scientific data

Check for updates

OPEN

DATA DESCRIPTOR

ENTRANT: A Large Financial Dataset for Table Understanding

Elias Zavitsanos ✉, Dimitris Mavroeidis, Eirini Spyropoulou, Manos Fergadiotis & Georgios Paliouras

Tabular data is a way to structure, organize, and present information conveniently and effectively. Real-world tables present data in two dimensions by arranging cells in matrices that summarize information and facilitate side-by-side comparisons. Recent research efforts aim to train large models to understand structured tables, a process that enables knowledge transfer in various downstream tasks. Model pre-training, though, requires large datasets, conveniently formatted to reflect cell and table characteristics. This paper presents ENTRANT, a financial dataset that comprises millions of tables, which are transformed to reflect cell attributes, as well as positional and hierarchical information. Hence, they facilitate, among other things, pre-training tasks for table understanding with deep learning methods. The dataset provides table and cell information along with the corresponding metadata in a machine-readable format. We have automated all data processing and curation and technically validated the dataset through unit testing of high code coverage. Finally, we demonstrate the use of the dataset in a pre-training task of a state-of-the-art model, which we use for downstream cell classification.

Background & Summary

Tables constitute fundamental structures to organize and represent data. They are widely used in web pages, spreadsheets, reports, and other types of documents. Tabular data are usually arranged in cells of bi-dimensional matrices, with stylistic formatting that effectively reflects important regions and data in the table. Tables are typically made to ease readability by humans; they are usually assumed to be read left-to-right and top-to-bottom, while related items are assumed to appear nearby. They present information using spacing, lines, rules, font style, weight, and indentation. In order to represent data as effectively as possible, real-world tables often imply hierarchical structures shaped by merged cells and indents to organize and summarize the data in a compact way that facilitates quick lookup and side-by-side comparisons. In the simplest case, a table can be organized in a horizontal or vertical relational form, as shown in Fig. 1(a), while in a more complex case, it may take the form of a matrix. The latter case is depicted in Fig. 1(b), where two hierarchies are implied simultaneously. A top-down hierarchy of three levels: a root node whose children are "Incidence" and "Mortality" at level 1, and their children of the last two are "Males" and "Females" at level 2. Likewise, there is a left-to-right hierarchy also of three levels: a root node whose children are the cells in bold text at level 1, and their corresponding children at level 2, such as "Melanoma of skin", "Non-melanoma skin cancer" and so on.

Tables are used to store valuable information that can be used, in turn, to answer questions, populate knowledge bases, generate natural language descriptions, and various other applications. Most of the time, though, tables do not include machine-readable semantics for the automated processing of the information they contain. This lack of semantics is a challenge that table understanding aims to address. In particular, table understanding aims at recovering the missing semantics that enables the extraction of facts from tables and facilitates interoperability, knowledge extraction, and information processing. This topic covers many challenging tasks, from table detection in documents and relational data extraction[1] to semantic table interpretation and representation[2]. In this work, we are primarily interested in the part of table understanding that concerns table representation and interpretation.

Increased attention towards table representation from the research community has led to various approaches to table understanding and its subtasks. In particular, several approaches have been proposed to solve structure-based tasks, such as table segmentation, cell role prediction, block detection, and join identification. Moreover, approaches focusing on knowledge alignment tasks deal with semantic typing and modeling, schema mapping, entity linking, knowledge extraction and augmentation. Finally, several downstream tasks concern

Institute of Informatics and Telecommunications, National Centre for Scientific Research "Demokritos", Aghia Paraskevi, 15341, Greece. ✉e-mail: izavits@iit.demokritos.gr

SQL term	Relational database term	Description
Row	*Tuple* or *record*	A data set representing a single item
Column	***Attribute*** or ***field***	A labeled element of a tuple, e.g. "Address" or "Date of birth"
Table	*Relation* or *Base relvar*	A set of tuples sharing the same attributes; a set of columns and rows
View or *result set*	***Derived relvar***	Any set of tuples; a data report from the RDBMS in response to a query

(a)

Cancer statistics in 2018	Incidence		Mortality	
	Males	Females	Males	Females
Skin				
Melanoma of skin	150,698	137,025	34,831	25,881
Non-melanoma skin cancer	637,733	404,323	38,345	26,810
Urinary tract				
Kidney and renal pelvis	254,507	148,755	113,822	61,276
Bladder	424,082	125,311	148,270	51,652
Respiratory system				
Larynx	154,977	22,445	81,806	12,965
Trachea, bronchus and lung	1,368,524	725,352	1,184,947	576,060
Mesothelioma	21,662	8,781	18,332	7,244

(b)

Fig. 1 Examples of structured tables: (**a**) relational web table, (**b**) spreadsheet table.

cell type classification, table type classification, formula prediction, summarization, question answering, and validation[3]. In other words, table understanding is a significant area of research with several constituent tasks. The understanding of table structure is a foundational step toward knowledge-centric tabular information processing. In recent years, the academic community has proposed several structural models in this direction, from traditional exploitation of cell-level features to models that incorporate contextual information and use deep learning.

Early attempts in the field started with the exploitation of cell-level features. An immediate extension was to use sequence-based features[4] integrating table content and layout features, using conditional random fields to find table cell boundaries, to classify cells as data or labels, and to associate data cells with their corresponding label cells. Using traditional supervised learning methods, Koci *et al.*[5] rely on content, stylistic, font, and spatial features to infer the layout of tables. This task is often referred to as table-type classification.

More recent approaches in table representation learning rely on cell and table embeddings to capture the underlying semantics. Initial approaches have used continuous bag-of-words and skip-gram models to learn cell embeddings[6]. Table2Vec[7], among the popular approaches, uses skip-gram neural networks to train word embeddings, aiming at capturing the semantics of table cells. Moreover, TabNet[8] relies on LSTMs to learn token embeddings, considering at the same time spatial information of the table, and more recently, Gunther *et al.*[9] proposed an embedding technique to be pre-trained directly on a large web table corpus to learn table-level embeddings.

As already mentioned, tables usually arrange similar items in nearby groups. Based on that, and since the value of a cell may be related to the neighborhood of that cell, many approaches rely on both semantic and spatial information and try to capture them jointly. In this direction, several neural network architectures have been proposed. For instance, CNNs have been used to model web and spreadsheet tables[10,11], RNNs and LSTMs have been employed to capture the order of rows and columns[6,8,12,13], and CNNs and RNNs have been proposed for column type classification[14]. Finally, besides recursive and convolutional network architectures, graph neural networks have also been explored for table understanding and particularly for question answering[15–17].

