft, wird im Hintergrund auch erst die Archivdatei des Arbeitsblattes extrahier. Auf diese Weise wartet der Nutzer bei öffnen einer Excel Datei nur einen Bruchteil einer Sekunde, bevor er mit dem Programm weiterarbeiten kann wohingegen Excel bei außergewöhnlich großen Dateien einige Sekunden benötigen kann, bis das erste Arbeitsblatt angezeigt werden kann.

Öffnet der Benutzer ein anderes Arbeitsblatt, so wird auch dieses erst beim Zugriff auf das Arbeitsblatt gepasst. Diese Vorgehensweise ermöglicht es uns jede Excel Datei nahezu sofort dem Benutzer zur Anzeige zu bringen und ihm keine Wartezeiten im Öffnen der.Excel Dateien zuzumuten.

4.4 Darstellung der Arbeitsblätter

Damit der Benutzer keine Schwierigkeiten hat, sich im Worksheet zurechtzufinden, legen wir großen wert darauf, die Arbeitsblätter so anzuzeigen, dass sie der Anzeige in Excel sehr nahe kommen.

4.4.1 Anzeige der Höhe und Breite. Unsere Lösung extrahiert die Informationen über die Spaltenbreiten aus dem *width* Attribut der *col*-knoten und die Zeilenhöhen aus dem *ht* Attribut der *row* Knoten aus der sheetX.xml Dateien. Dabei ist darauf zu achten, dass in Excel die Spaltenbreiten und Zeilenhöhen in Excel Einheiten gemessen werden. Diese müssen mit einem Faktor mit multipliziert werden, um die tatsächliche Höhe oder Breite in Pixeln zu erhalten. Wichtig dabei zu betrachten ist, dass dieser Faktor für die Spaltenbreiten und die Zeilenhöhen jeweils unterschiedlich ist. Damit die Spaltenbreite genauso bereit erscheint, wie sie in Excel erscheint, muss die angegebene Einheit in der XML Datei mit dem Faktor 7 multipliziert werden. Um die tatsächliche Zeilenhöher in Pixeln zu erreichen, muss dagegen der Faktor 4/3 mit dem Wert der in der Excel Datei angegeben wird, multipliziert werden. Die Werte haben wir durch eine Reihe von Tests ermittelt, indem wir Excel Dateien mit unterschiedlich großen Zellen erstellt haben und dann in der Worksheet Datei die Werte der Spaltenbreiten und Zeilenhöhen abgelesen haben. Nachdem wir nach einer Reihe solcher tests die Werte eine schätzung der Werte ermittelt haben, haben wir die Breiten und Höhen in den Arbeitsblätter modifiziert und die Excel Datei geöffnet um die Breiten und höhen zu messen und zu verifizieren, dass unsere Schätzung korrekt war.

Wie es zu den Unterschiedlichen Faktoren für die Zeilenhöhe und die Spaltenbreite kommt, konnten wir dagegen nicht durch Recherche herrausfinden.

Durch das anwenden dieser Faktoren können wir die Spaltenbreiten und Zeilenhöhen so genau wie möglich darstellen, wie sie in Excel dargestellt werden.

4.4.2 Formatierungen. Die Formatierungen der Zellen werden in der styles.xml Datei gespeichert. In dieser Datei werden die Formatierungen in Form von Stilen gespeichert, die dann in der sheetX.xml Datei referenziert werden. Dieser Index verweist auf den entsprechenden *xf*-Knoten im *cellStyleXfs* Knoten. Diese *xf*-Knoten enthalten wiederum Indeces der Schriftart, Hintergrundfarbe und Border über die Attribute *fontId*, *fillId* und *borderId*, welche auf die entprechenden *font*-Knoten im *fonts*-Knoten, die *fill*-Knoten im *fills*-Knoten und die *border*-Knoten im *borders*-Knoten verweisen. Diese Informationen nutzen wir, um die Zellen so genau wie möglich darzustellen, wie sie in Excel dargestellt werden.

4.4.3 Overflow. Genau wie in Excel auch achten wir darauf, dass Texte, die nicht in die Zelle hineinpassen durch einen Overflow in die anliegenden Zellen hineinragen. Doch das geschiet nur, wenn die anliegenden Zellen nicht selbst Werte enthalten. In dem Fall, dass die anliegenden Zellen Werte enthalten, wird der Text abgeschnitten. Dieses verhalten armen wir nach um die visuelle representation der Excel Datei so genau wie möglich nachzubilden.

