MORPHER: Structural Transformation of Ill-formed Rows

Mazhar Hameed
Hasso Plattner Institute
University of Potsdam, Germany
mazhar.hameed@hpi.de

Gerardo Vitagliano
Hasso Plattner Institute
University of Potsdam, Germany
gerardo.vitagliano@hpi.de

Felix Naumann
Hasso Plattner Institute
University of Potsdam, Germany
felix.naumann@hpi.de

ABSTRACT

Open data portals contain a plethora of data files, with comma-separated value (CSV) files being particularly popular with users and businesses due to their flexible standard. However, this flexibility comes with much responsibility for data consumers, as many files contain various structural problems, e.g., a different number of cells across data rows, multiple value formats within the same column, different variants of quoted fields due to user specifications, etc. We refer to rows that contain such structural inconsistencies as *ill-formed*. Consequently, ingesting them into a host system, such as a database or an analytics platform, often requires prior data preparation steps.

We propose to demonstrate Morpher, a desktop-based system that incorporates our state-of-the-art error detection system, Suragh [9] and extends it to also clean the files at hand. Morpher facilitates ingesting CSV files by automatically identifying and cleaning ill-formed rows while preserving all data. It comprises three key components: 1) The pattern modeler, which generates syntax-based patterns for each row of the input file. The system uses these patterns to classify rows into ill-formed and well-formed. 2) The pattern classifier obtains row patterns for ill-formed rows and uses them to distinguish ill-formed but wanted rows from ill-formed unwanted rows. 3) The pattern wrangler transforms the identified wanted rows into well-formed rows, effectively repairing a wide range of formatting problems.

CCS CONCEPTS

• **Information systems** → **Data cleaning**; **Extraction, transformation and loading**.

KEYWORDS

data preparation, data representation, file structure transformation

ACM Reference Format:
Mazhar Hameed, Gerardo Vitagliano, and Felix Naumann. 2023. MORPHER: Structural Transformation of ill-formed Rows. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3583780.3614747

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3614747

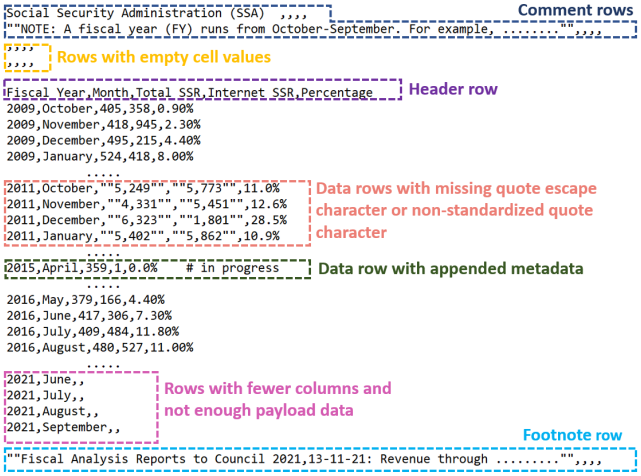


Figure 1: A sample of a raw CSV file with *ill-formed* rows due to structural inconsistencies that exist at both column- and row-levels.

1 ILL-FORMED AND WELL-FORMED ROWS

As a widely adopted data format, comma-separated value (CSV) files have become a staple for data scientists and machine learning engineers, providing an easy-to-use option for data sharing and processing [2, 14]. However, due to their loose format, these files appear in various dialects [4, 18] deviating from the RFC standard [11], and may contain various structural inconsistencies [3, 9]. Consequently, accurately loading them into data-driven systems requires addressing various challenges, among which detecting and cleaning ill-formed rows pose significant difficulties.

Morpher is an interactive system that aims to assist users in detecting and cleaning ill-formed rows in CSV files. Its interface allows users to visualize, for an input file, a classification of ill-formed *wanted* and ill-formed *unwanted* rows with a corresponding cleaned version and provides a seamless export of the final results as both CSV and Microsoft Excel workbook (.xlsx) formats for convenient use.

