Detecting Layout Templates in Complex Multiregion Files

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

Lan Jiang
lan.jiang@hpi.de
Hasso Plattner Institute, University of Potsdam, Germany

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

ABSTRACT

Spreadsheets are among the most commonly used file formats for data management, distribution, and analysis. Their widespread employment makes it easy to gather large collections of data, but their flexible canvas-based structure makes automated analysis difficult without heavy preparation. One of the common problems that practitioners face is the presence of multiple, independent regions in a single spreadsheet, possibly separated by repeated empty cells. We define such files as "multiregion" files. In collections of various spreadsheets, we can observe that some share the same layout.

We present the Mondrian approach to automatically identify layout templates across multiple files and systematically extract the corresponding regions. Our approach is composed of three phases: first, each file is rendered as an image and inspected for elements that could form regions; then, using a clustering algorithm, the identified elements are grouped to form regions; finally, every file layout is represented as a graph and compared with others to find layout templates. We compare our method to state-of-the-art table recognition algorithms on two corpora of real-world enterprise spreadsheets. Our approach shows the best performances in detecting reliable region boundaries within each file and is able to correctly identify recurring layouts across files.

PVLDB Reference Format:
Gerardo Vitagliano, Lan Jiang, and Felix Naumann. Detecting Layout Templates in Complex Multiregion Files. PVLDB, 15(1): XXX-XXX, 2022.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/vitaglianog/mondrian_vldb.

1 STRUCTURAL FILE TEMPLATES

Data comes in all shapes and forms. The recent blossom of open data portals has made large quantities of spreadsheet files available for public consumption [9, 18, 20]. It is common knowledge that much human time and effort in data-oriented workflows are spent on preparing data files. In fact, even spreadsheets that are meant for distribution and analysis can be affected by data quality issues and human-induced errors that make information extraction difficult [5, 12]: they are often used as canvases in which data is spread out in multiple, independent regions with a custom layout and without a well-defined tabular format. In many cases there are multiple tables, but metadata regions are also common, e.g., spreadsheet titles, comment sections, or notes to data cells.

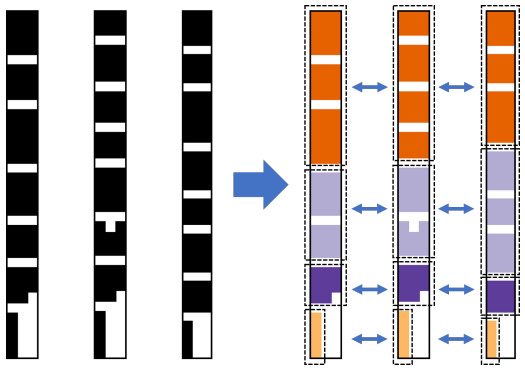


Figure 1: Visual rendering of three different files sharing the same multiregion layout.

As an example, Figure 1 depicts the visual structure of three different spreadsheet files from the FUSE corpus [20]: they all contain the same three tables (albeit with different data points) and a footnote region, arranged in the same layout. Due to missing values and empty rows, it is difficult not only to draw the correct table boundaries within one file, but also to recognize that the three files share the same layout.

In large collections, such as enterprise data lakes or open data repositories, multiple data files may follow the same "layout template" that dictates the number, schema and visual distribution of tables and metadata regions (for a more formal definition, cf. Section 3.2). In the light of a data-oriented workflow, the repeated occurrence of the same layout in large collections of files is a valuable resource to exploit, for example to assist data exploration, to automate data preparation, or to perform data integration.

However, no existing approach considers detecting multiregion layout templates. Previous research addressed the problems of correctly detecting and recognizing tabular regions in single spreadsheets [3, 6, 16], yet, multiregion layouts are a common occurrence across spreadsheet data sources. In the Deco dataset [14], an annotated sample of 854 files from the ENRON Excel corpus, almost 75% of the sheets (621) contain more than one region with 71 layouts recurring in more than one file; in a randomly sampled subset of 886 files from the FUSE spreadsheet dataset, annotated by the authors of this paper, almost half of them (391) show multiple regions, with 31 recurring layouts; Mitlöhner et al. reported that, out of 141k csv files retrieved from open data portals, roughly 3% of the correctly parsed files contained more than one table, and 4.6% of those that could not be correctly parsed were showing "too many tables" [18].

Proceedings of the VLDB Endowment, Vol. 15, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

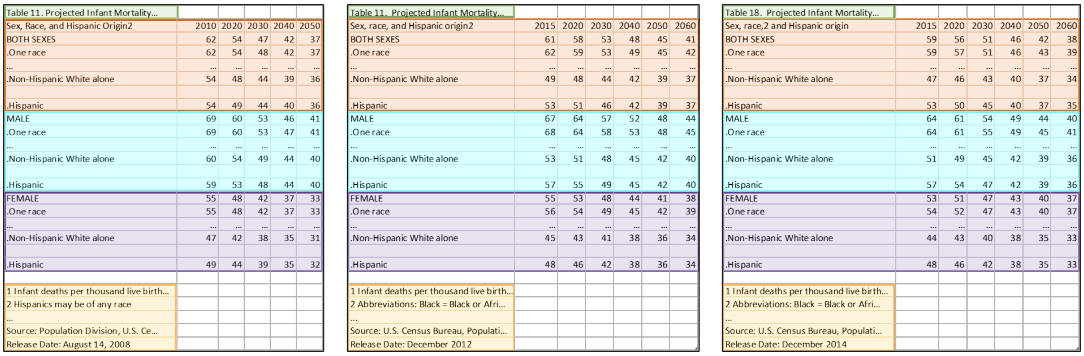


Table 11. Projected Infant Mortality...					
Sex, Race, and Hispanic Origin2	2010	2020	2030	2040	2050
BOTH SEXES	62	54	47	42	37
.One race	62	54	48	42	37
...	...	...	...	...	...
.Non-Hispanic White alone	54	48	44	39	36
.Hispanic	54	49	44	40	36
MALE	69	60	53	46	41
.One race	69	60	53	47	41
...	...	...	...	...	...
.Non-Hispanic White alone	60	54	49	44	40
.Hispanic	59	53	48	44	40
FEMALE	55	48	42	37	33
.One race	55	48	42	37	33
...	...	...	...	...	...
.Non-Hispanic White alone	47	42	38	35	31
.Hispanic	49	44	39	35	32
1 Infant deaths per thousand live birth...					
2 Hispanics may be of any race					
...					
Source: Population Division, U.S. Ce...					
Release Date: August 14, 2008					

Table 11. Projected Infant Mortality...						
Sex, race, and Hispanic origin2	2015	2020	2030	2040	2050	2060
BOTH SEXES	61	58	53	48	45	41
.One race	62	59	53	49	45	42
...	...	...	...	...	...	...
.Non-Hispanic White alone	49	48	44	42	39	37
.Hispanic	53	51	46	42	39	37
MALE	67	64	57	52	48	44
.One race	68	64	58	53	48	45
...	...	...	...	...	...	...
.Non-Hispanic White alone	53	51	48	45	42	40
.Hispanic	57	55	49	45	42	40
FEMALE	55	53	48	44	41	38
.One race	56	54	49	45	42	39
...	...	...	...	...	...	...
.Non-Hispanic White alone	45	43	41	38	36	34
.Hispanic	48	46	42	38	36	34
1 Infant deaths per thousand live birth...						
2 Abbreviations: Black = Black or Afri...						
...						
Source: U.S. Census Bureau, Populati...						
Release Date: December 2012						

Table 18. Projected Infant Mortality...						
Sex, race,2 and Hispanic origin	2015	2020	2030	2040	2050	2060
BOTH SEXES	59	56	51	46	42	38
.One race	59	57	51	46	43	39
...	...	...	...	...	...	...
.Non-Hispanic White alone	47	46	43	40	37	34
.Hispanic	53	50	45	40	37	35
MALE	64	61	54	49	44	40
.One race	64	61	55	49	45	41
...	...	...	...	...	...	...
.Non-Hispanic White alone	51	49	45	42	39	36
.Hispanic	57	54	47	42	39	36
FEMALE	53	51	47	43	40	37
.One race	54	52	47	43	40	37
...	...	...	...	...	...	...
.Non-Hispanic White alone	44	43	40	38	35	33
.Hispanic	48	46	42	38	35	33
1 Infant deaths per thousand live birth...						
2 Abbreviations: Black = Black or Afri...						
...						
Source: U.S. Census Bureau, Populati...						
Release Date: December 2014						

Figure 2: Detailed view of three US Census files sharing the same multiregion layout.

What is more, previous approaches for automated table extraction in spreadsheet usually rely on format-specific style features. However, files are more often shared in .csv format. For example, of 15,497 files distributed on the UK open data portal (data.gov.uk), 44.18% are in .csv format, compared to 8.81% in an Excel-specific format (.xls/.xlsx). The same trend is true for the US open data portal (data.gov), where of 192,335 datasets, 9.61% have a "csv" tag, while only 3.19% have an "excel" tag.