The most recent work on table understanding, though, relies heavily on table pre-training, which is actively and widely studied, aiming to build encoders suitable for structured tabular data and natural language context. In this direction, TaBERT[18] aims to recover column names and cell values from unmasked table cells. TaPas[19], on the other hand, relies on conventional Masked Language Modeling (MLM) to pre-train a model that provides a basic understanding of tables that can be used in downstream table question-answering tasks. STRUG[20] uses different pre-training objectives and focuses on explicitly learning the alignment between text and tables via a classification task, which improves schema link performance in downstream text-to-SQL tasks. In addition, TURL[21] learns representations from relational tables to enhance table knowledge matching and augmentation.

Finally, among the most recent approaches that are also closely related since they use the same pre-training data are TUTA[22] and FORTAP[23]. TUTA is a transformer-based method for understanding general table structure, which pre-trains on general tables that may contain hierarchical structures. The pre-trained model can be used as a basis for table question answering, formulae prediction, cell type classification, table type classification, and other downstream tasks. FORTAP, on the other hand, has been suggested as a method to fill the gap in inner-table numeric reasoning capabilities, which are essential for downstream tasks that involve calculations and reference resolution. For this reason, FORTAP needs to pre-train on data containing spreadsheet formulae, among others.

The increased interest of the research community in such large models for table understanding requires the use of large datasets that comprise millions of tables. These tables should be represented in a convenient and machine-readable form. The most widely used datasets are based on WikiTables[24]. For instance, TURL[21] relies on a subset of WikiTables containing 1.6 million tables, while TaPas[19] pre-trains on another subset of 6.2 million tables. WikiTables has also been used in combination with other sources. StruBERT[25], for instance, has been pre-trained on a dataset comprising data from WikiTables and data from a subset of PubMed Central (PMC)[26]. TABBIE[27] and TaBERT[18] used Wikipedia tables and tables from Common Crawl (https://commoncrawl.org/) to compile a pre-training dataset of 26.6 million tables.

Moving to larger pre-training datasets, Gunther *et al.*[9] use the DWTC Web Table corpus[28], comprising 125 million tables. Finally, TUTA[22] and FORTAP[23] combine web tables from Wikipedia with the WDC WebTable corpus[29] and web-crawled spreadsheets to compile a final dataset of 57.9 million tables. The latter spreadsheets dataset is

among the very few financial data used in the domain, together with the dataset used in TAT-QA[30], which is particularly small and contains 20K financial tables (downloaded from https://www.annualreports.com/).

Although there are publicly available datasets that can be used for pre-training, they mainly come from general-purpose sources, such as Wikipedia, Common Crawl, and Web data (WDC), that additionally require much pre-processing to extract the tables and the corresponding cell features in a machine-ready form that large models can easily consume. Moreover, datasets with tables from spreadsheets that contain both textual and numerical values are hard to find. For instance, the spreadsheets used in TUTA and FORTAP are not publicly available. On the other hand, the financial tables used in TAT-QA constitute a relatively small dataset for pre-training purposes. In order to fill this gap, we compile a large dataset that we call ENTRANT, consisting of millions of financial tables with both textual and numerical information from the EDGAR database, which provides free public access to corporate information. In addition, we transform the raw data into a convenient format that can be consumed by state-of-the-art models and provides both table and cell characteristics, including several cell attributes, positional information, and hierarchical information. Although the primary aim of this paper is to introduce a large dataset to enhance model pre-training, this work also contributes the following:

- We identify additional usages of the data that enable the application of machine learning methods in financial use cases.
- We release the code to encourage the augmentation of the dataset with financial tables from EDGAR.
- We enable the automated transformation of other corporate spreadsheets to the machine-readable format by releasing the code scripts that transform excel files to the proposed format.

In what follows, we describe our methodology for data gathering and transformation. Next, we present the data structure and provide an overview of the data files and their format. Last, we provide the analysis we followed to support and validate the dataset's technical quality. As an additional verification, we demonstrate the usage of the compiled dataset for pre-training a state-of-the-art model that we later use to experiment in a downstream classification task.

Methods

The data construction process comprised two main phases. The first concerned the raw data gathering, and the second the data transformation to produce a machine-readable format ready to be consumed by large machine learning architectures. The following sections describe in detail the steps followed in each phase.

Raw data gathering. At a high level, the dataset constitutes a collection of financial statements and disclosures in tabular format. Although this information is freely available for public companies, gathering large amounts of tabular data from the public statements for training is not practical. For this reason, we crawl the Electronic Data Gathering, Analysis, and Retrieval system (EDGAR) (https://www.sec.gov/edgar/about). EDGAR is the primary system for companies submitting reports, statements, and documents under the US Securities and Exchange Commission (SEC) (https://www.sec.gov/). It contains millions of company and individual filings, ensuring transparency and fairness in the securities markets and providing benefits for investors, corporations, and researchers. According to EDGAR, the system processes a few thousand filings daily and accommodates 40 thousand new files per year on average. For these reasons, it constitutes an ideal source of financial information that is updated constantly and contains various types of spreadsheet data, depending on the type of filing.

Regarding accessing the EDGAR data, as stated in the access policy, anyone can access and download the company information for free or query it through a variety of EDGAR public services (https://www.sec.gov/os/accessing-edgar-data). To ensure that everyone has equitable access to EDGAR we use efficient scripting to guarantee fair access by (a) limiting the maximum request rate to ten requests per second and (b) declaring the *user agent* in the request headers, as per EDGAR instructions.

EDGAR contains filings from the '90s up to this day. Many systems, standards, regulations, and filing requirements have changed during this period. Historical filings from the '90s and early '00s are available in plain text or HTML, which makes it difficult or inefficient to extract the financial tables robustly. Such an endeavor would require enormous human effort to ensure that all tables have been extracted correctly and that each table cell contains the correct data. In 2009, SEC adopted XBRL (https://www.xbrl.org/), an open international standard for digital business reporting. XBRL provides a language in which reporting terms can be authoritatively defined. In other words, XBRL delivers human-readable financial statements in a machine-readable, structured data format. Most importantly, publicly traded companies are required to create financial statements and assign an XBRL tag to every number, table, accounting policy, statement, and note. Hence, XBRL largely standardized the filing format, and nowadays, financial reports are structured, and financial statements appear inside not only HTML reports but also as separate spreadsheets, annotated and ready to be processed in an automated way.

For the compilation of our dataset, we focus on reports that are filed in XBRL format, ensuring that the spreadsheets we crawl are free from inconsistencies and that they can be parsed and processed by programming libraries in an automated way. The data gathering phase consists of two main steps; (a) checking for valid XBRL representations to filter out reports that are not appropriately tagged, and (b) crawling the reports from EDGAR, as shown in Fig. 2.

In particular, each company in EDGAR is uniquely identified by a Central Index Key (CIK), which is used in the SEC computer systems to identify corporations and individuals, who have filed a disclosure with the SEC. EDGAR uses the CIK to construct the Accession number, a unique identifier assigned automatically to an accepted submission by EDGAR. With each company's Accession number, we can construct the API endpoint of EDGAR to request the company filings (Listing 1 provides an example of an endpoint). If a company has filed a report that is not XBRL tagged, the report is filtered out as the corresponding spreadsheets cannot be retrieved robustly.