Our research has shown that ill-formed rows are common in CSV files, often resulting from loosely defined schemata, formatting errors, and discrepancies in row structures [9]. These challenges are also not foreign to the research community. Surveys and studies on CSV files have revealed various challenges, including dialect variations [8, 14], multiple tables [3], comment lines [13], and complex headers [9]. Despite research advancements in CSV parsing [5, 18] and classification [1, 13], it remains unclear how these findings have been applied in real-world systems and the extent to which data preparation burden still falls on end-users [19]. This issue is

also prevalent in CSV files available on open data portals, such as www.data.gov. Out of 2 066 files randomly selected from this portal, 20.2% (418 files) could not be directly loaded into many advanced data preparation and cleaning tools, such as Trifacta Wrangler [12], Tableau [17], and OpenRefine [7], due to the presence of ill-formed rows. Figure 1 shows a sample of a raw CSV file obtained from a government data portal, highlighting groups of ill-formed rows with different inconsistencies. We aim to automatically *detect* those ill-formed rows containing data and *clean* them by repairing their structural inconsistencies.

To detect ill-formed rows, we make use of our pattern-based system, Suragh[1], which abstracts row structures into structural patterns based on a syntactic pattern grammar. With Morpher, we not only improve the row classification by recognizing wanted and unwanted ill-formed rows but also enable accurate data loading by cleaning up the structure of wanted rows, ensuring that all remaining rows are structurally well-formed (see Section 2 for details). Morpher is capable of handling various types of structural inconsistencies, such as incorrect field boundaries, ambiguous row boundaries, values containing special characters for reference, missing quotes and escape characters, mismatched quote and escape characters, fields spanning to multiple rows, data mixed with metadata, rows with varying lengths, and superfluous information (comments, notes, etc.).

The graphical interface of Morpher allows for seamless interaction with the results of automated row classification and transformation. Users can easily navigate through the rows within a file and review their classification. With automatic row transformation, users can clean up the structure of detected ill-formed wanted rows with just a click. An interactive Morpher demo, accompanied by a demonstration scenario video, can be accessed online[2].

The subsequent sections of the paper are structured as follows: Section 2 provides a brief overview of Morpher. Section 3 demonstrates the practical usage of our system through the graphical interface. Finally, Section 4 summarizes the contributions of the paper.

2 AN OVERVIEW OF MORPHER

Morpher performs row classification and cleaning in three phases. In the first phase, it uses Suragh to classify input file rows as either ill-formed or well-formed based on the dominant row pattern(s) (Section 2.1). Additionally, Suragh generates row patterns for the ill-formed rows, which are referred to as potential row patterns. This process is repeated incrementally until no ill-formed rows remain without a potential pattern. In the second phase, Morpher obtains the potential row patterns from the previous phase and further classifies ill-formed rows into wanted and unwanted ones, based on a comparison of column patterns (Section 2.1). Finally, in the third phase, Morpher transforms the wanted rows into well-formed ones using a set of pattern transformations (Section 2.2).

Due to the proposal's limited space, we provide only a brief overview of the system components and highlight the user experience of Morpher. For a more comprehensive understanding of algorithmic details, we encourage readers to refer to our prior work, which includes detailed descriptions and extensive experimental analysis on real-world datasets [9]. Figure 2 provides a brief overview of Morpher's functionalities through a graphical user interface (GUI).

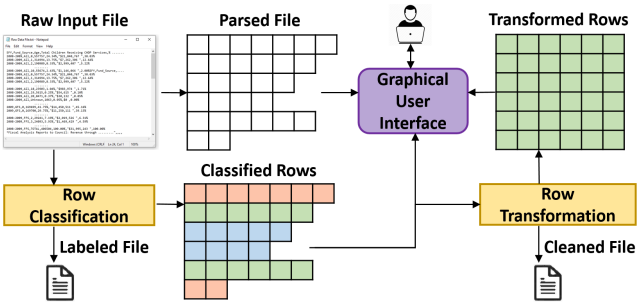


Figure 2: Morpher overview

2.1 Row Classification

Morpher builds upon Suragh, which classifies the rows of an input file as either ill-formed or well-formed. To do so, Suragh aims to identify the structure of input file rows by generating syntax-based patterns through a defined grammar. The grammar contains two types of production rules: (1) encoders and (2) aggregators, known as abstractions. Encoders transform individual characters into more general representations. For example, the character "1" is represented as $\langle D \rangle$, which stands for "Digit". The aggregator abstractions combine the representations from other encoders and aggregators based on the predefined rules. For example, the character sequence "123" is first encoded as $\langle D \rangle\langle D \rangle\langle D \rangle$, and then can be combined into a single abstraction $\langle SEQD \rangle$ – a "Sequence of Digits".