Therefore, we design our approach, Mondrian, to be general with respect to file format, ignoring rich-text features as encoded in Excel files. While additional metadata, such as file and/or sheet names could also prove useful for the purpose of template detection, we observe that these can be unreliable and/or unavailable in real-world scenarios (consider, e.g., how often sheets are labeled "Sheet1", or files are machine-named). Our intuition is to leverage the visual distribution and the literal content of individual cells by converting each file into an image and segment it to find heterogeneous regions: first, we graphically identify individual segments of adjacent data, and then we partition them to have finer-grained elements to cluster together. Once regions have been detected, file layouts are described as graphs and compared using a similarity flooding-based algorithm to find layout templates. The graphical rendering of a template inspired us to name our approach after the abstract painter Piet Mondrian. In proposing Mondrian, we make the following contributions:

(1) An unsupervised approach that leverages a novel mapping between spreadsheets and the visual image domain to detect and match different regions in spreadsheet files.
(2) A framework to analyze and compare multiregion spreadsheets, using a graph representation with an associated similarity algorithm to detect layout templates.
(3) A publicly available dataset of structural annotations for 886 spreadsheets, classifying the position and purpose of their composing regions, and a set of template annotations for two datasets, summing up to a total of above 1500 files, identifying classes of files with the same layout.
(4) A comprehensive set of experiments to prove the effectiveness of the Mondrian approach in solving the region detection and template inference problems, evaluating and comparing it with state-of-the-art automated methods.

2 MOTIVATING EXAMPLE

This section introduces a real-world example to highlight one of the possible use-cases of Mondrian. Consider the historical population data of the United States Census, made publicly available through an open data portal[1]. The data from each year is summarized in different tables contained in spreadsheet files, and although some tables are unique to specific years, others recur in multiple years. The files that contain the same tables all share the same layout: they have similar title and footnote cells, and all their tables (when more than one) have the same schema.

For example, the three files in Figure 2 contain data about projected infant mortality (some rows excluded for visual clarity). All have three tables, a title, and a footnote region, arranged with the same layout. However, there are slight differences in the files across years. For example, in the footnote region the last cell reflects the year, and sometimes cells have different content while the semantic meaning is the same (E.g. "Source: Population Division, U.S. Census Bureau" and "Source: U.S. Census Bureau, Population division"). The tables themselves have a different number of columns across files, and also their headers are updated. Finally, the table title also changes from "Table 11" to "Table 18". Nonetheless, it is obvious at a glance that the three files come from the same layout template.

With manual human inspection and domain knowledge, it is possible to consolidate tables from the same templates into a single source of truth to enable downstream tasks, after some necessary data preparation steps, e.g. to remove empty lines. Without Mondrian, these steps have to be carried out manually for each file, becoming more and more cumbersome and time-consuming the larger the set of input files.

With Mondrian, it is possible to leverage the recurring structure of the templates and prepare at once all files that belong to a template. In the US Census example, out of 99 spreadsheets, in the span of few minutes, our system identifies the layouts of every file and groups them into fifteen different templates. For example, for the three files of Figure 2, Mondrian detects the region boundaries for each file layout, identifies that all layouts belong to the same template, and determines that different regions across different files are equivalent. Using the results of Mondrian, end-users may perform template-wide transformations, for example deleting all title and

[1] https://www2.census.gov/programs-surveys/popproj/tables/ (accessed 25/05/21)

footnote regions, separate the tables, and remove all empty rows without going through the different files.

3 DESCRIBING MULTIREGION LAYOUTS

Before describing the details of our solution, we provide definitions for the concepts of multiregion files, layouts and layout templates. Typically, multiregion files can be found in comma-separated values format (*.csv*) or Microsoft Excel format (*.xls/.xlsx*). Complex layouts with multiple regions are a byproduct of spreadsheet software rendering data on "canvases" where users freely lay out different data (and possibly metadata).[2] Here, we first formalize the concepts needed to describe the layout of multiregion files. Then, we formulate a hierarchy of equivalence notions to compare layouts and their composing parts. Finally, we state the research problems addressed by our approach.

3.1 Multiregion spreadsheets

Our sources of data are spreadsheets, defined as value-delimited files that contain data in cells with a grid structure. We assume no specific row- or column-based structure of the content.

We assign each cell a unique identifier (x, y), where $x, y \in \mathbb{N}_0$ correspond to the column and row indices, respectively. We can consider these (x, y) coordinates as points in a Euclidean space with its origin at the top-left corner, in analogy to spreadsheet design. Every cell serves some purpose in the spreadsheet. We consider three fundamental types of cells:

Definition 1 (Cell types)**.** A cell c of a spreadsheet S belongs to one of the following mutually disjoint cell types:

(1) **Data**, if it carries the data values of a file;
(2) **Metadata**, if its information is related to a set of data cells;
(3) **Empty**, if it does not contain any data or only whitespace characters, e.g., it is used for visual formatting.

Elements are simple structures, grouping cells of the same type:

Definition 2 (Element)**.** Given a spreadsheet file S, an element e is a rectangular set of adjacent cells of S of the same type. The element type of e corresponds to the cell type of its cells.

According to its position in the spreadsheet, an element can be described with the vector $(x_0, y_0, x_1, y_1) \in \{\mathbb{N}_0\}^4$, where the coordinates (x_0, y_0) represent an element's top-left cell and (x_1, y_1) its bottom-right cell.

Since elements are groups of adjacent cells, in a given spreadsheet we can identify several of them and describe their spatial relationships. Considering the elements' rectangular nature and the grid-like space of spreadsheets, we encode the relationship between two elements with three features: alignment direction, alignment magnitude, and distance. The alignment direction is based on the overlap of the elements' projection on the x-axis and the y-axis:

[2]Some formats and tools allow a spreadsheet to have more than one "worksheet". Without loss of generality, we consider each worksheet as a separate file.

Definition 3 (Alignment)**.** Two elements $a := (a_{x_0}, a_{y_0}, a_{x_1}, a_{y_1})$, $b := (b_{x_0}, b_{y_0}, b_{x_1}, b_{y_1})$ are aligned:

$$\begin{cases} \textit{Vertically (V)} & \text{if } max(a_{y_0}, b_{y_0}) \leq min(a_{y_1}, b_{y_1}) \\ \textit{Horizontally (H)} & \text{if } max(a_{x_0}, b_{x_0}) \leq min(a_{x_1}, b_{x_1}) \\ \textit{Not aligned (N)} & \text{otherwise} \end{cases}$$

It is worthwhile noting that, as they are adjacent groups of cells, the areas of two any given elements in a spreadsheet cannot overlap. The alignment magnitude is the number of shared points across the axis in case of horizontal or vertical alignment:

Definition 4 (Alignment magnitude)**.** The alignment magnitude between elements a, b is:

$$\begin{cases} \min(a_{y_1}, b_{y_1}) - \max(a_{y_0}, b_{y_0}) + 1 & \text{if } \textit{alignment}(a, b) = V \\ \min(a_{x_1}, b_{x_1}) - \max(a_{x_0}, b_{x_0}) + 1 & \text{if } \textit{alignment}(a, b) = H \\ 0 & \text{otherwise} \end{cases}$$

The distance between the elements is calculated as the distance of their two closest points. In case the two elements are horizontally or vertically aligned, this resolves to the distance between their closest boundaries; otherwise, it is calculated as the Euclidean distance of the two closest corners:

Definition 5 (Distance)**.** The distance between elements a, b is:

$$\begin{cases} d_v = |\min(a_{x_1}, b_{x_1}) - \max(a_{x_0}, b_{x_0}) + 1| & \text{if } \textit{alignment}(a, b) = V \\ d_h = |\min(a_{y_1}, b_{y_1}) - \max(a_{y_0}, b_{y_0}) + 1| & \text{if } \textit{alignment}(a, b) = H \\ d_o = \sqrt{d_v^2 + d_h^2} & \text{otherwise} \end{cases}$$

Often, especially in spreadsheets with complex cell layouts, even non-adjacent cells could be logically grouped. For example, a table may have missing values that result in empty rows in-between valid data rows (As seen in Figure 1). Elements are therefore not sufficient to completely describe the layout of a spreadsheet, and we need a higher-order abstraction to group semantically related elements, which are not necessarily adjacent to each other. As we are concerned with the structural layout of files, we deliberately leave the "semantic relatedness" property without a clear definition. Groups of elements can serve different purposes: examples are tables, preambles, footnotes, or any other domain-specific construct. To abstract their specific purpose, we identify them as regions:

Definition 6 (Region)**.** A region R is a complete graph having as nodes a set of semantically related, non-empty elements $\mathcal{E}$, connected with edges labeled with their pairwise spatial relationships.

One example of such a graph is shown in Figure 3: the header element is horizontally aligned to all three data elements. One data element is adjacent to the header and aligned for three cells, therefore the edge connecting the two nodes has the label (H,3,0), which implies a horizontal alignment with a magnitude of 3 cells and a distance of 0 cells. In contrast, the data element with the cell containing the string "Genesee" is horizontally aligned with the header for one cell and is 3 cells distant to it, therefore the edge connecting the two nodes has a label of (H,1,3).