Abbildung 8 zeigt, wie es aussieht, wenn man das nicht umsetzt. Der Text "Nursing Staff" ragt in die Zelle hinein, die rechts daneben liegt. In Excel würde der Text in die Zelle hineinragen, die rechts daneben liegt, wenn diese Zelle leer ist. In unserem Programm wird der Text abgeschnitten, wenn die Zelle rechts daneben einen Wert enthält.

To the table of c

Supply of Nurs

Age from ... to under ... Years	Nursing Staff					
	Year					
	2024	2029	2034	2039	2044	2049
Total	1673	1710	1738	1790	1839	1867

Fig. 8

4.5 Performance

Um hohe bildwiederholraten auch bei großen Arbeitsblätter zu gewährleisten, haben wir den Spreadsheet Data extractor so optimiert, dass er nur die Zellen Zeichnet , die auch tatsächlich sichtbar sind. Zu diesem zweck nutzen wird die seit Aug 17, 2023 erstmals erschiene package two_dimensional_scrollables. [9].

Da wir alle Spaltenbreiten und Zeilen Höhen aus den xml Dateien extrahieren, können wir diese Nutzen um zu berechnen, wie hoch und wie breit jede zelle ist und indem die breiten bzw höhen aufaddiert werden können wir auch berechnen an welcher koordinate sich die Zelle befindet. Die Koordinaten und breiten und höhen lkönnen dann genutzt werden um zu berechnen, welche Zellen aktuell sichtbar sind und alle anderen Zellen beim Zeichnen ignorieren. Dazu wird das aktuelle Scroll Offset in der x und y Achse berücksichtigt. Durch die Höhe und breite des Panels, welches das Excel Arbeitsblatt anzeigt, wird ein viewport aufgespannt, über den berechnet werden kann, welche Zellen darin sichtbar sind und welche außerhalb dieses viewports liegen. Welche Zellen aktuell sichtbar sind Indem die Spaltenbreiten so lange addiert werden, bis die Summe die linke Kante des Viewports erreicht. Alle Zellen links davon werden beim Zeichnen ignoriert. Weiterhin werden die Spaltenbreite Breiten weiter addiert, bis sie die rechte Kante des Viewports erreicht.Alles Zellen rechts davon werden beim Zeichnen ignoriert. Weiterhin werden die Zeilenhöhen entweder aus der Standard Höhe.Des Arbeitsblattes oder über die Benutzer definierten Höhen der Zeilen extrahiert und so lange aufaddiert, bis die Summe die obere Kante des Viewports erreicht.Alle Zellen, die oberhalb dieser Kante liegen, werden beim Zeichnen ignoriert.Die Zeilenhöhen werden weiter aufaddiert, bis sie die.Unsere Kante des Viewports erreicht.Nur diese Zellen werden gezeichnet. Alle Zellen unterhalb dieser Kante werden ignoriert.

das two_dimensional_scrollables package bietet eine Funktion, die es erlaubt, die sichtbaren Zellen zu berechnen und nur diese zu zeichnen. in Dieser Funktion werden das horizontalOffset und das verticalOffset und die viewportWidth und die viewportHeight als Parameter übergeben, die die Position des Viewports in der x und y Achse beschreiben und die genutzt werden kann um daran die sichtbaren zellen zu berechnen. Der Algorithmus zum berechnen der sichtbaren Zellen ist in Algorithmus 1 dargestellt.