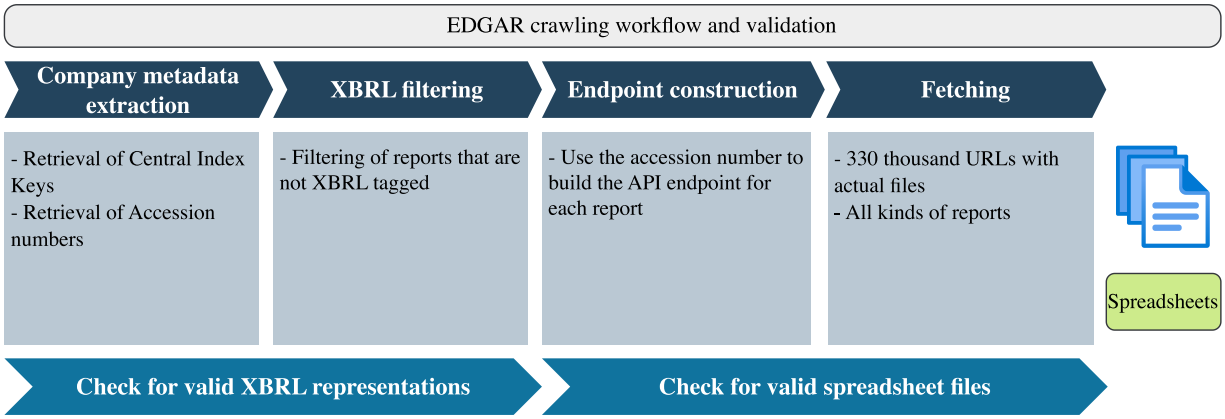

Fig. 2 EDGAR crawling workflow. Each company is uniquely identified by a central index key and an accession number that is used to build an endpoint for retrieving the report. Each endpoint is used to get the company spreadsheets, as long as the filing is in XBRL format.

```
"https://www.sec.gov/Archives/edgar/data/" + \
                ['cik_for_url'] + "/" +\
                ['accession_number_for_url'] + \
                '/Financial_Report.xlsx'
```

Listing 1 Example API endpoint for fetching EDGAR spreadsheets.

To ensure that most retrieved reports conform to XBRL, we crawl EDGAR for reports filed from 2013 to 2021. We stopped at 2021 due to the use of the data for our research on material misstatement detection[31]. Material misstatement can take - on average- two-to-three years to be discovered[31,32]. Using this cutoff, we are confident that we miss only a few misstatements and we minimize the risk of accounting inconsistencies. Under these assumptions, we generated 330 thousand API endpoints with actual files. These include annual and quarterly financial reports, current reports, registration statements, and more.

SEC specifies that each kind of report is filed in a different form. For instance, a 10-K report is an annual report with consolidated financial statements, and a 10-Q report is a quarterly report with financial statements. Table 1 presents the types of reports gathered in this phase. Gathering reports of different types is particularly important because, eventually, the compiled dataset not only includes tables with financial indices and numbers but also tables and cells with textual information, including standard XBRL concepts, such as "Net Income", "Current Liabilities", as well as short disclosures and notes that accompany significant events. We, therefore, produce a financial dataset primarily based on financial statements but with a rich vocabulary.

The above-described crawling phase gathered about 330 thousand reports from the generated API endpoints, grouped by report type, as Table 2 shows. As we will see in the following section, each spreadsheet contains several financial tables, resulting in the final dataset comprising millions of tables.

Data transformation. The second phase of the methodology transforms the data to a machine-readable format that provides as much exploitable information as possible. This process takes as input a raw data file from the previous phase, i.e., a workbook, and extracts the tables that are contained within the worksheets (or spreadsheets) of the workbook. Then, it proceeds to the processing of the contents of the tables in order to annotate cells with valuable attributes. Finally, it extracts the implied hierarchies of the table (if any) and stores the transformed workbook in a key-value format as a JSON file. Figure 3 provides a high-level overview of this phase.

In more detail, the procedure that we follow includes the following steps:

1. Extract the worksheets of the given workbook: A given workbook may contain several worksheets (i.e., spreadsheets) and thus contain several tables. For instance, if the workbook is from an annual report, it contains a different spreadsheet for each financial statement, a spreadsheet with company information, and some with notes, small disclosures, and highlighted economic characteristics. This step extracts all the worksheets of each workbook. Then, it identifies the different tables by retrieving the table dimensions and validating that there is actual content in the enclosed cells for the retrieved table range, not just empty space. Also, for each table, we identify the merged regions.
2. Process table: Given the tables identified at the previous step, each table is processed to extract useful information. To reduce noise and keep what is useful, we drop tiny tables that may include just a few cells containing footnotes or remarks that are not directly related to a broader context. In addition, we drop broken tables, meaning they have sparse values and many empty rows in between. The actual processing of each table starts by iterating over table cells and assessing whether they belong in a top or left header. Then, the values and possible stylistic attributes of each cell are extracted (i.e., whether there are borders, italics, foreground and background color, alignment, orientation, and more). Finally, we keep the position of the cell in the table as its spatial coordinates.

Type	Description
10-K	Annual report with a comprehensive review of the company for the past year
10-KT	Annual transition reports and supplementary material to 10-K
10-Q	Quarterly report with a continuing view of a company's financial position during the year
18-K	Annual report for foreign governments and political subdivisions
20-F	Annual and transition report of foreign private issuers
485BPOS	Registration statement for separate accounts
497	Definitive materials filed by investment companies
8-K	A report of unscheduled material events or corporate changes which could be of importance
S-1	Pre-effective registration statement submitted when a company decides to go public
S-4	Filing for the registration of securities issued in business combination transactions

Table 1. Types of report gathered from EDGAR.

10-K	10-KT	10-Q	18-K	20-F	485BPOS	497	8-K	S-1	S-4
50847	199	160045	2	3833	6652	12079	92675	1893	3223
Total number of reports: 331448									

Table 2. Number of reports downloaded for each type of report.

3. Bi-tree extraction: This step extracts the implied top-down and left-right hierarchies of the table. The hierarchical organization of a table is typically implied by merged regions and indentation. In this step we enrich the information already extracted from each cell with the cell's position in a semantic hierarchy. A prerequisite for this process is to identify how many rows (starting from the top of the table) and how many columns (starting from the left) are headers, in order to determine the depth of each hierarchy. Figure 4 shows an example, where we want to encode the information that a data region cell belongs to node A8 in the hierarchy, and this node has a sibling (A7), and their parent node is A6.
4. Dataset compilation: The last step of the process integrates all the extracted information in a key-value memory structure for each table. This allows quick and efficient lookup of the table-related information. Finally, all tables of the given workbook are exported in JSON format. Thus, at the end of this process, we have a separate JSON file for each workbook, including all the extracted tables.