Considering the definition of regions, a multiregion spreadsheet is trivially defined as a spreadsheet containing multiple regions.

Ultimately, our goal is to find structural similarity across different, possibly multiregion files. To do so, it is first important to

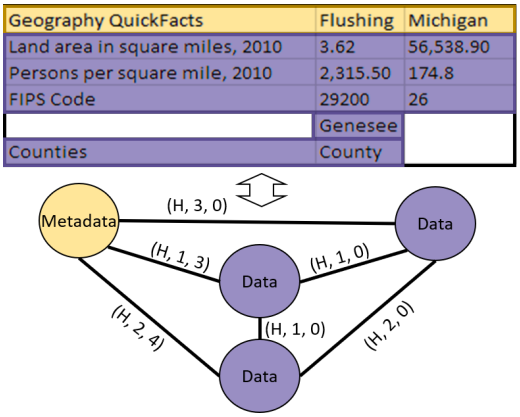


Figure 3: A region layout and its graph representation.

identify a "meaningful" set of regions for each file: that is to say, draw the boundaries of different regions such that they are independent and serve distinct purposes. To describe the coordinates of a region boundary in the spreadsheet space, we use the bounding box of its set of elements:

Definition 7 (Region boundary)**.** The boundary of a region R, with its elements $\mathcal{E}$, is defined as a rectangle (x_0, y_0, x_1, y_1), where:

$$x_0 = \min_{e \in \mathcal{E}} e_{x_0}, \quad y_0 = \min_{e \in \mathcal{E}} e_{y_0}, \quad x_1 = \max_{e \in \mathcal{E}} e_{x_1}, \quad y_1 = \max_{e \in \mathcal{E}} e_{y_1}$$

Once regions have been identified, we are concerned with their layout. We naturally extend the spatial relationship feature vector, defined for pairs of elements, to pairs of regions, using the (x_0, y_0, x_1, y_1) coordinates of their boundaries to compute alignment direction, magnitude, and distance.

One caveat is that considering their boundaries, two given regions can, in general, have overlapping bounding boxes, which is not the case for elements. We extend the spatial relationship feature vector for overlapping regions as:

Definition 8 (Overlapping regions)**.** Given two regions, $A := (a_{x_0}, a_{y_0}, a_{x_1}, a_{y_1})$ and $B := (b_{x_0}, b_{y_0}, b_{x_1}, b_{y_1})$, their alignment direction is "overlapping" (O) if $\max(a_{y_0}, b_{y_0}) \leq \min(a_{y_1}, b_{y_1})$ and $\max(a_{x_0}, b_{x_0}) \leq \min(a_{x_1}, b_{x_1})$. Then, the alignment magnitude is $(\min(a_{y_1}, b_{y_1}) - \max(a_{y_0}, b_{y_0}) + 1) \cdot (\min(a_{x_1}, b_{x_1}) - \max(a_{x_0}, b_{x_0}) + 1)$ and the distance is 0.

The magnitude corresponds to the area of the overlap, which ultimately equals the product of the horizontal and vertical alignment magnitudes, considering that two overlapping regions are both horizontally and vertically aligned.

Finally, describing a set of non-empty regions with a complete graph, we can define the layout of a spreadsheet:

Definition 9 (Spreadsheet layout)**.** The layout of a spreadsheet file S is a complete graph having as nodes its set of non-empty regions, connected with edges labeled with their pairwise spatial relationship.

3.2 Templates as recurring structures

Often, region and file layouts are not one-off models but stem from a systematic creation process. For example, the US Census open data portal contains the same data report for multiple geographical entities, each downloadable as a separate CSV file[3]. Our goal is to provide a framework to define and analyze *templates*, i.e., classes of structural equivalence across multiple files. We compose a hierarchy of equivalence notions, beginning with the finest-grained unit of comparison, the cell, and extend it to elements:

Definition 10 (Cell equivalence)**.** Two cells c_1, c_2 are equivalent if their type and content are equal. Two empty cells are always equivalent.

Definition 11 (Element equivalence)**.** Two elements e_1, e_2 are equivalent if their types are the same and if there is a one-to-one equivalence between their cells, regardless of their position. Two empty elements are always equivalent.

Similar to cells, we consider empty elements equivalent, regardless of their shape, as their purpose is to provide visual information about region layout to end-users. Recalling Definition 6, this information is encoded within the attributes of the edges (i.e., the spatial relationship between nodes) of a region graph. We define element equivalence to be insensitive of cell position to be able to match elements that have equal content differing only for their layout, e.g., two tables with the same column in a different position. To define region equivalence, we must also be able to include regions with equal structure but different data values, e.g., two tables with the same schema but different data.

Definition 12 (Region equivalence)**.** Two regions R_1, R_2 are equivalent if there is a one-to-one equivalence between their metadata nodes and their graphs are isomorphic.

At the spreadsheet level, the definition for layout is similar:

Definition 13 (Layout equivalence)**.** Two layouts L_1, L_2 are equivalent if there is a one-to-one equivalence between their regions and their graphs are isomorphic.

In practice, if many files are collected from several different sources, we want to be able to discover entire sets of equivalent spreadsheets:

Definition 14 (Layout template)**.** A layout template $\mathcal{L}$ is a class of equivalent file layouts.

Recognizing templates is of great value for data preparation, as it potentially saves users the time to manually inspect and prepare individual files: a pipeline of preparation steps can be defined once and executed repeatedly on different files from the same template. To find templates, in Section 4 we relax the stricter equivalence assumptions and use approximate similarity metrics.

3.3 Automated layout inference

Given the definitions stated, the problem of recognizing and matching multiregion spreadsheet layouts is composed of several distinct sub-problems that have an inherently visual nature. The first fundamental problem is to find the correct region boundaries. A human expert would solve this task by understanding the semantics of the data as well as its spatial distribution. Then, to identify recurring layouts, they would be required to manually inspect and compare

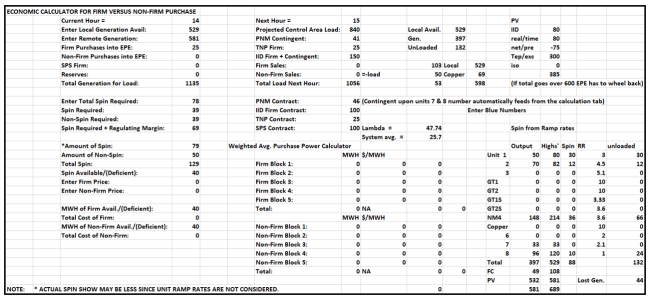
[3] https://www.census.gov/quickfacts/, last accessed Sep. 07, 2020

each separate file looking at its data – a cumbersome, error-prone, and time-consuming task. According to our definitions of equivalence, this task requires semantic concepts and possibly domain knowledge, e.g., to distinguish table schemata. However, to design a general and domain-independent approach, we focus only on structural properties. We present the Mondrian approach to automatically detect multiregion layout templates. In doing so, we address the following research problems:

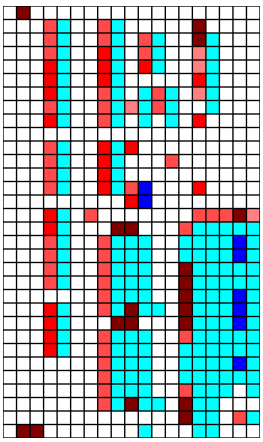
(1) *Region detection:* How to detect the correct boundaries of the regions R that compose a multiregion file?
(2) *Region matching:* Given two regions from one or multiple files, how to approximate their equivalence without semantic information?
(3) *Template inference:* Given a set of different files, how to measure the similarity of their layouts and group them in templates?

4 THE MONDRIAN APPROACH

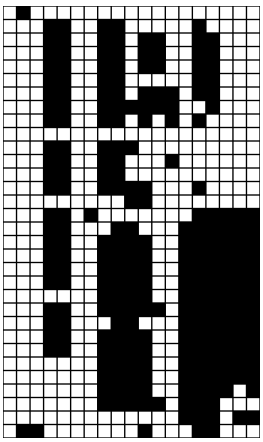
To identify the conceptual entities defined in Section 3.1 in practice, without resorting to semantic knowledge, the intuition of Mondrian is to transform the domain of spreadsheets from data content to image. We convert cells into pixels, encoding their syntactical types into colors. Then, we find elements by segmenting the file images with a partitioning algorithm and cluster them to detect region boundaries. Once regions are identified, we analyze their structural properties and use a similarity measure to match regions across different files. If two (or more) files are found having similar regions, we measure the similarity between the graph representations for their layouts and possibly group them into a template.



a. A file of the ENRON corpus viewed in a spreadsheet software.



b. Spreadsheet image parsing.



c. Connected components.

Figure 4: Core intuition of Mondrian – transposing a spreadsheet to the image domain.

4.1 Image parsing and segmentation

To cover the most general cases, our approach takes as input comma-separated value files. Files with different delimiters or formatted with XML markup, such as Microsoft Excel files, can be easily converted into a '.csv' file. Ignoring possible markup information is the trade-off for a method applicable to a wide spectrum of spreadsheets, independent of their format specifications.