Algorithm 1 Layout Spreadsheet Cells in Grid

Initialize Indices
  leadingColumnIndex ← column index based on *horizontalOffset*
  leadingRowIndex ← row index based on the *verticalOffset*
  trailingColumnIndex ← column index based on *horizontalOffset* + the *viewportWidth*
  trailingRowIndex ← row index based on the *verticalOffset* + the *viewportHeight*
Calculate Offsets
  leadingColumnOffset ← sum of widths from the first column to *leadingColumnIndex*
  leadingRowOffset ← sum of heights from the first row to *leadingRowIndex*
  horizontalLayoutOffset ← *leadingColumnOffset* − *horizontalOffset*
for each *columnIndex* from *leadingColumnIndex* to *trailingColumnIndex* **do**
    verticalLayoutOffset ← *leadingRowOffset* − *verticalOffset*
  for each *rowIndex* from *leadingRowIndex* to *trailingRowIndex* **do**
      child ← build the child for the *columnIndex* and *rowIndex* or obtain the cached one
      layout the *child* using the current *horizontalLayoutOffset* and *verticalLayoutOffset*
    if the row for *rowIndex* exists in the worksheet row definitions **then**
        verticalLayoutOffset ← *verticalLayoutOffset* + height of the row for *rowIndex*
    else
        verticalLayoutOffset ← *verticalLayoutOffset* + *defaultRowHeight*
    end if
  end for
    columnWidth ← width of the column for *columnIndex*
    horizontalLayoutOffset ← *horizontalLayoutOffset* + *columnWidth*
end for

5 EVALUATION

Der Ansatzes des Spreadsheet Data Extractors aus sich auf dem Converter von Alexander Aue et al. auf dem wier aufbauen nicht verändert. Die effektivität dieses Ansatzen wurde bereits untetsucht. Sie haben die Extraktion von Daten aus über 500 Excel Dateien evaluiert. Die Zeit, die für jede Datei benötigt wurde, wurde aus einer Stichprobe von 331 verarbeiteten Excel Dateien bestimmt, die 3.093 Arbeitsblätter umfassen. Im Durchschnitt benötigten die studentischen Hilfskräfte 15 Minuten pro Datei und 95 Sekunden pro Arbeitsblatt.

Wir konzentrieren uns auf die verbesserung der Benutzererfahrung und die Optimierung der Leistung des Spreadsheet Data Extractors. Die Benutzererfahrung durch die Darstellung der Excel-Arbeitsblätter ähnlich wie sie auch in Excel dargestellt werden vereinfacht und die anzahl der nötigen klickts durch die vereinigung der Benutzeroberflächen für die Selektionshierachie, dem Worksheet-View und der Vorschau der Ausgabe in einem View verringert. Die Leistung wurde durch das inkrementelle laden der Excel Dateien und das nur zeichnen der sichtbaren Zellen verbessert.

5.1 beschleunigung beim öffnen der Datei

Zum evaluieren der beschleunigung beim öffnen der Datei haben wir eine Reihe von Tests durchgeführt. Wir haben den gesamten Datensatz an Exceldateien von Destatis heruntergeladen und die größte Datei identifiziert. Mithilfe eines VBA Scriptes haben wir diese Exceldatei geöffnet und Werte in einem Arbeitsblatt ausgelesen. Dieser Test wurde 10 mal wiederholt und die Zeit gemessen, die benötigt wurde, um die Datei zu öffnen und die Werte auszulesen. Ein Equivalent dieses Scriptes haben wir als Unit Test geschrieben welcher dieselbe Datei mit der für den Spreadsheet Data Extrator implementierten Funktionen öffnet und dieselben Zellen aussliest Die Ergenisse dieser Laufzeiten sind in Abbildung putput zu sehen.

Der Spreadsheet Data Extractor öffnete das Arbeitsblatt im median in 120 Millisekunden, im Durchschnitt 178 Millisekunden und im ersten durchlauf 668 Millisekunden, was durduch sein kann, weil die Datei noch nicht im Cache war und erst von der Festplatte geladen werden musste.

Excel öffnete das Arbeitsblatt im median in 40 und 281 Millisekunden, im Durchschnitt 41 Sekunden und 138 Millisekunden.

Das dart package brauchte im ersten durchlauf 13 Minuten und 15 Sekunden zum öffnen des Arbeitsblattes. Die 9 anderen Durchläufe konnten nicht abgeschlossen werden da während des zweiten durchlaufs eine Out of Memory Exception geworfen wurde.

Damit öffnete der Spreadsheet Data Extractor das Arbeitsblatt over two orders of magnitude faster than Excel and nearly four orders of magnitude faster than the excel package used in prior work.

Die Ergebnisse sind in 9 zu sehen.