Using the given pattern grammar, Suragh generates syntax-based patterns at cell, column, and row level. The application of the grammar rules is order-dependent, and every input value can be transformed into one or more *cell patterns*. The generated patterns have different levels of "specificity", which depend on the abstraction levels they comprise. Once Suragh generates patterns for each cell value, it compiles them into *column patterns* that represent one or more cell values in a column. Suragh obtains a *row pattern* by choosing one column pattern for each input file column. To determine useful row patterns that represent one or more rows, Suragh examines all possible combinations of column patterns, selecting those with high specificity and coverage.

Row patterns that are not a proper subset of another row pattern are called *dominant* patterns. To avoid pattern redundancy, Suragh detects and removes all non-dominant row patterns. The remaining dominant row pattern(s) are then used to classify each row as either ill-formed or well-formed: A row *conforms to* a dominant row pattern if it has the same number of columns as the dominant pattern, and if all of its column values conform to the corresponding column patterns of the dominant pattern. We call such a row *well-formed*, and *ill-formed* otherwise [9].

Figure 3 shows the output of Suragh for the input file presented in Figure 1: a dominant row pattern, the rows that comply with the dominant pattern, and the non-compliant (ill-formed) rows.

[1] Suragh is an Urdu word that refers to examining an event.
[2] https://github.com/HMazharHameed/MORPHER

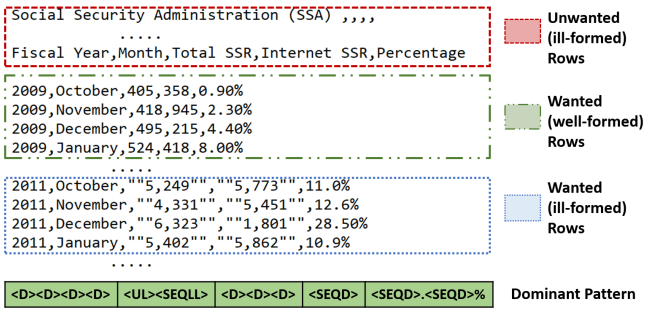
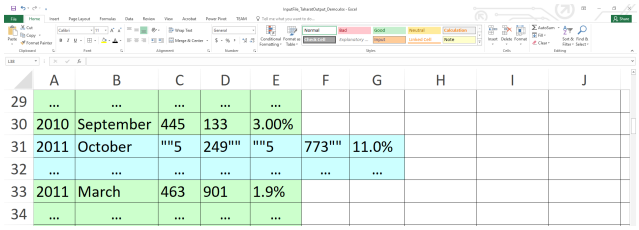
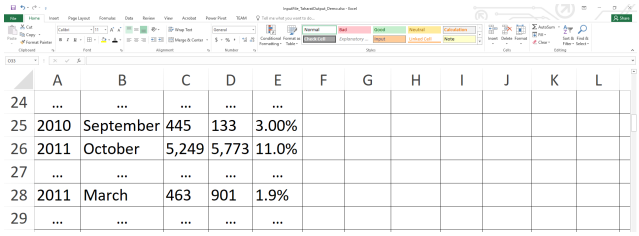


Figure 3: Selected rows of the CSV file of Figure 1 with a dominant row pattern (shaded green table) for well-formed rows (highlighted green), ill-formed wanted rows (highlighted blue), and ill-formed unwanted rows (highlighted red). The cell separators in the table indicate the ⟨*DEL*⟩ abstraction, which we omit due to space limitation.



	A	B	C	D	E	F	G
29	...	...	...	...	...		
30	2010	September	445	133	3.00%		
31	2011	October	""5	249""	""5	773""	11.0%
32	...	...	...	...	...	...	...
33	2011	March	463	901	1.9%		
34	...	...	...	...	...		

(a) An overview of classified rows of the CSV file of Figure 1



	A	B	C	D	E
24	...	...	...	...	...
25	2010	September	445	133	3.00%
26	2011	October	5,249	5,773	11.0%
27	...	...	...	...	...
28	2011	March	463	901	1.9%
29	...	...	...	...	...