For native csv files, we cannot assume that all rows have the same number of delimiters. Thus, we pad rows with empty cells up to the length of the longest row. Given a csv file with M rows and N columns, we create an image with the dimensions $M \times N$, where each pixel represents a cell in the csv file. Our definitions of entities and their equivalence build upon the concept of "cell type": in practice, we substitute semantic types with *syntactic types* and, correspondingly relax their equivalences into an approximate *structural similarity*.

We identify four fundamental syntactic types: *number*, *datetime*, *string*, *empty*. Except for *empty*, each of these types can be further refined in sub-types: a *number* can be *integer* or *floating point*; a *datetime* can be a *time* or a *date*; a string can be either *uppercase*, *lowercase*, *titlecase* or *generic*. In parsing the spreadsheet as an image, we transform every cell into a pixel with a different color according to its type (Cf. Figures 4a and 4b). Table 1 shows the color corresponding to each data type and a sample cell from Figure 4a[4] that was parsed according to that type.

Type	Specific type	Sample cell	Color
Empty	Empty	" "	White
Number	Integer	"14"	Light Blue
	Floating point	"47.74"	Dark Blue
Datetime	Time	"17:00"	Light Green
	Date	"17/9/20"	Dark Green
String	Uppercase	"MWH"	Maroon
	Lowercase	"real/time"	Salmon Red
	Titlecase	"Firm Sales"	Tomato Red
	Generic	"System avg. ="	Scarlet Red

Table 1: Data types and their colors.

Recognizing the syntactic type of a cell without semantic knowledge is, in general, a coarse-grained and error-prone operation: consider the uncertain nature of the value "1990", which can be a date or a number. As our experiments in Section 5.4 demonstrate, however, a coarse-grained parsing is sufficient to approximate region equivalence for the task of template inference, with the reasonable assumption that any parsing mistake would be reflected across all similar files.

[4]Except for the time and date types, which were not present in the original file.

To segment the file into elements, we first find connected components, which reflect cell aggregates that could not be so easily recognized in a spreadsheet software view (Figure 4c). The change in width/height proportion is because in the image each cell occupies one square pixel, while in the spreadsheet software cell columns and rows can have different widths or heights, usually set according to the length of their values. With this "cell normalization", for example, a human observer is more likely to note the four aligned vertical elements on the left of the image.

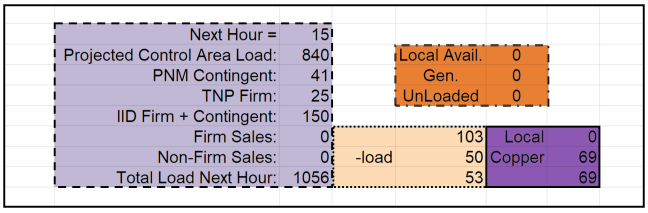
However, considering connected components as elements could lead to incorrect region boundaries: as highlighted by Figure 5a, sometimes regions can be adjacent to each other. In the example, different rectangular regions compose a single connected component with irregular edges (Figure 5b). Therefore, to identify a valid set of elements that leads to correct region boundaries, we need a segmentation strategy for connected components. We "cut" the connected components along their non-concave edges (Figure 5c).

Formally, we partition the components following a *rectilinear* cut that is obtained by extending the edges incident to concave vertices towards the interior of the polygon, until a polygon boundary is met. Bajuelo et al. show that each given polygon, with v concave vertices, can be split into $O(v^2)$ elements, with $2v + 1$ as a minimum [1].

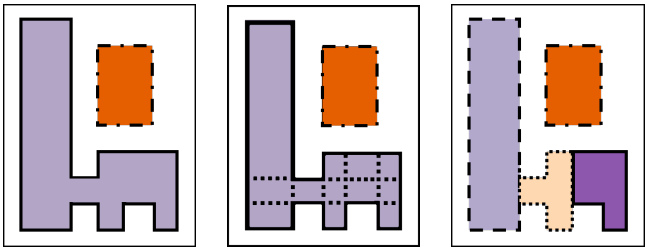
With this method, even coherent elements could be initially decomposed. This is eventually corrected while searching for regions in the next phase – clustering – where finer-grained elements can be either merged or not, granting the ability to even discover regions that appear directly adjacent in the spreadsheet (Figure 5d).

4.2 Region detection and matching

The next phase of Mondrian has the objective of clustering together elements that belong to the same region. For a given spreadsheet, we have no prior knowledge of the number of regions that it contains. Thus, we cannot use *centroid-based* clustering approaches,



a. Detail of Figure 4a highlighting adjacent, independent regions.



b. Connected components. **c. Cutting into finer grained elements.** **d. Clustering results.**

Figure 5: Partitioning is necessary to detect adjacent tables.

such as k-means. Instead, we resort to a customized *density-based* approach, modifying DBSCAN [8] to operate with a custom distance metric that highlights the structural properties we seek for. The DBSCAN optimization problem aims at finding points in dense neighborhoods of a given space: if we consider spreadsheet elements as points, a region corresponds to an area with a high density of points. Given a distance function and a minimum number of points m that form a cluster, the algorithm defines as *core points* of a cluster all those that have at least m points closer than a threshold ε, also called *radius* of the search space. Then, it groups all points that are within ε from a core point, or within ε from non-core points belonging to a cluster.

In the original DBSCAN algorithm, every leftover point is labeled as noise. In our scenario, we are interested in labeling all the elements of a spreadsheet. Therefore we do not consider any element as noise and set the minimum number of elements that can form a region as $m = 1$. The distance function we use to compare elements is a weighted sum of three terms:

(1) **Distance**: The Euclidean distance of their closest cells (Definition 5).
(2) **Size difference**: Considering a_0, a_1 as the areas of two elements, with the larger being a_1, the size difference is $1 - a_0/a_1$.
(3) **Alignment magnitude**: The number of shared points across the horizontal or vertical axis (Definition 4).

The weights for these terms are α, β, γ, respectively, and can be fine-tuned globally or for a given spreadsheet as hyperparameters for optimal boundary detection. Additionally, the value of the radius ε plays an important role in the success of the clustering as different files can have different properties regarding the size of regions and the mutual distances of their elements. We hypothesize that larger spreadsheets have, on average, a higher number of elements with greater distances, and therefore benefit from larger radii. As Section 4.2 points out, the best performances are obtained when setting a custom radius for each file. To reflect a scenario with no specific hyperparameter selection, we also experimented with our approach to find a suitable fixed hyperparameter setting for all files.

Once their boundaries have been identified, we are interested in equivalent regions. Our region equivalence definition (Definition 12), is based on element boundaries and their types: for example, two footnote regions are equivalent if their entire content is equal, while two tabular regions are equivalent if their header elements are the same, regardless of the actual data content.

As Mondrian lacks semantic knowledge about cell types and relies on image segmentation and clustering to identify element and region boundaries, we need a suitable similarity measure to estimate actual equivalence. Our approximate region similarity is therefore based on syntactic cell types and their color encoding. Note from Table 1 how our color encoding assigns one primary color (red, green, blue, white) to each fundamental data type and then varying shades of the primary color to each sub-type belonging to the same fundamental data type. For example, *string* is associated with red, with *lowercase* being "tomato red" (RGB (255, 75, 75)) and *titlecase* being "scarlet red" (RGB (255, 0, 0)).

In this way, cells with the same fundamental data type but different sub-types are more similar in the color space than cells from

different fundamental types. A given region can be described with the color histograms of its cells, computed with 64 bins for each channel, for a total of 192 bins. The color histogram is a global descriptor of each region that acts as a region "fingerprint": its values are dependent on the amount and distribution of cells of different types. The similarity of any two regions can then be computed as the cross-correlation of their color histograms. Furthermore, the color encoding can be easily extended including more, or further refined, data types. If two highly similar regions (that is, whose similarity is over a given threshold) are found in two different files, they are considered equivalent and the file layouts that contain them are candidate instances of the same template (Cf. Section 4.4 for a detailed explanation).

4.3 Layout similarity

Each spreadsheet file, once its regions have been detected, has an associated file layout, represented as a complete graph with regions as nodes and labeled edges that describe their spatial relationships (Definition 9). As we did with regions, we resort to a similarity measure to approximate layout equivalence. Our algorithm is based on the similarity flooding approach proposed by Melnik et al. for graph matching [17]. The core intuition is to first compute an initial pair-wise similarity of nodes across the two file layout graphs using the region similarity metric described in Section 4.2. If the graph $\mathcal{G}_a$ has U nodes and the graph $\mathcal{G}_b$ has V nodes, we obtain a matrix σ^0 of $U \times V$ values.

Additionally, we build a $\binom{u+1}{2} \times \binom{v+1}{2}$ matrix Φ of edge similarities, where the value in position $\Phi(i+j,\ k+l)$ with $i, j, k, l \in \mathbb{N}_0$ corresponds to the edge similarity of $edge(u_i,\ u_k)$ and $edge(v_j,\ v_l)$.