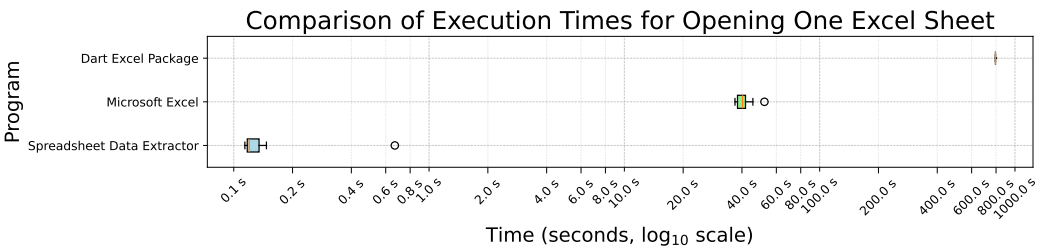


Fig. 9

6 OUTLOOK

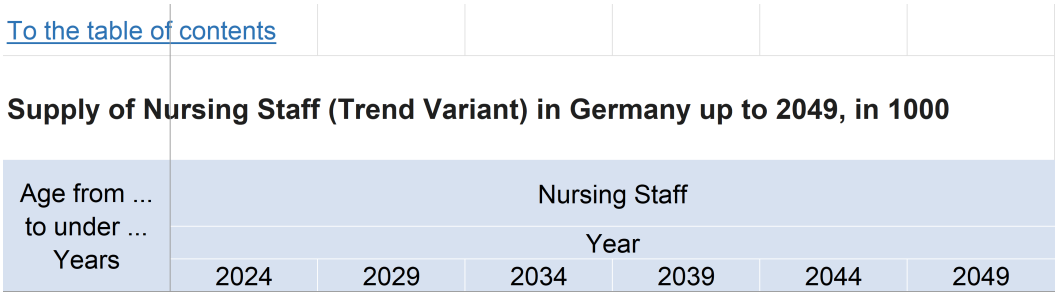
Wir haben vor den Spreadsheet Data Extractor weiter zu verbessern. Dazu gehört die Implementierung von Features, die die Benutzererfahrung weiter verbessern wie zum Beispiel die korrekte Darstellung von Texten für die in Excel das horizontale Text Alignment "Center Across Selection" verwendet wurde.

Paralel soll das Tool an weiteren Daten erprobt werden, um die Effektivität des Tools zu evaluieren.

6.1 Center Across Selection

AUch wenn wir uns mühe geben die Anzeige der Excel Dateien so genau wie möglich nachzubilden, gibt es immer noch einige Unterschiede zwischen der Anzeige in Excel und der Anzeige in unserem Spreadsheet Data Extractor. Die Unterschiede sind unbeabsichtigt und könnten behoben werden. Einer solcher Unterschiede ist in Abbildung 10 zu sehen. In Excel werden die Text "Nursing Staff" und "Year" über den gesamten Spaltenkopf hinweg zentriert angezeigt, während er in unserem

Spreadsheet Data Extractor linksbündig angezeigt wird. Dieser Unterschied ist darauf zurückzuführen, dass wir zwar merged cells parsen, die für gewöhnlich für das zentrieren solcher texte über mehrere zellen hinweg verwendet werden, doch in diesem Arbeitsblatt wurden keine merged cells verwendet um diese visuelle representation herzustellen. Stattdessen hat die erste Zelle in der styles.xml datei zugeordneten xml knoten alignment für das attribut horizontal den Wert centerContinuous. Bisherige Memühungen herrauszufinden über welche zellenkoordinaten sich diese horizontale zentrierung erstreckt sind jedoch bisher gescheitert. Weitere tests oder recherche ist notwendig um herrauszufinden, wie solche zentrierten Texte in der xml struktur der excel datei gespeichert sind.


To the table of contents

Supply of Nursing Staff (Trend Variant) in Germany up to 2049, in 1000

Age from ... to under ... Years	Nursing Staff					
	Year					
	2024	2029	2034	2039	2044	2049

Fig. 10

6.2 Evaluation an Real-World Daten

Bisher wurde die Basisversion des Tools an einem Datensatz des Agricultural Structure Survey on land use and livestock in Germany für 2020 erprobt. Interessant wäre nun die neue Version des Tools an den Daten vor 2020 zu testen, um zu sehen, ob die Verbesserungen, die wir vorgenommen haben, auch die Effektivität des Tools verbessern. Dazu können wir auf die in den ausgabe csv Dateien dokumentierten Timestamps zurückgreifen, um sie dann bei einem erneuten test mit den neuen timestamps zu vergleichen um so herrauszufinden ob die studentischen Hilfskräfte schneller mit dem neuen Tool arbeiten können.