(b) An overview of transformed rows of the CSV file of Figure 1

Figure 4: MORPHER's output exported as an Excel workbook

With MORPHER, we further refine this process by classifying ill-formed rows that carry data as *wanted*, and all others as *unwanted*. The output of MORPHER's classification process is merged with the SURAGH's results in the same figure, minimizing visual clutter.

Some rows are ill-formed and contain no data, such as table titles, footnotes, or empty rows. We refer to these rows as *ill-formed unwanted*. Other rows may contain data but still be ill-formed due to additional structural or formatting information and possibly extra columns, necessitating data preparation for them to be properly formatted. We refer to these rows as *ill-formed wanted*. In contrast, rows without any inconsistencies are *well-formed* and are wanted by default. To obtain a structurally sound file, we need to clean up the structure of ill-formed wanted rows and remove ill-formed unwanted ones. Although it is easily possible to retain unwanted rows to prevent data loss, we choose to remove them to streamline the file loading process.

To classify ill-formed rows into wanted and unwanted, MORPHER uses the generated potential patterns to assess their similarity to dominant patterns that cover data rows by introducing a pattern-level distance measure inspired by sequence alignment [6, 10].

The operations used for the alignment include "match", "mismatch", and "indel" (insertion and deletion), each with a specific cost. We introduce a distance-based alignment framework for pattern-by-pattern alignments of entire row patterns, where the input row patterns are compared column by column, splitting them on the delimiter character of the raw CSV file. Our alignment framework calculates the distance between dominant and potential row patterns by aligning them using a dynamic programming approach, similar to other sequence alignment methods [15, 16].

To align row patterns, we use the information from the calculated dynamic programming matrix to pad shorter patterns with gaps ("-"), if needed, to generate two patterns with the same number of columns. We then calculate the distance between a dominant and potential pattern's column patterns using a formula that considers the pattern's abstraction details. The row pattern distance is obtained by averaging the column-wise distances, which determines the similarity between potential and dominant row patterns. The resulting score ranges from 0 to 1, where 1 represents complete dissimilarity. A lower distance score indicates higher similarity, allowing us to classify a potential pattern as wanted.

Ultimately, the classification of potential patterns determines whether the corresponding ill-formed rows contain data (wanted) or not (unwanted).

2.2 Row Transformation

In this phase, MORPHER collects the necessary information from the row classification phase to transform ill-formed wanted rows into well-formed ones. Using the best alignment between dominant and wanted patterns, it determines the necessary transformations to transform one pattern into another. A set of pattern transformation operators is used to transform one pattern into another and to transform the corresponding rows, including Extract that extracts a (wanted) part from a column pattern, Merge that concatenates column patterns and appends the merged column pattern to the specified position, and Reformat that offers four functionalities: adding or removing quotes, escapes, field separators, and row separators from a given column pattern.

MORPHER takes the aligned column patterns one at a time from the row patterns, applies the necessary transformations using the above-mentioned operators, and stores the results of each column pattern transformation in a transformation queue. Then it proceeds to apply transformations to the remaining column patterns. Once all transformations are complete, it applies the preferred ones from the queue to the corresponding data rows. For example, Figure 4a shows wanted rows of the CSV file of Figure 1 where values are shifted either due to the absence of an escape character or incorrectly quote character. To address this inconsistency, MORPHER uses the operators Merge and Reformat. The resulting cleaned rows are presented in Figure 4b, where MORPHER follows the RFC 4180 standard [11] specifications to standardize the incorrect quote and escape characters. Moreover, MORPHER can deal with a wide range of formatting problems, as detailed in Section 1. As a final result, MORPHER outputs a clean and structured file.