The edge similarity is set to 0 if any of the node pairs $(u_i,\ u_k) \in \mathcal{G}_a$, $(v_j,\ v_l) \in \mathcal{G}_b$ have no connecting edge (including the case of both being the same node), or if the two edges have a different *alignment direction*. Otherwise, the edge similarity is computed as the Euclidean distance between the vectors composed of the features *(alignment magnitude, distance)*, normalized by the maximum value to have a similarity score in $[0, 1]$.

The similarity of the nodes in σ^0 is then iteratively "flooded" by multiplying the similarity of each node pair with the similarity of the neighboring node pairs, weighted by the edge similarity in Φ. In formal terms, the similarity of the i-th node of $\mathcal{G}_a$ and the j-th node of $\mathcal{G}_b$ is iteratively updated using the formula

$$\sigma^k(i,j) = \sigma^0(i,j) + \sum_{m=0\ldots V,\ n=0\ldots U} \sigma^{k-1}(m,n) \cdot \Phi(i+m, j+n)$$

As we look for a 1:1 node match, we ensure that for every neighboring node pair $(u_i,\ u_j) \in \mathcal{G}_a$ only the node pair $(v_j,\ v_l) \in \mathcal{G}_b$ with the maximum edge similarity is used. To avoid imbalance in similarities for node pairs (u,v) where any of u or v has a high number of neighbors, we normalize the value of Φ dividing $\Phi(u+v, u_i+v_j)$ by 2^{n-m}, where $n,\ m$ are the number of neighbors of u and v, respectively. Finally, at each iteration, we normalize the values of σ^i. The iterative computation is stopped either when the matrix distance $||\sigma^{i+1}, \sigma^i||_2$ falls below a given threshold, or when a maximum number of iterations is reached. During our experimentation, we empirically observed that in most cases the matrix difference falls quickly (in a handful of iterations) to values in the range $[0.01, 0.1]$ and then stabilizes, reaching values under 0.01 with a much slower convergence speed (in thousands of iterations). We, therefore, recommend setting a threshold of 0.1 and a maximum number of iterations to 10, which we deem sufficient considering the satisfactory results obtained on the template inference task reported in Section 5.4.

At the end of the similarity flooding stage, we can consider the matrix σ as the weight matrix of a fully connected bipartite graph $\mathcal{B}$, with the two partitions composed of the nodes of $\mathcal{G}_a$ and $\mathcal{G}_b$, respectively. To compute the final similarity score of $(\mathcal{G}_a, \mathcal{G}_b)$, we find a maximum weighted matching on $\mathcal{B}$ and average the corresponding weights found, including zero values in the computation for every $|\ |\mathcal{G}_0|$ - $|\mathcal{G}_1|\ |$ node left unmatched. In formal terms, given the weights $w(u,v)$ for nodes $u \in \mathcal{G}_a, v \in \mathcal{G}_b$, the similarity between $\mathcal{G}_a$ and $\mathcal{G}_b$ is computed as:

$$sim(\mathcal{G}_a, \mathcal{G}_b) = \frac{\sum_{u \in \mathcal{G}_a, v \in \mathcal{G}_b} w(u,v)}{max(|\mathcal{G}_a|, |\mathcal{G}_b|)}$$

As this graph similarity is asymmetrical, because of the matrix normalization included in the calculations, for every pair of files f_a, f_b we compute the final file layout similarity $sim(f_a, f_b)$ averaging between $sim(\mathcal{G}_a, \mathcal{G}_b)$ and $sim(\mathcal{G}_b, \mathcal{G}_a)$.

4.4 Template inference

The ultimate goal of Mondrian is to find spreadsheet layout templates. As we approximate pairwise layout equivalence with our graph-based similarity measure, we consider two files layout to be instances of the same template if their pairwise similarity is above a given threshold τ_f (subject to evaluation in Section 5.4).

To extend template inference beyond pairs of files, we use an "inductive" approach: given a set of files each with its detected regions, we examine the set iteratively. The first file f_0 is considered an instance of a template t_0, and its regions are added to a global index of regions $\mathcal{R}$, along with the information that these regions are found in the layout of f_0.

When a new file f_k is examined, first its regions are compared with all the regions in $\mathcal{R}$. If a region r_{f_k} is similar to a region r_t in $\mathcal{R}$ more than a threshold τ_r, we add the file layouts that contain r_t to the list of possible similar layout candidates for file f_k. During our experimentation, we discovered a region threshold $\tau_r = 0.75$ to be sufficient to obtain valid similar layout candidates.

If the layout of file f_k has a similarity greater than τ_f to the layout of a candidate file f_l, we group f_k, f_l, and, recursively, all files grouped with both f_k and f_l. In this way, we assume templates are transitively closed, an assumption that holds true given the results of Section 5.4. Nonetheless, the results for a file set are independent from the order the spreadsheets are processed: at the last iteration, all regions will have been compared against each other, and so will all layouts containing matching regions. If at any given point a file is found matching two distinct templates, these are merged.

Regardless of the outcome of template matching, once a file f_k has been compared with candidate files (if any), the region index R is updated with the regions of f_k that were not found to be similar to any other previous regions.

We chose this iterative approach for different reasons: first, it suits a continuous development scenario, where the region index

and template layouts are persistently stored and can be reused in later stages as new files are pre-processed. Second, it is significantly less computationally expensive to precompute region similarities and prune the template search space rather than perform graph similarity for each pair of files, which would anyway include computing the pairwise region similarity for all pairs of regions found across all files.

5 EVALUATION

Multiregion spreadsheets pose interesting data engineering challenges. In Section 3.3 we described three related research problems: region detection, region matching, and template inference. We conducted a set of experiments to evaluate whether is possible to address these problems using an automated approach that is general with respect to the spreadsheet format, i.e., it does not rely on proprietary file formats and vendor-specific features, and with respect to domain knowledge, i.e., it does not make any assumption on the nature of the files. While the performances on the region detection and template inference task are explicitly measured using a reference gold standard, we implicitly evaluate the region matching task with the results of template inference, as our approach makes use of region matching to select candidate layouts for a template and calculate their similarity. We compare Mondrian to a system that uses connected components to discover tables [7], an approach for genetic algorithm-based table recognition [16], and a CNN-based machine learning model [10].

5.1 Evaluation datasets and their properties

To evaluate our approach, we use two datasets of real-world spreadsheets. The first, Deco [14], is a publicly available annotated file sample of enterprise spreadsheets extracted from the ENRON corpus [12]. It is composed of 1,165 MS Excel files used in an energy company and found in email attachments from 2000 to 2001, annotated by Koci et al. [14]. Of those, roughly 27% are classified by the authors as not containing a table (e.g., containing only charts). For the remaining 854 files, in the case of multiple worksheets per file, the authors annotated only one worksheet with regions. We use these regions as candidates for our region detection task. In addition, we manually annotated the dataset at the file level to identify files with the same layout, for the template inference task[5].

The second dataset is sampled from FUSE, a large-scale corpus of spreadsheets crawled from various internet sources [20]. For our evaluation, we annotated the region layout and the templates of all relevant 886 worksheets from 780 unique, randomly sampled spreadsheet files. In the remainder of this section, we call this annotated subset Fuste (FUSE Sample for Template Extraction). The region-level annotations of Fuste have been obtained with the tools proposed in the original Deco paper [14], to stay consistent with those from this dataset. Table 2 reports the main characteristics of the two datasets with respect to their file layout. The first consideration is the wide presence, in both sources, of multiregion files: roughly 72% and 45% of files from Deco and Fuste, respectively, have more than one region.

Fuste has overall a greater number of single region files and on average much fewer regions per file than Deco (2.09 and 4.43, respectively), with Deco having more complex files with a huge number of regions – the maximum being 321. For the rest of the experiments, we regard as outliers and therefore exclude those files with more regions than the 99.9% of the remaining files in the same dataset. These files, two for Deco and one for Fuste, were characterized by an unusually large number of regions sparsely distributed across the spreadsheet.

Dataset	Deco	Fuste
Files	854	886
Files with one/multiple regions	233/621	495/391
Overall layout templates	750	136
Templates with one/more than one files	679/71	105/31

Table 2: Synthetic overview of the evaluation datasets.

In the similarity of layouts across files, the two datasets also show opposite natures. Deco has a low level of layout recurrence, with 750 different layout templates for 854 files, 679 of which are "singletons", i.e., covering only one file. Fuste, on the other hand, contains 136 templates for 886 files, with one encompassing as many as 381 different files and only 105 singleton templates. Mondrian handles both extremes well.

Various considerations arise from these fundamental differences. First, both manually annotated datasets represent a relatively small sample of the entire collection of files from the original corpora: ENRON, the source of Deco, is composed of 15,770 unique spreadsheet files with 79,983 sheets [12]; FUSE, the source of Fuste, has 249,376 unique spreadsheets. We are confident that the templates we discover typically cover many more spreadsheets than in our sample. Additionally, the origin and thus usage of the two datasets are different: Deco is a set of enterprise spreadsheets, in which files have possibly a "single-use" scope, be it for reporting or analysis purposes within the company; Fuste is a set of documents crawled from various public internet sources, most likely designed for sharing, with a high homogeneity of files originating from the same source. Finally, as surfaced during our annotation of Deco templates, this dataset could have shown a higher percentage of file similarity with a different choice of the worksheets: the choice of annotating only one worksheet per file excluded various worksheets from the original files that showed the same layout.