It is important to note that throughout the data transformation process, a test suite framework, comprising more than 40 unit tests that cover the complete functionality of the process, ensures the validity of the extracted information. Executing the process on the gathered data, we produced 6.7 million tables (6,735,407 to be precise). The entire data transformation phase is fully automated using Python scripts. In particular, we use OpenPyXL (https://openpyxl.readthedocs.io/en/stable/) to process the spreadsheets and extract content and attributes.

Regarding the Bi-tree extraction, the related work assumes hierarchies of maximum depth equal to four[22,23]. We follow the same assumption as it covers all the table types we downloaded in the data gathering phase. To represent each hierarchy (in memory and in the JSON format), we use a key-value dictionary. Each cell in the hierarchy constitutes a node that is again represented as a dictionary, containing the row and column indices (positional coordinates) and a list of its children nodes. The children nodes follow the same key-value representation, including in turn their positional coordinates and their children nodes. This way a nested structure that represents the hierarchy is created, as shown in Listing 2. The coordinates of the root node are (-1,-1) since the root is assumed to lie "outside" the table (see Fig. 4). Figure 5 provides an example of a financial table and illustrates how the top hierarchy is represented after the data transformation process.

```
node = {
        'RI': <RowIndex>,
        'CI': <ColumnIndex>,
        'Cd': [{
              'RI': <RowIndex>,
              'CI': <ColumnIndex>,
              'Cd': [
                      ...
              ]
            },
            ...
        ]
    }
```

Listing 2 Representation of hierarchy nodes.

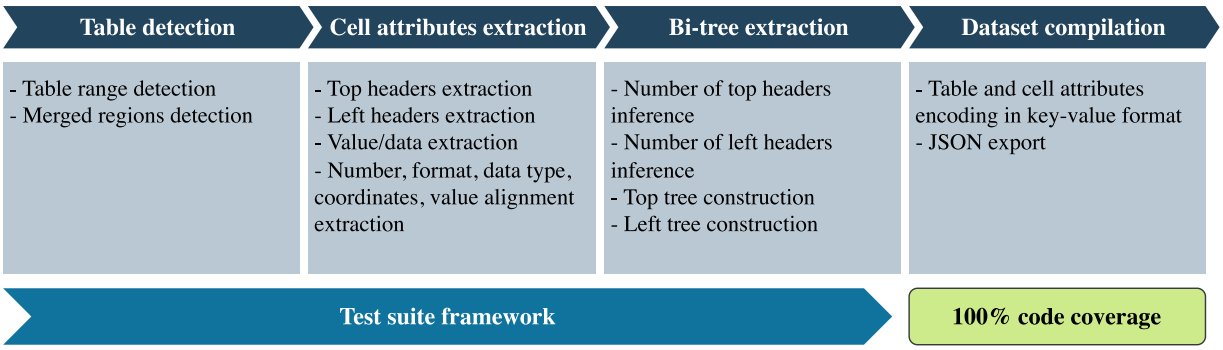

Fig. 3 Data transformation workflow. For each spreadsheet, the table boundaries and the merged regions are detected. Then, headers and cell values and attributes are extracted. At the third step, the top-down and left-right hierarchies are constructed and finally, the transformed data are stored in JSON format. A test suite that provides unit testing of all steps covers the functionality of the entire process.

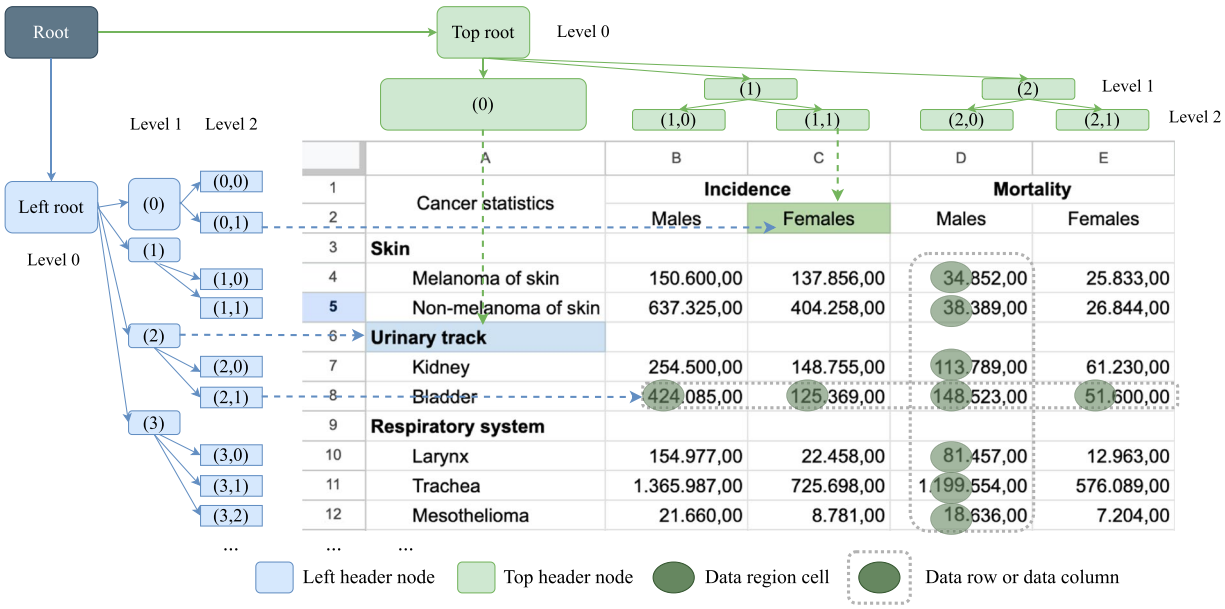

Fig. 4 Implied hierarchies in table: both the top-down hierarchy and the left-to-right hierarchy contain three levels each. Cell A6 is the parent node of cells A7 and A8, and its siblings are A3 and A9. D2 is a child of DE1 and its sibling is E2. (Image inspired from Wang *et al.*[22]).

Data Records

The dataset is available at Zenodo[33], with this section being the primary source of information on the availability and content of the data being described.

As mentioned above, the compiled dataset consists of about 330 thousand JSON files. Each file corresponds to a company filing (e.g., a report) and contains several tables. The files are organized in a directory structure that includes different folders for different types of report, as shown in Fig. 6. This structure facilitates file system handling and allows selective focus based on different types of report and company filing. Therefore, the parent directory contains as many subfolders as the different type of reports (see Table 1), e.g., a subfolder named "10-K" containing the data extracted from "10-K" reports, a subfolder named "10-Q" containing the data extracted from "10-Q" reports, and so forth. Each subfolder is compressed as a zip file and uploaded to Zenodo.