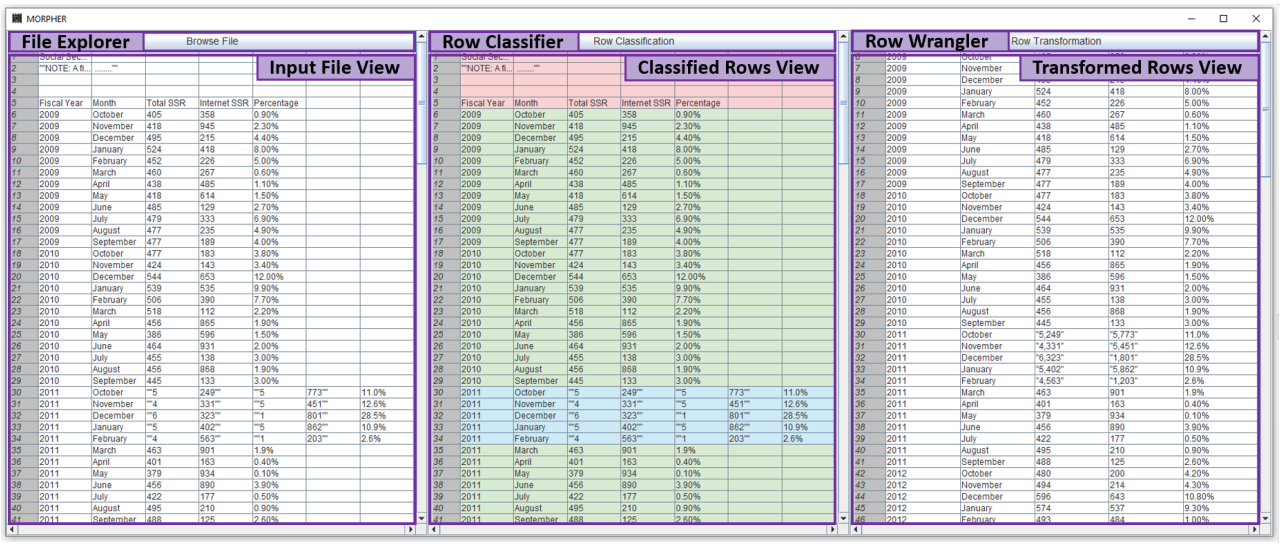


Figure 5: Morpher's desktop-based user interface

3 DEMO WALK-THROUGH

We present the simple but effective Morpher GUI through a demonstration scenario shown in Figure 5. To explore and interact with the system, users can download[3] Morpher as a desktop application, along with a set of 148 exemplary raw CSV files obtained from four different open data sources.

We present a use-case in which a data scientist analyzes a company's growth and gains insights from the statistics in a CSV file. To begin the analysis, the scientist might need to load the file into a database connected to the analytics platform. Unfortunately, the file contains ill-formed rows that disrupt the file ingestion process. After encountering a halt during the file loading, the data scientist begins a manual inspection to narrow down the causes. However, due to the sheer amount of data and the complexity of the file, the data scientist may have overlooked some issues or needed help finding them amidst the wall of characters. The data scientist may have initially identified one or a few ill-formed rows, but cannot assume those are the only issues in the file. While some ill-formed rows may be easy to spot by simply scrolling through a file, such as those containing preambles or footnotes, others are more complicated to recognize, such as rows containing cell values with user-specific dialect details (non-standardized) or those containing additional metadata, making it challenging for even expert users to identify them manually. To ensure accurate analytics and reliable results, thoroughly inspecting the entire file to identify all problematic rows is crucial.

To overcome these challenges, the data scientist utilizes the functionalities of Morpher, our unsupervised tool designed to detect and clean ill-formed rows in a CSV file automatically. The tool's GUI includes a ***file viewer***, which facilitates navigation and viewing of the input file. Using the GUI, with one click the data scientist runs Morpher's automatic ***row classifier***, which assigns each row a label as either well-formed (i.e., clean), ill-formed but wanted (i.e., erroneous but can be cleaned), or ill-formed unwanted (i.e., non-data row). Figure 5 displays the output of the row classification module on the tool's GUI, where each row is color-coded based on its classification label for ease of interpretation.

After classification, the data scientist executes the final stage of Morpher's pipeline: ***row wrangler***, automatically cleaning the ill-formed wanted rows with a single click. The transformed and standardized output file, as shown in Figure 5, can then be exported in both CSV and Excel formats, providing the data scientist with a handy resource for their intended task. Moreover, users have the convenience of executing the entire process through the command line interface.

The ability to efficiently identify and clean ill-formed rows streamlines the data processing pipeline and allows data scientists and machine learning engineers to spend more time on higher-level tasks, such as modeling and analytics.