5.2 Related approaches for comparison

The experiments conducted to evaluate the performance of our region detection approach include, for comparison, the results obtained on the same task using the connected component detection algorithm outlined in the work of Coletta et al. [7], the genetic-based table recognition approach proposed by Koci et al. [16], and the CNN-based TableSense [10]. Furthermore, simply selecting Coletta et al.'s connected component approach can be considered a baseline for our approach: it is the first step from which we build upon with element partitioning and clustering. Therefore, comparing also against it, we can directly observe the influence and possible benefits of applying element partitioning and clustering.

The genetic-based approach is a more sophisticated process, involving two steps that rely on supervised machine learning methods. In the first step, a random forest classifier is trained on cell

[5] Annotated files are available at https://github.com/vitaglianog/mondrian_vldb.

features to label each spreadsheet cell according to its role (e.g., data, header, aggregate) [15]. Afterward, neighboring cells with the same label are grouped and a graph is formed, with cell groups as vertices and their spatial relationship as edges [13]. Different tables are recognized as sets of vertices obtained by partitioning the graph [16] using a supervised genetic-based algorithm.

This overall approach relies on rich features extracted from Excel files and aims at solving the more complex task of table recognition. Recall that the region detection task we solve is slightly different in goal and assumptions: we are interested in detecting region boundaries in general multiregion spreadsheets, without assuming special formatting features nor tabular structures.

The comparison was conducted with the help of the original authors, reusing the source code for the feature extraction, cell classification as well as for the genetic approach[6]. For a fair comparison, we experimented with two versions of the genetic-based approach: one using the full set of Excel-specific features available, and one restricting the input information to only cell content and position, excluding style features, thus simulating a *.csv* file input. The model, following the setup described by the authors in [16], is trained and tested on each dataset using 10-fold cross-validation.

TableSense, proposed by Dong et al. [10], is based on Mask R-CNN [11], a convolutional neural network developed for instance segmentation in images. TableSense extends this architecture for the task of table detection in spreadsheets with two specialized modules: a feature extraction stage to map spreadsheets into feature maps that are served as input to the network, and a Precise Bounding Box Regression layer to refine the coordinates of Mask R-CNN detected regions' bounding boxes. The intuition of TableSense, like Mondrian, is to map the region detection task on the visual domain: using a convolutional architecture, it leverages the 2D distribution of cells on a spreadsheet to identify "Regions of Interest", candidate areas of the input file, which are then classified as tables and whose boundaries are refined by the PBR module. The authors report experimental results of TableSense training the model on the WebSheet10K dataset and testing it on the WebSheet400 dataset. As neither the datasets nor trained models or source code are publicly available, to compare it with Mondrian in a similar setup we obtained the results training the model on one dataset and testing on the other, i.e., the results for Deco are obtained training TableSense on Fuste and vice-versa.

Due to the non-deterministic nature of the approaches that involve machine learning approaches (Genetic-based and TableSense), we repeated the experiments involving the full pipeline three times, and report average scores, with confidence intervals obtained from the standard deviation of the experiment results.

Mondrian is the first approach to address the template inference task. Therefore, we evaluate its performance against the baseline provided by using the connected components baseline as well as the correct regions from the gold standard to evaluate our layout similarity measure with perfect region boundaries. We use two setups regarding the choice of the clustering radius for region detection: one using an optimal, "dynamic" choice of the clustering radius for each file, and one with a "static" radius used across all dataset files. In the dynamic radius setting, we ran our clustering method on each file, varying the size of the radius between [0.1,2] in steps of 0.1, between [2,10] in steps of 1; and between [10,100] in steps 10. Additionally, we experimented with different configurations of the distance features' weights: we kept $\alpha = 1$ as a fixed reference value and varied $\beta, \gamma \in \{0, 0.5, 1, 5, 10\}$. The hyperparameter configuration that showed the best results was $\alpha = 1, \beta = 0.5, \gamma = 1$ for Deco, and $\alpha = 1, \beta = 1, \gamma = 1$ for Fuste. We use these values for experimenting in the "static" radius setting, in which we tried to find the single radius that showed the best performances across all files. The search space for the radii was the same as the one used in the dynamic setting. We report the result obtained using the radius with the best performance for each dataset, namely 1.5 for Deco and 1.4 for Fuste.

[6]https://github.com/ddenron/gen_table_rec, last accessed Feb 25, 2020

5.3 Region detection accuracy

To evaluate the level of accuracy in region detection, we use the Intersection-over-Union score (IoU) and the Error-of-Boundary score (EoB), defined in [10]. The first value is the graphical equivalent of the Jaccard index for sets. It is obtained, for a given predicted region-target region pair, by calculating the area corresponding to the intersection of the two bounding rectangles and normalizing it with the sum of the areas of the two rectangles, minus the shared cells. To be more faithful to the actual density of the regions, we include in the area calculations only the contribution of non-empty cells. The EoB is calculated as the maximum absolute distance between the pairs of the top, bottom, left, and right coordinates of the predicted and target rectangles. In general, each target region can be split over different prediction clusters, and vice-versa a cluster can span multiple regions. Therefore, if there are M predicted regions and N target regions, for all N target regions there will be M IoU and EoB scores calculated for each predicted region. To have one value for every target region, we select the maximum IoU score and the minimum EoB score. An IoU score of 1 corresponds to perfectly detected regions and a score of 0 to missed regions, whereas a perfectly detected region has an EoB of 0, with no upper limit for incorrect detections. EoB is undefined in the case of no detected region: whenever such a case arises, we set the EoB as the maximum of the height and width of the file, simulating a completely out of boundary detection.

To avoid setting a threshold over which to consider detections as "correct", Figure 6 shows the performance of the different approaches over varying thresholds: the y-axis represents the percentage of tables or regions correctly detected in the two datasets, assuming as "correct" an IoU/EoB score greater/smaller than the given reference on the x-axis. We report the performance for tabular regions only ("table detection"), and the performance across all types of regions ("region detection"), which include tables but also notes, spreadsheet titles, etc.

The best results for all regions are obtained, for both datasets, with our clustering approach assuming a dynamic, optimal choice of the radius for each file. It is interesting to note the difference in the behavior of Mondrian on the two different datasets. Deco, which contains more multiregion files and on average more regions per file, proves to be the harder of the two with approximately 45% of regions perfectly detected (100% IoU). On FUSE, instead, with fewer complex multiregion files, around 75% of the regions

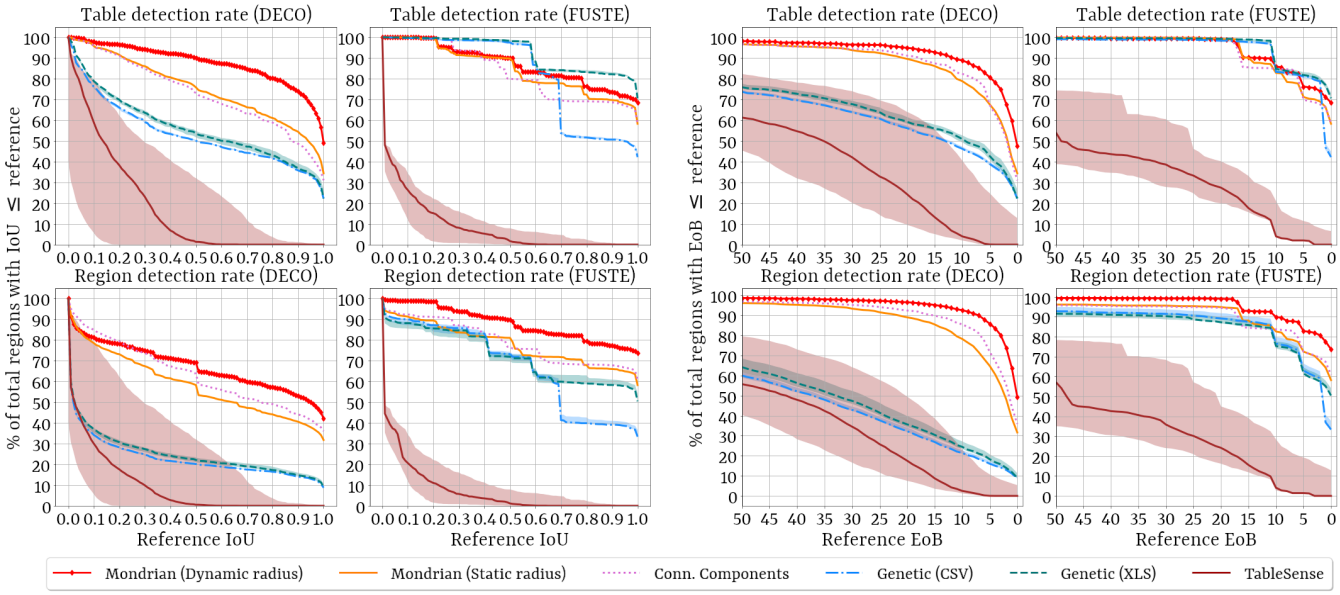


Figure 6: Comparative table and region detection performance.

are correctly detected. The usage of a static radius yields lower performance: in the case of tables the accuracy is comparable to detecting connected components, while on other region types it yields slightly worse results. In our experiments, a smaller radius (≤ 1) made the clustering degenerate into connected component detection, grouping only adjacent partitioned elements. A larger radius, such as the one selected for our static approach (namely 1.5), improves table detection, since a high number of tables is composed of separated connected components, but also brings together different non-tabular regions, which are usually independent. Because of this, the static radius variant of our clustering approach shows slightly worse performance in detecting general regions than tables.