The name convention of the JSON files follows the pattern $< CIK > \_ < YEAR > \_ < TYPE > \_ < ACCESSION\_NUMBER > .json$, where *CIK* is the central index key of the company, *YEAR* is the year of the filing submission, *TYPE* is the type of report (e.g., 10-K), and *ACCESSION_NUMBER* is the unique identifier by EDGAR.

Each JSON file contains a list. Each element of that list corresponds to a table represented as a dictionary. Figure 6 shows an example of a JSON file containing three dictionaries with the first one expanded. For each table, we store metadata, such as the storage description, the language, and the spreadsheet name. We also include table-specific information that comprises the dimensions of the table ("RangeAddress"), its title, its content ("Cells"), the identified merged regions, the number of top and left headers ("TopHeaderRowsNumber" and "LeftHeaderColumnsNumber" respectively) and the Bi-tree ("TopTreeRoot" and "LeftTreeRoot").

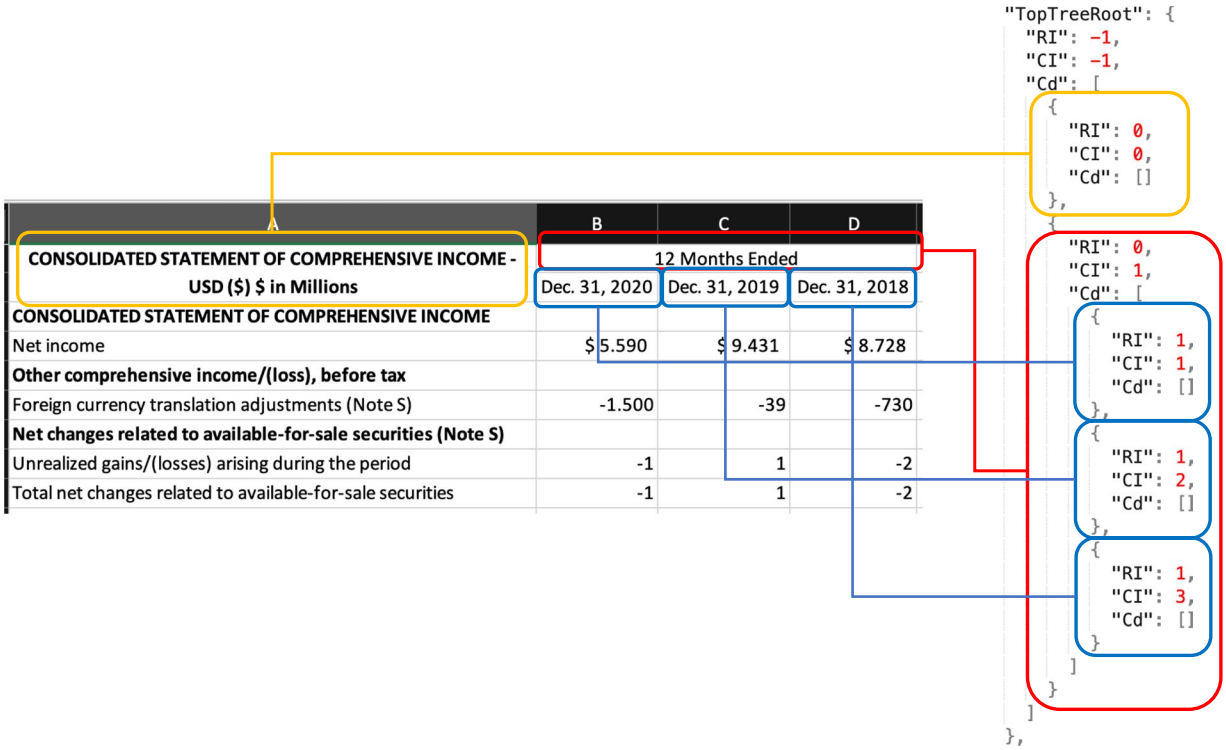

Fig. 5 An example of the top hierarchy representation at the data transformation phase.

All the table content with its attributes is under the key "Cells". This structure is essentially a list of lists that represent the rows and columns of the table. Thus, the elements of the outer list are the rows, and each row is a list, the elements of which are the cells of the corresponding columns. As shown in Fig. 6, each cell is a dictionary comprising the following information:

- "T": the value of the cell as string
- "V": the value of the cell
- "is_header": whether the cell participates in a header row
- "is_attribute": whether the cell participates in a header column
- "coordinates": the spatial coordinates of the cell ([row index, column index])
- "HF": whether the cell has formula
- "A1": formula specific: absolute cell reference
- "R1": formula specific: relative cell reference
- "font_name": the name of the font
- "font_size": the size of the font
- "wrap_text": whether the text is wrapped within the cell dimension
- "BC": whether there is a non-white background
- "FC": whether there is a non-black font color
- "FB": whether the font is bold
- "I": whether the font is italic
- "NS": the number format of the cell
- "DT": the data type of the cell
- "LB": whether the cell has a left border
- "TB": whether the cell has a top border
- "BB": whether the cell has a bottom border
- "RB": whether the cell has a right border
- "O": the orientation of the cell
- "HA": whether the cell is horizontally aligned
- "VA": whether the cell is vertically aligned

The remaining table attributes are the number of top header rows and the number of the left header columns, the two hierarchies presented in the previous section (see also Fig. 5 as an example), and the "MergedRegions" that contains all the identified merged areas of the table. The latter is a list of dictionaries, one for every identified merged region, indicating the starting row and column and the ending row and column of the region.

Regarding dataset statistics, there are about 6.7 million tables in total. On average, there are 20 tables per filing, i.e., per JSON file. Each table has approximately 25 rows on average and five columns. The number of