7 CONCLUSION

In this paper, we presented the Spreadsheet Data Extractor, a tool that allows users to extract relational data from complex Excel files. By adopting a user-centric approach that gives users full control over data selection and metadata hierarchy definition, we provide a robust and accessible solution for data extraction. Our tool offers user-friendly features such as the ability to duplicate hierarchies of columns and tables, and to move them over similar structures for reuse, reducing the need for repetitive configurations. We also contribute our tool under the GNU General Public License v3.0, allowing the community to access, use, and improve the tool freely. By combining the strengths of manual control with enhanced user interface features and performance optimizations, our tool offers a valuable asset for efficient and reliable data extraction from diverse spreadsheet formats.

REFERENCES

[1] Andrea Ackermann Alexander Aue. 2024. Converting data organised for visual perception into machine-readable formats. In *44. GIL-Jahrestagung, Biodiversität fördern durch digitale Landwirtschaft*. Gesellschaft für Informatik eV, 179–184.

[2] Daniel W Barowy, Sumit Gulwani, Ted Hart, and Benjamin Zorn. 2015. FlashRelate: extracting relational data from semi-structured spreadsheets using examples. *ACM SIGPLAN Notices* 50, 6 (2015), 218–228.

[3] D.J. Berndt, J.W. Fisher, A.R. Hevner, and J. Studnicki. 2001. Healthcare data warehousing and quality assurance. *Computer* 34, 12 (2001), 56–65. https://doi.org/10.1109/2.970578

[4] Zhe Chen, Michael Cafarella, Jun Chen, Daniel Prevo, and Junfeng Zhuang. 2013. Senbazuru: A prototype spreadsheet database management system. *Proceedings of the VLDB Endowment* 6, 12, 1202–1205.

[5] Haoyu Dong, Shijie Liu, Shi Han, Zhouyu Fu, and Dongmei Zhang. 2019. Tablesense: Spreadsheet table detection with convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 69–76.

[6] Angus Dunn. 2010. Spreadsheets-the Good, the Bad and the Downright Ugly. *arXiv preprint arXiv:1009.5705* (2010).

[7] Julian Eberius, Christoper Werner, Maik Thiele, Katrin Braunschweig, Lars Dannecker, and Wolfgang Lehner. 2013. DeExcelerator: a framework for extracting relational data from partially structured documents. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2477–2480.

[8] Elvis Koci, Dana Kuban, Nico Luettig, Dominik Olwig, Maik Thiele, Julius Gonsior, Wolfgang Lehner, and Oscar Romero. 2019. Xlindy: Interactive recognition and information extraction in spreadsheets. In *Proceedings of the ACM Symposium on Document Engineering 2019*. 1–4.

[9] Kate Lovett. 2023. two_dimensional_scrollables package - Commit 4c16f3e. https://github.com/flutter/packages/commit/4c16f3ef40333aa0aebe8a1e46ef7b9fef9a1c1f Accessed: 2023-08-17.

[10] Gursharan Singh, Leah Findlater, Kentaro Toyama, Scott Helmer, Rikin Gandhi, and Ravin Balakrishnan. 2009. Numeric paper forms for NGOs. In *2009 International Conference on Information and Communication Technologies and Development (ICTD)*. IEEE, 406–416.

[11] Statistisches Bundesamt (Destatis). 2024. *Statistischer Bericht - Pflegekräftevorausberechnung - 2024 bis 2070.* Technical Report. Statistisches Bundesamt, Wiesbaden, Germany. https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Bevoelkerung/Bevoelkerungsvorausberechnung/Publikationen/Downloads-Vorausberechnung/statistischer-bericht-pflegekraeftevorausberechnung-2070-5124210249005.html Statistical Report - Projection of Nursing Staff - 2024 to 2070.

[12] Hannah West and Gina Green. 2008. Because excel will mind me! the state of constituent data management in small nonprofit organizations. In *Proceedings of the Fourteenth Americas Conference on Information Systems*. Association for Information Systems, AIS Electronic Library (AISeL). https://aisel.aisnet.org/amcis2008/336

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009