It is not surprising that the genetic-based approach shows better results for tables than for generic regions, as it was specifically designed for table recognition. To analyze its behavior, we individually checked the correctness of its two steps. For Deco, the cell classification using csv-compatible features showed a 93.22% accuracy, which increased to 94.60% using Excel-specific features. For Fuste, both versions showed above 99% accuracy. We note, however, how the overall result is highly skewed on data-type cells, which by far outnumber other types of cells in the dataset. Its graph-partitioning and table detection phases, when run on perfect cell labels, had similarly good performance with 87.22% of the regions detected above a 95% IoU threshold. However, when the two steps were combined, the graph partitioning step proved to be sensitive to even small errors in the cell classification, with the results visible for the Deco dataset in Figure 6.

On Fuste, where the classification errors were minimal, the genetic-based approach showed much better results. We explain this phenomenon by considering the reliance of the genetic-based search on correctly labeled region boundaries. Some incorrectly classified cells cause fragmentation of what should be one coherent graph vertex into different vertices, some of them necessarily erroneous. Moreover, it appears that non-data cells, such as header or aggregation cells, are crucial for recognizing tabular structure. Such classification errors propagate into unreliable weight learning for the quality measures of the fitness function and finally cascade into poor table boundaries. It is worth noting how for Fuste the contribution of Excel-specific features is much more significant than for Deco: the gap between the two versions of the genetic approach is much wider.

The results of TableSense show low performance with a high variance. This behavior can be explained noting the considerable amount of regions that are completely missed: on average, 48.81% for Deco and 32.92% for Fuste. Contrarily to Mondrian, which by design does not ignore any non-empty input cell, the CNN architecture of TableSense may completely ignore entire areas of the input if they are not considered "Regions of Interest", or classified as containing an object. This behavior is inherited from the original domain of Mask R-CNN, designed for instance segmentation in images, which may or may not contain relevant objects. Overall, the poor accuracy of TableSense is most likely due to the high complexity of the model, which is composed of more than 85 million trainable parameters, and the limited number of training files available for our use-case.

Finally, comparing the plots across the two datasets, we note an interesting difference: the Deco plots are much smoother than those of Fuste, which show abrupt drops in the percentage of tables and regions recognized above a certain threshold. This phenomenon reflects the different dataset natures, as analyzed in Section 5.1. Considering that Deco has roughly twice the quantity of regions compared to Fuste, it is natural to expect a more continuous plot. What is more, Fuste contains a greater number of files that share the same templates: on average 5.33 files sharing the same layout compared to the 1.13 in Deco. In particular, one can observe how the percentage of tables (and regions) detected correctly in Fuste

drops from 80% (60%) to 50% (40%) for the Genetic-CSV approach as soon as the threshold for the IoU is increased from 69% to 70%. Looking for the causes of this behavior, we found that 323 different regions, coming from just as many files with the same layout, were detected in the same way. The absence of the same drop from the Genetic-XLS approach suggests that including style features helped in recognizing these regions correctly.

In general, we conclude from the results of this experiment that automatic region boundary detection is a difficult problem, subject to the complexity of the file layouts involved. For DECO, the hardest dataset, no approach showed convincing performance, with our clustering approach being able to perfectly detect around half of the region boundaries with a dynamic choice of the radius.

We observed the importance of recognizing recurring file layouts: failure to do so can cause a single mistake to span a huge number of files. As the next experiments in Section 5.4 show, detecting partially erroneous region boundaries does not prevent successful layout template inference, considering that the same incorrect boundaries would be repeated across all files with the same layout.

5.4 Template inference accuracy

In evaluating the template inference task, we rely on three external entropy-based measures for clustering: *homogeneity*, *completeness*, and *v-measure* [19]. The value range of all three scores is [0,1], with 1 being a perfect result. Using the gold standard, homogeneity quantifies how many data points in each predicted cluster belong to the same template. For our problem, in a perfectly homogeneous solution, all files that are grouped indeed share the same layout. Completeness, conversely, quantifies the percentage of elements from the same template that are grouped. V-measure is the harmonic mean of homogeneity and completeness.

As described in Section 4.4, we group files transitively based on their layout similarity being above a given threshold. We experimented with thresholds in the range [0.7,1] with a spacing of 0.01. To save computational time while repeating the experiments for different thresholds, we did not calculate the layout similarities of pairs for which we can guarantee a threshold lower than 0.7. This pruning was possible given the nature of our approach, where the similarity of two graphs is bound by the absolute difference in their number of nodes, normalized by the maximum number of nodes across the two graphs. Figure 7 shows the influence of the threshold value on the results of template recognition using the regions automatically detected by Mondrian in the static radius scenario, for the DECO and FUSTE datasets.

Considering how, especially for DECO, there is a significant number of singleton templates, i.e., templates that occur in only one file, we report the results of our template recognition approach for the full dataset as well as for the sub-set of files that constitute non-singleton templates (175 files for DECO and 781 for FUSTE, cf. Table 2). Increasing the threshold leads to a more selective behavior: for the maximum threshold of 1, homogeneity reaches a perfect value, as the resulting templates are always comprised of one file and therefore trivially homogeneous. This is compensated by the drop of completeness for high thresholds, especially noticeable in the FUSTE dataset. This effect is mitigated on the full DECO

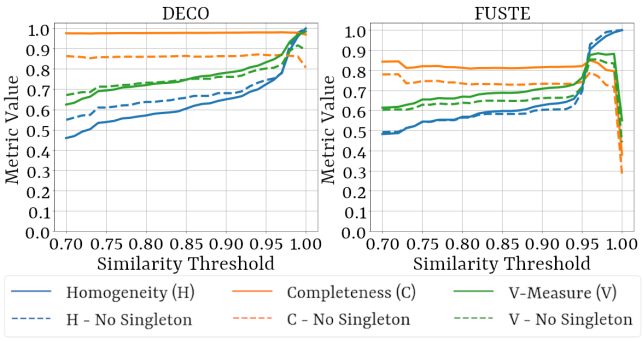


Figure 7: Performance of Mondrian on template inference.

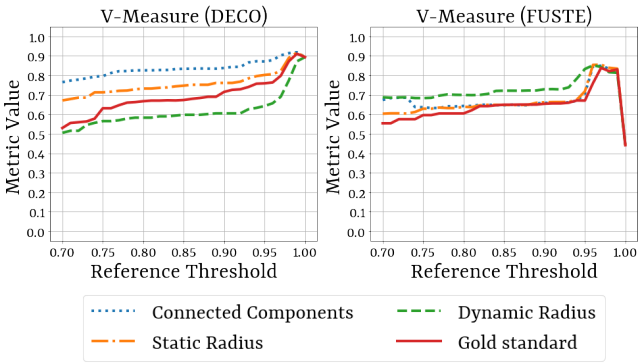


Figure 8: Effect of region detection on template inference.

dataset thanks to the high number of singleton templates. Overall, the performances of our template inference approach benefit from choosing high thresholds: across the two datasets, the best v-measures are obtained with thresholds between 0.95 and 1.00.

The performance of our template inference algorithm is also dependent on the results of the prior region detection phase. To analyze the sensitivity of the graph matching to region boundaries, we experimented with four different region detection strategies: (i) the manually annotated regions from the gold standard, (ii) those recognized by Mondrian using a dynamic radius and (iii) a static radius, and (iv) the ones obtained using the connected components baseline.

Figure 8 reports the v-measure for the different region detection strategies and baselines across datasets (excluding singleton templates). First, we highlight how, irrespective of the specific region detection strategy, the v-measures reach equal peaks for high thresholds. Surprisingly, for lower thresholds, using gold standard regions does not lead to better results. We attribute this effect to the increased complexity of the graphs produced with sub-optimal regions: as there may be potentially more automatically detected regions than needed, the resulting graphs contain more (noisy) information and therefore show a greater absolute difference in the case of different templates.

As further evidence, consider the execution times for the template recognition task shown in Table 3, obtained as the average run-times of our Python 3.8 scripts across three separate runs on

Region detection strategy	Template inference time (s) DECO	FUSTE
Gold Standard	198.07 ± 0.48	178.29 ± 0.46
Dynamic Radius	214.52 ± 3.32	16,661.11 ± 173.89
Static Radius	597.41 ± 3.62	4,181.31 ± 38.76
Connected Components	11,727.15 ± 385.04	6,218.21 ± 47.60

Table 3: Time performance of template inference.