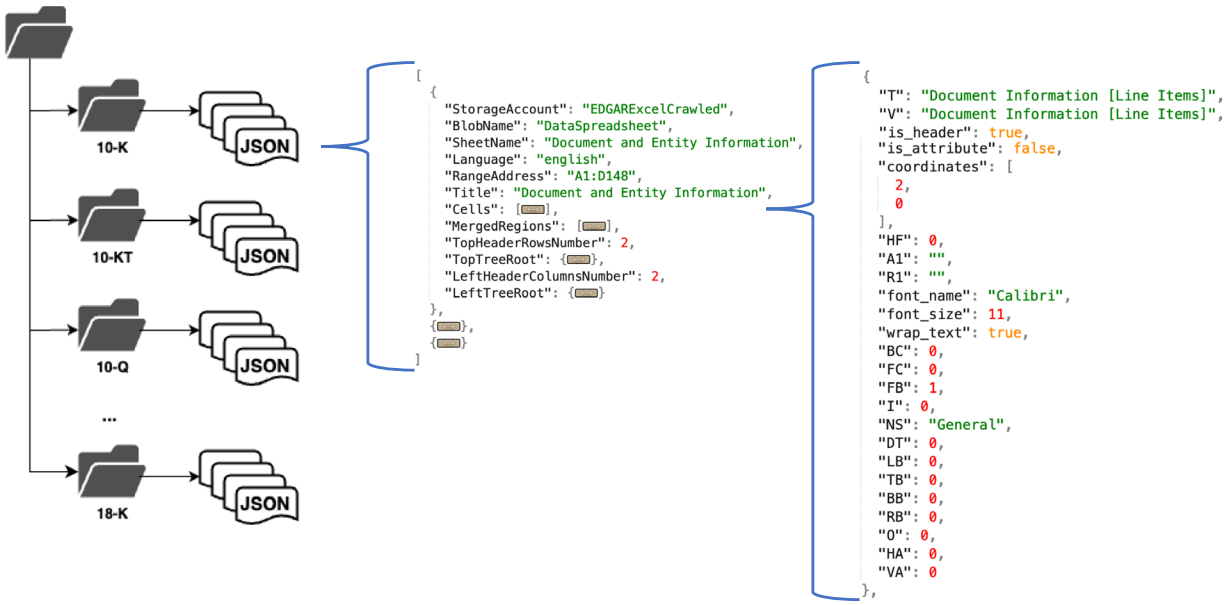

Fig. 6 Directory structure of the dataset and analysis of JSON contents.

Statistics	Tables per JSON	Rows per table	Columns per table	Cells per table
Mean	20.32	24.84	4.54	130.91
Q1 (25%)	1.00	8.00	3.00	27.00
Q2 (50%)	18.00	15.00	3.00	54.00
Q3 (75%)	31.00	28.00	5.00	124.00

Table 3. Descriptive statistics of the produced dataset.

columns does not vary much. On the other hand, the number of rows may deviate much more from the mean. Finally, the number of cells per table also varies largely, with a mean value of 130 and with many tables containing hundreds or even thousands of cells. Table 3 summarizes the essential statistical information of the dataset. The heavy-tail distribution of the rows per table (and consequently for the cells per table) is due to the variety of types of report in the dataset. Reports of financial statements often include additional highlights with notes and numerical indices across years that tend to produce tables with many rows and cells. All the above contribute to the large variety in tables and vocabulary this dataset offers.

Technical Validation

This article is the first public release of the ENTRANT dataset and the code used to produce it. Thus, the research community has not yet had the opportunity to review it and provide feedback. However, all users are encouraged to fork the data and code, report issues, and make pull requests with their modifications.

Having said that, we produced the ENTRANT dataset intending to be readily consumable by large models, such as TUTA[22] and FORTAP[23], as well as to be easily exploitable and manageable for other use cases. The code used to build ENTRANT was version-controlled and well tested. For this purpose, a test suite framework was implemented, comprising more than 40 unit tests that cover all the functionality of the code. The unit tests focus on data quality and representational inconsistencies that may occur in the data transformation process due to either human error or noise in the initial raw data. Specifically, the unit and sanity tests cover the following aspects:

- Table identification: validates that all tables that could have been extracted from a given workbook should have been extracted. Additionally, we test for the correct extraction of the table titles and the table dimensions. Regarding the latter, we ensure that the number of rows and columns is consistent in the table and matches the identified dimensions. Moreover, we check for the correct header identification, and we perform these tests for several types of workbook (i.e., workbooks that refer to 10-K, 10-Q, and 8-K reports).
- Merged region identification: validates that all merged regions and table cells have been identified and encoded correctly in the final JSON representation. It also ensures that the number of the identified merged regions matches the size of the data structure that is used for the representation.
- Cell attribute extraction: validates that all cell attributes have been computed. We test for cells that participate in header rows and those that do not participate in table headers to ensure that the expected number of

Model	DeEx	SAUS
TUTA original	76.6	90.2
TUTA (WikiTables, WDC, ENTRANT)	78.8	91.9
TUTA (ENTRANT)	74.2	88.4

Table 4. Results in terms of Macro F1 scores on cell type classification datasets.

attributes has been calculated and that the corresponding attribute values have been correctly extracted from the worksheet. Again, we perform these tests for several types of workbook.

- Bi-tree extraction: validates that the two implied hierarchies have been extracted correctly. This set includes tests on the number of top headers and the number of left headers. We also check for the correct coordinates of the top tree root and the left tree root. Finally, we ensure each node contains the correct positional coordinates and the corresponding list of its children nodes.

At the time of the code publication, all unit tests are successfully completed, as shown in Listing 3.

```
collected 45 items
...
tests/all_tables_from_10K_test.py                                    PASSED [ 20%]
tests/all_tables_from_10Q_test.py                                    PASSED [ 31%]
tests/all_tables_from_8K_test.py                                     PASSED [ 42%]
tests/bitree_test.py::TestBiTreeExtraction10K::testLeftTreeRoot PASSED [ 46%]
tests/bitree_test.py::TestBiTreeExtraction10K::testTopHeaders   PASSED [ 48%]
tests/bitree_test.py::TestBiTreeExtraction10K::testTopTree      PASSED [ 51%]
tests/bitree_test.py::TestBiTreeExtraction10K::testTopTreeRoot  PASSED [ 53%]
tests/bitree_test.py::TestBiTreeExtraction10K::test_left_tree   PASSED [ 55%]
tests/cell_attributes_from_10K_test.py                               PASSED [ 60%]
tests/cell_attributes_from_10Q_test.py                               PASSED [ 64%]
tests/cell_attributes_from_8K_test.py                                PASSED [ 68%]
tests/merged_regions_test.py                                         PASSED [ 73%]
tests/table_from_10K_test.py                                         PASSED [ 82%]
tests/table_from_10Q_test.py                                         PASSED [ 91%]
tests/table_from_8K_test.py                                          PASSED [100%]
========================================== 45 passed in 51.89s ==========================================
```

Listing 3 Successful test session. High-level unit tests are shown.

Besides the unit tests, as an additional validation step, a post-processing procedure checks that there are no empty regions, rows and columns in each extracted table. This step ensures that all corresponding values have been successfully retrieved and that there are no empty spaces from the initial workbooks that have been considered as parts of the extracted tables. As a final technical validation step, all produced JSON files have been parsed an loaded to memory as dictionaries successfully, using the JSON Python module, in order to endure that all data can be parsed and loaded correctly.

Regarding the actual usage of the dataset, we used it to pre-train TUTA[22], a large model for table understanding that exploits all the information that is encoded in the data, including cell attributes, table attributes, and bi-trees. We used the code published (https://github.com/microsoft/TUTA_table_understanding) by the authors of the model to pre-train it and then employed it on a downstream task that focuses on cell type classification.