4 CONCLUSION

Morpher is an innovative tool that addresses a common data preparation challenge that data scientists face when working with CSV files – the presence of ill-formed rows. Through its intuitive GUI, Morpher simplifies identifying and cleaning ill-formed rows, allowing data scientists to navigate and parse their files more efficiently. Morpher automates a crucial step in data preparation, freeing up time and resources to be better spent during the later stages of a data processing pipeline. Overall, Morpher represents a valuable contribution to the field of data science and has the potential to benefit a wide range of researchers and practitioners.

ACKNOWLEDGMENTS

This research was funded by the HPI research school on Data Science and Engineering.

[3] https://github.com/HMazharHameed/MORPHER

REFERENCES

[1] Marco D Adelfio and Hanan Samet. 2013. Schema extraction for tabular data on the web. *PVLDB* 6, 6 (2013), 421–432.

[2] Sara Bonfitto, Luca Cappelletti, Fabrizio Trovato, Giorgio Valentini, and Marco Mesiti. 2021. Semi-automatic column type inference for CSV table understanding. In *International Conference on Current Trends in Theory and Practice of Informatics*. Springer, 535–549.

[3] Christina Christodoulakis, Eric B Munson, Moshe Gabel, Angela Demke Brown, and Renée J Miller. 2020. Pytheas: pattern-based table discovery in CSV files. *PVLDB* 13, 12 (2020), 2075–2089.

[4] Till Döhmen, Hannes Mühleisen, and Peter Boncz. 2017. Multi-hypothesis CSV parsing. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*. 1–12.

[5] Chang Ge, Yinan Li, Eric Eilebrecht, Badrish Chandramouli, and Donald Kossmann. 2019. Speculative distributed CSV data parsing for big data analytics. In *Proceedings of the International Conference on Management of Data (SIGMOD)*. 883–899.

[6] Martin Gollery. 2005. Bioinformatics: sequence and genome analysis. *Clinical Chemistry* 51, 11 (2005), 2219–2220.

[7] Inc. Google. 2021. *OpenRefine*. www.openrefine.org (last accessed August 30th, 2022).

[8] Mazhar Hameed and Felix Naumann. 2020. Data Preparation: A Survey of Commercial Tools. *SIGMOD Record* 49, 3 (2020), 18–29.

[9] Mazhar Hameed, Gerardo Vitagliano, Lan Jiang, and Felix Naumann. 2022. SURAGH: Syntactic Pattern Matching to Identify Ill-Formed Records.. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*. 143–154.

[10] Yeye He, Xu Chu, Kris Ganjam, Yudian Zheng, Vivek Narasayya, and Surajit Chaudhuri. 2018. Transform-data-by-example (TDE) an extensible search engine for data transformations. *PVLDB* 11, 10 (2018), 1165–1177.

[11] IETF 2005. RFC 4180. https://tools.ietf.org/html/rfc4180. (last accessed February 7th, 2023).

[12] Trifacta Inc. 2021. *Trifacta Data Engineering Cloud*. www.trifacta.com (last accessed August 30th, 2022).

[13] Lan Jiang, Gerardo Vitagliano, and Felix Naumann. 2021. Structure Detection in Verbose CSV Files. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*. 193–204.

[14] Johann Mitlöhner, Sebastian Neumaier, Jürgen Umbrich, and Axel Polleres. 2016. Characteristics of open data CSV files. In *Proceedings of the International Conference on Open and Big Data (OBD)*. IEEE, 72–79.

[15] Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology* 48, 3 (1970), 443–453.

[16] National Library of Medicine. 2022. *BLAST*. https://blast.ncbi.nlm.nih.gov/Blast.cgi (last accessed February 7th, 2023).

[17] LLC Tableau Software. 2022. *Tableau*. www.tableau.com (last accessed August 30th, 2022).

[18] Gerrit JJ van den Burg, Alfredo Nazábal, and Charles Sutton. 2019. Wrangling messy CSV files by detecting row and type patterns. *Data Mining and Knowledge Discovery* 33, 6 (2019), 1799–1820.

[19] Gerardo Vitagliano, Mazhar Hameed, Lan Jiang, Lucas Reisener, Eugene Wu, and Felix Naumann. 2023. Pollock: A Data Loading Benchmark. *PVLDB* 16, 8 (2023), 1870–1882.