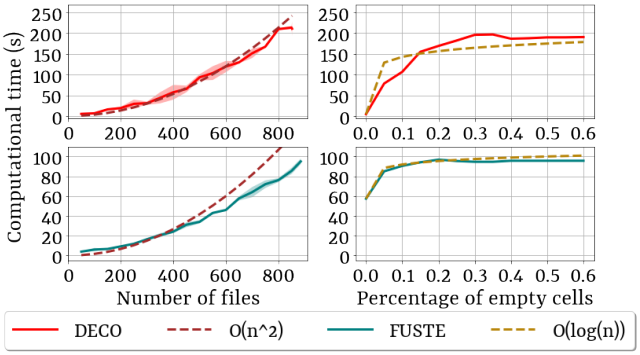


Figure 9: Effect of number of files and empty cells.

a machine equipped with an AMD Epyc 9 7702P Xeon 3,35 GHz CPU, 512 GB of RAM and an NVMe M.2 internal SSD. The region detection strategies that proved to be better for template inference at lower thresholds in Figure 8 are also the ones that needed significantly more time to execute: connected components for DECO and dynamic radius for FUSTE. The time needed to recognize file templates mostly corresponds to the time needed to compute graph similarity: larger graph structures, i.e., files with more detected regions, need greater time for the comparison.

Furthermore, Figure 9 shows the influence of the number of files and percentage of empty cells in files on the computational time of template detection using perfectly recognized regions. For the former, we experimented selecting random file sub-samples, with a step size of 50. For the latter, the sub-samples corresponded to all files with a number of empty cells up to a given percentage of the total file area, with a step size of 0.05%. In both cases, the file sets were sampled without repetition until full coverage of the dataset. The plot shows that the performances with respect to the number of input files follow a quadratic behavior, as Mondrian performs layout comparison for each pair of files in the input set. In turn, increasing the percentage of empty cells leads to a logarithmic behavior. Therefore, we conclude that the most impactful factor affecting the complexity of Mondrian is the number of input files, as well as the correctness of the region detection stage.

In conclusion, detecting regions and multiregion file templates automatically with Mondrian provides a convenient tradeoff between complexity and correctness of template inference.

6 RELATED WORK

While there is a substantial amount of research aimed at detecting and recognizing tables in single files of various formats, there is no research to recognize structural templates spanning different multiregion files. The two systems WebSmatch [7] and TableSense [10], like Mondrian, leverage the intuition of analyzing spreadsheets applying techniques from the computer vision domain. The first is an internet application that uses connected component detection and machine learning for table recognition to integrate semantically related tables within a dataset. The second uses a convolutional neural network architecture to address spreadsheet table detection based on a set of spreadsheet-specific cell features. As demonstrated by the experimental results in Section 5, supervised machine learning necessitates large quantities of training data, while the unsupervised nature of Mondrian makes it fit even for smaller datasets.

Supervised learning is also used in Pytheas, by Christodoulakis et al. [6]. This system employs a rule-based algorithm to discover tables in .csv files. Due to the nature of .csv files, tabular structures are expected to appear concatenated in one dimension, i.e., as subsequent lines (or columns). In contrast, Mondrian can detect region layouts with an extra degree of freedom, recognizing both horizontal and vertical alignments (Cf. Figure 4).

Encoding tabular layouts as graphs is at the core of the approach presented by Koci et al. [16], which tackles table recognition in spreadsheet files with a combination of supervised machine learning and genetic-based algorithms. While Mondrian uses complete graphs that encode all regions in a file with their pairwise distances, the genetic-based approach focuses on tabular regions and therefore misses information about the general file layout.

Existing spreadsheet systems that build on region boundary extraction can integrate well with Mondrian and make use of its layout template recognition. For example, to perform information extraction, the work of Chen et al. [4], given table boundaries, leverages active learning to detect interesting "spreadsheet properties", such as aggregation rows or hierarchies. Detecting spreadsheet templates can lead to a significant reduction of the number of files for which user feedback is required. Spreadsheet data management systems, like Senbazuru [2], can be empowered with Mondrian, e.g., using layout templates as database indices, or enriching query results with template information.

7 CONCLUSIONS

In this work, we formalized a framework for describing multiregion layout templates and identified three main challenges: detecting independent region boundaries in a single spreadsheet; matching similar regions on a structural level; finding a suitable similarity for file layouts. We presented the Mondrian approach, which combines automated region detection with an algorithm to identify similar file layouts. Experiments show that our approach works well in detecting the boundaries of different regions in a multiregion spreadsheet and in identifying layout templates.

Further research will focus on improving the accuracy of boundary detection and increasing the quality of the detected file layouts. We plan to include more information in the structure similarity computation, e.g., a finer-grained classification for the content of spreadsheet cells, to better identify structural patterns and correlations within templates.

ACKNOWLEDGMENTS

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

REFERENCES

[1] António Leslie Bajuelos, Ana Paula Tomás, and Fábio Marques. 2004. Partitioning orthogonal polygons by extension of all edges incident to reflex vertices: lower and upper bounds on the number of pieces. In *International Conference on Computational Science and Its Applications, (ICCSA)*. 127–136.

[2] Zhe Chen, Michael Cafarella, Jun Chen, Daniel Prevo, and Junfeng Zhuang. 2013. Senbazuru: A Prototype Spreadsheet Database Management System. *PVLDB* 6, 12 (2013), 1202–1205.

[3] Zhe Chen and Michael J. Cafarella. 2013. Automatic web spreadsheet data extraction. In *International Workshop on Semantic Search over the Web (SSW)*. 1:1–1:8.

[4] Zhe Chen, Sasha Dadiomov, Richard Wesley, Gang Xiao, Daniel Cory, Michael J. Cafarella, and Jock D. Mackinlay. 2017. Spreadsheet Property Detection With Rule-assisted Active Learning. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. 999–1008.

[5] Laura Chiticariu, Yunyao Li, Sriram Raghavan, and Frederick R. Reiss. 2010. Enterprise Information Extraction: Recent Developments and Open Challenges. In *Proceedings of the International Conference on Management of Data (SIGMOD)*. 1257–1258.

[6] Christina Christodoulakis, Eric Munson, Moshe Gabel, Angela Demke Brown, and Renée J. Miller. 2020. Pytheas: Pattern-based Table Discovery in CSV Files. *PVLDB* 13, 11 (2020), 2075–2089.

[7] Remi Coletta, Emmanuel Castanier, Patrick Valduriez, Christian Frisch, DuyHoa Ngo, and Zohra Bellahsene. 2012. Public data integration with WebSmatch. In *Proceedings of the International Workshop on Open Data (WOD)*. 5–12.

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 5–12.

[9] Marc Fisher and Gregg Rothermel. 2005. The EUSES spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. *ACM SIGSOFT Software Engineering Notes* 30, 4 (2005), 1–5.

[10] Dong Haoyu, Liu Shijie, Han Shi, Fu Zhouyu, and Zhang Dongmei. 2019. TableSense: Spreadsheet Table Detection with Convolutional Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 69–76.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. 2017. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision, ICCV*. 2980–2988.

[12] Felienne Hermans and Emerson Murphy-Hill. 2015. Enron's Spreadsheets and Related Emails: A Dataset and Analysis. In *Proceedings of the International Conference on Software Engineering (ICSE)*. 7–16.

[13] Elvis Koci, Maik Thiele, Wolfgang Lehner, and Oscar Romero. 2018. Table recognition in spreadsheets via a graph representation. In *Proceedings of the IAPR International Workshop on Document Analysis Systems (DAS)*. 139–144.

[14] Elvis Koci, Maik Thiele, Josephine Rehak, Oscar Romero, and Wolfgang Lehner. 2019. DECO: A Dataset of Annotated Spreadsheets for Layout and Table Recognition. In *Proceedings of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. 1280–1285.

[15] Elvis Koci, Maik Thiele, Oscar Romero, and Wolfgang Lehner. 2016. A Machine Learning Approach for Layout Inference in Spreadsheets. In *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*. 77–88.

[16] Elvis Koci, Maik Thiele, Oscar Romero, and Wolfgang Lehner. 2019. A Genetic-based Search for Adaptive Table Recognition in Spreadsheets. In *Proceedings of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. 1274–1279.

[17] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. 2002. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *Proceedings of the International Conference on Data Engineering (ICDE)*. 117–128.

[18] Johann Mitlöhner, Sebastian Neumaier, Jürgen Umbrich, and Axel Polleres. 2016. Characteristics of open data CSV files. In *Proceedings of the Image Analysis and Processing Conference (ICIAP)*. 72–79.

[19] Andrew Rosenberg and Julia Hirschberg. 2007. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 410–420.

[20] Barik Titus, Lubick Kevin, Smith Justin, Slankas John, and Murphy-Hill Emerson R. 2015. Fuse: A Reproducible, Extendable, Internet-Scale Corpus of Spreadsheets. In *IEEE/ACM Working Conference on Mining Software Repositories, MSR*. 486–489.