In the pre-training phase, we trained two models. The first used three large datasets, namely the WikiTable (provided by the authors of TUTA), WDC (also provided by the authors), and ENTRANT. The second model consumed only the ENTRANT data. The downstream task aimed to identify fine-grained cell types. Cell type classification has been widely studied[6,11,34–36] with several smaller datasets, and it is a task that requires models to capture both semantic and structural information, which is something that ENTRANT encodes successfully.

The datasets used for the downstream task are DeEx[35] and SAUS[6], which are collected from various domains (financial, educational, public health) and thus contain tables with various structures and semantics. DeEx and SAUS categorize cells into general types: “metadata”, “notes”, “data”, “top attribute”, “left attribute”, and “derived”. We, therefore, evaluated the models on a multi-class classification setting. Table 4 provides the classification results in terms of Macro F1 score. The models we included in this task are those mentioned above, i.e., the two pre-trained TUTA models, plus the original TUTA that was pre-trained using WikiTable, WDC, and web-crawled spreadsheets that are not publicly available.

The most important conclusion of this experiment, for the purposes of the present paper, was that the compiled dataset is easily consumable and useful for state-of-the-art models. Additionally, the pre-trained model that uses only the ENTRANT data provides results that do not deviate much from the other two models. When combined with other public data (WikiTables and WDC) it leads to state-of-the-art results. Therefore, although the dataset is focused on the financial domain, it provides a rich vocabulary and useful semantic and structural information.

Usage Notes

All researchers and interested parties can access the dataset freely at Zenodo[33]. The dataset is based on publicly and freely available information from SEC's EDGAR, enriched with metadata and table and cell attributes to facilitate research and experiments that rely on financial information. According to EDGAR, access to its public database is free, allowing everyone to retrieve public companies' financial information and operations, by reviewing the filings the companies make with the SEC.

The way the dataset is organized facilitates quick lookups for tables of specific filing types and specific years. The JSON format facilitates quick data loading and processing, as well as interoperability between applications, without explicit requirements for specialized libraries.

The dataset can be conveniently processed and used for various tasks. The primary aim of this research was to compile an extensive dataset for pre-training large models for table understanding. One direct usage, therefore, is to pre-train a model like TUTA[22] or FORTAP[23]. We have already demonstrated this by pre-training TUTA.

In addition, the data can be used as a source to extract financial indices from the financial statements, which can in turn be used as features for machine learning methods in financial use cases. For example, financial misstatement detection and distress or bankruptcy prediction models rely heavily on such features in combination with market data or other operational and categorical variables.

Finally, the variety of the tables that this dataset contains allows the selection or sampling of tables that meet specific criteria for table classification, cell classification, or other tasks, in order to create smaller datasets for direct training and testing. For instance, the dataset could be used to create smaller and focused datasets for argument mining tasks or table question-answering. Tables that correspond to specific filings (e.g., annual reports) of companies can be combined with textual information or the auditor's opinion that appears in the reports to support argument mining regarding the concerns that the auditors express, given the financial status of the company[37].

Code availability

We automated both phases of the data compilation process presented in this work, using the Python programming language and corresponding libraries that are freely distributed by the Python Package Index (PyPI). The data resources described in this paper, including Python scripts, can be accessed without restrictions at GitHub (https://github.com/iit-Demokritos/entrant). Anyone can browse the content of the repository, and everyone is encouraged to fork or create pull requests with improvements and enhancements. The code is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as appropriate credit to the original author(s) and the source are given, a link to the Creative Commons license is provided, and any changes are clearly indicated.

By releasing the code for this work, we aim to enable further augmentation of the dataset, as anyone can use it to fetch more reports from EDGAR and transform them in the described format. An additional advantage is that the transformation process alone is a powerful tool, since it allows the transformation of excel files that follow similar table formats to the described JSON format.

Received: 23 February 2024; Accepted: 4 July 2024;
Published online: 13 August 2024

References

1. Eberius, J. *et al*. Deexcelerator: A framework for extracting relational data from partially structured documents. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, 2477-2480, https://doi.org/10.1145/2505515.2508210 (Association for Computing Machinery, New York, NY, USA, 2013).
2. Shigarov, A. Table understanding: Problem overview. *WIREs Data Mining and Knowledge Discovery* **13**, e1482, https://doi.org/10.1002/widm.1482 (2023).
3. Pujara, J., Szekely, P., Sun, H. & Chen, M. From tables to knowledge: Recent advances in table understanding. In *KDD '21: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 4060–4061, https://doi.org/10.1145/3447548.3470809 (2021).
4. Pinto, D., McCallum, A., Wei, X. & Croft, W. Table extraction using conditional random fields. In *SIGIR'03*, https://doi.org/10.1145/860435.860479 (Toronto, Canada, 2003).
5. Koci, E., Thiele, M., Romero, O. & Lehner, W. A machine learning approach for layout inference in spreadsheets. In *International Conference on Knowledge Discovery and Information Retrieval*, https://doi.org/10.5220/0006052200770088 (2016).
6. Gol, M. G., Pujara, J. & Szekely, P. Tabular cell classification using pre-trained cell embeddings. In *2019 IEEE International Conference on Data Mining (ICDM)*, 230–239, https://doi.org/10.1109/ICDM.2019.00033 (2019).
7. Zhang, L., Zhang, S. & Balog, K. Table2vec: Neural word and entity embeddings for table population and retrieval. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1029-1032, https://doi.org/10.1145/3331184.3331333 (ACM, 2019).
8. Nishida, K., Sadamitsu, K., Higashinaka, R. & YoshihiroMatsuo. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, https://doi.org/10.1609/aaai.v31i1.10484 (2017).
9. Günther, M., Thiele, M., Gonsior, J. & Lehner, W. Pre-trained web table embeddings for table discovery. In *Proceedings of the Fourth International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, aiDM '21, 24-31, https://doi.org/10.1145/3464509.3464892 (Association for Computing Machinery, New York, NY, USA, 2021).
10. Chen, J., Jiménez-Ruiz, E., Horrocks, I. & Sutton, C. Colnet: Embedding the semantics of web tables for column type prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 29–36, https://doi.org/10.48550/arXiv.1811.01304 (2019).
11. Dong, H., Liu, S., Fu, Z., Han, S. & Zhang, D. Semantic structure extraction for spreadsheet tables with a multi-task learning architecture. In *Workshop on Document Intelligence at NeurIPS*, https://openreview.net/forum?id=r1x3GTq5IB (2019).
12. Fetahu, B., Anand, A. & Koutraki, M. Tablenet: An approach for determining fine-grained relations for wikipedia tables. In *The World Wide Web Conference*, 2736-2742, https://doi.org/10.48550/arXiv.1902.01740 (2019).
13. Kardas, M. *et al*. AxCell: Automatic extraction of results from machine learning papers. In Webber, B., Cohn, T., He, Y. & Liu, Y. (eds.) *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 8580–8594, https://doi.org/10.18653/v1/2020.emnlp-main.692 (Association for Computational Linguistics, Online, 2020).
14. Chen, J., Jiménez-Ruiz, E., Horrocks, I. & Sutton, C. Learning semantic annotations for tabular data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, https://doi.org/10.48550/arXiv.1906.00781 (2019).

15. Koci, E., Thiele, M., Lehner, W., & Romero, O. Table recognition in spreadsheets via a graph representation. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, 139-144, https://doi.org/10.1109/DAS.2018.48 (IEEE, 2018).
16. Zayats, V., Toutanova, K. & Ostendorf, M. Representations for question answering from documents with tables and text. In Merlo, P., Tiedemann, J. & Tsarfaty, R. (eds.) *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2895–2906, https://doi.org/10.18653/v1/2021.eacl-main.253 (Association for Computational Linguistics, Online, 2021).
17. Zhang, X. *et al.* A graph representation of semi-structured data for web question answering. In *Proceedings of the 28th International Conference on Computational Linguistics*, 51–61, https://doi.org/10.48550/arXiv.2010.06801 (2020).
18. Yin, P., Neubig, G., Yih, W.-t. & Riedel, S. TaBERT: Pretraining for joint understanding of textual and tabular data. In Jurafsky, D., Chai, J., Schluter, N. & Tetreault, J. (eds.) *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 8413–8426, https://doi.org/10.18653/v1/2020.acl-main.745 (Association for Computational Linguistics, Online, 2020).
19. Herzig, J., Nowak, P. K., Müller, T., Piccinno, F. & Eisenschlos, J. TaPas: Weakly supervised table parsing via pre-training. In Jurafsky, D., Chai, J., Schluter, N. & Tetreault, J. (eds.) *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4320–4333, https://doi.org/10.18653/v1/2020.acl-main.398 (Association for Computational Linguistics, Online, 2020).
20. Deng, X. *et al.* Structure-grounded pretraining for text-to-SQL. In Toutanova, K. *et al.* (eds.) *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1337–1350, https://doi.org/10.18653/v1/2021.naacl-main.105 (Association for Computational Linguistics, Online, 2021).
21. Deng, X., Sun, H., Lees, A., Wu, Y. & Yu, C. Turl: Table under-standing through representation learning, https://doi.org/10.48550/arXiv.2006.14806 (2020). 2006.14806.
22. Wang, Z. *et al.* Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, 1780-1790, https://doi.org/10.1145/3447548.3467434 (Association for Computing Machinery, New York, NY, USA, 2021).
23. Cheng, Z. *et al.* Fortap: Using formulas for numerical-reasoning-aware table pretraining. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 1150–1166, https://doi.org/10.48550/arXiv.2109.07323 (2022).
24. Bhagavatula, C., Noraset, T. & Downey, D. Tabel: Entity linking in web tables. In *Proceedings of the International Semantic Web Conference*, https://doi.org/10.1007/978-3-319-25007-6_25 (2015).
25. Trabelsi, M., Chen, Z., Zhang, S., Davison, B. D. & Heflin, J. Strubert: Structure-aware bert for table search and matching. In *Proceedings of the ACM Web Conference 2022 (WWW '22)*, https://doi.org/10.1145/3485447.3511972 (ACM, Virtual Event, Lyon, France, 2022).
26. Habibi, M., Starlinger, J. & Leser, U. Tabsim: A siamese neural network for accurate estimation of table similarity. In *2020 IEEE International Conference on Big Data (Big Data)*, 930–937, https://doi.org/10.1109/BigData50022.2020.9378077 (2020).
27. Iida, H., Thai, D., Manjunatha, V. & Iyyer, M. TABBIE: Pretrained representations of tabular data. In Toutanova, K.*et al.* (eds.) *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 3446–3456, https://doi.org/10.18653/v1/2021.naacl-main.270 (Association for Computational Linguistics, Online, 2021).
28. Eberius, J. *et al.* Building the dresden web table corpus: A classification approach. In *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*, 41–50, https://doi.org/10.1109/BDC.2015.30 (2015).
29. Lehmberg, O., Ritze, D., Meusel, R. & Bizer, C. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, 75-76, https://doi.org/10.1145/2872518.2889386 (International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2016).
30. Zhu, F. *et al.* TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In Zong, C., Xia, F., Li, W. & Navigli, R. (eds.) *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 3277–3287, https://doi.org/10.18653/v1/2021.acl-long.254 (Association for Computational Linguistics, Online, 2021).
31. Zavitsanos, E. *et al.* Financial misstatement detection: a realistic evaluation. In *2nd ACM International Conference on AI in Finance (ICAIF`21)*, 1–9, https://doi.org/10.1145/3490354.3494453 (Association for Computing Machinery, November 3-5, 2021, Virtual Event, USA., 2021).
32. Dyck, A., Morse, A. & Zingales, L. Who blows the whistle on corporate fraud? *The Journal of Finance* **65**, 2213–2253, https://doi.org/10.1111/j.1540-6261.2010.01614.x (2010).
33. Zavitsanos, E., Mavroeidis, D., Spyropoulou, E., Fergadiotis, M. & Paliouras, G. Entrant: A large financial dataset for table understanding. *Zenodo* https://doi.org/10.5281/zenodo.10667088 (2024).
34. Gonsior, J. *et al.* Active learning for spreadsheet cell classification. In *EDBT/ICDT Workshops* (2020).
35. Koci, E., Thiele, M., Rehak, J., Romero, O. & Lehner, W. Deco: A dataset of annotated spreadsheets for layout and table recognition. In *International Conference on Document Analysis and Recognition*, https://doi.org/10.1109/ICDAR.2019.00207 (IEEE, 2019).
36. Sun, K., Rayudu, H. & Pujara, J. A hybrid probabilistic approach for table understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 4366–4374, https://doi.org/10.1609/aaai.v35i5.16562 (2021).
37. Bougiatiotis, K., Zavitsanos, E. & Paliouras, G. Identifying going concern issues in auditor opinions: link to bankruptcy events. In *2023 IEEE International Conference on Big Data (BigData)*, 2805–2813, https://doi.org/10.1109/BigData59044.2023.10386452 (2023).

Acknowledgements

We appreciate the support of "SKEL | The AI Lab" of the Institute of Informatics and Telecommunications, NCSR "Demokritos" for providing computational and storage resources. We would also like to acknowledge the financial support of Qualco SA. The opinions of the authors expressed herein do not necessarily state or reflect those of Qualco SA.

Author contributions

Work conception and design: E.Z., E.S., M.F., D.M., G.P. Data acquisition: D.M. Data transformations and analysis: E.Z. Revisions of code, workflow and unit testing: E.Z., D.M., M.F. Data usability: M.F., D.M. Wrote the paper: E.Z. All authors reviewed the manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to E.Z.