Elvis Koci, Maik Thiele, Oscar Romero, Wolfgang Lehner

Cell Classification for Layout Recognition in Spreadsheets

Erstveröffentlichung in / First published in:

IC3K: International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. Porto, 09. – 11.11.2016. SpringerLink, S. 78 – 100. ISBN 978-3-319-99701-8.

DOI: https://doi.org/10.1007/978-3-319-99701-8_4

Final edited form was published in "International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. Porto 2016", S. 78 - 100. ISBN: 978-3-319-99701-8
https://doi.org/10.1007/978-3-319-99701-8_4

Cell Classification for Layout Recognition in Spreadsheets

Elvis Koci[1,2], Maik Thiele[1], Oscar Romero[2], and Wolfgang Lehner[1]

[1] Database Technology Group, Department of Computer Science,
Technische Universität Dresden, Dresden, Germany
{elvis.koci,maik.thiele,wolfgang.lehner}@tu-dresden.de
[2] Departament d'Enginyeria de Serveis i Sistemes d'Informaciò,
Universitat Politecnica de Catalunya, Barcelona, Spain
{ekoci,oromero}@essi.upc.edu

Abstract. Spreadsheets compose a notably large and valuable dataset of documents within the enterprise settings and on the Web. Although spreadsheets are intuitive to use and equipped with powerful functionalities, extracting and reusing data from them remains a cumbersome and mostly manual task. Their greatest strength, the large degree of freedom they provide to the user, is at the same time also their greatest weakness, since data can be arbitrarily structured. Therefore, in this paper we propose a supervised learning approach for layout recognition in spreadsheets. We work on the cell level, aiming at predicting their correct layout role, out of five predefined alternatives. For this task we have considered a large number of features not covered before by related work. Moreover, we gather a considerably large dataset of annotated cells, from spreadsheets exhibiting variability in format and content. Our experiments, with five different classification algorithms, show that we can predict cell layout roles with high accuracy. Subsequently, in this paper we focus on revising the classification results, with the aim of repairing misclassifications. We propose a sophisticated approach, composed of three steps, which effectively corrects a reasonable number of inaccurate predictions.

Keywords: Speadsheet · Tabular · Table · Document · Layout Recognition · Analysis · Classification

1 Introduction

Spreadsheet applications have evolved to be a tool of great importance for transforming, analyzing, and representing data in visual way. In industry, a considerable amount of the enterprise knowledge is stored and managed in this format. Domain experts use spreadsheets for financial analysis, logistics and planning. Also, spreadsheets are a popular format on the Web. Of particular importance are those that can be found in Open Data platforms, where governments, impor-tant institutions, and non profit organizations are making their data available.

All this make spreadsheets a valuable source of information. However, they are optimized to be user-friendly rather than machine-friendly. The same data can be formatted in different ways depending on the information the user wants to convey. It is relatively easy for humans to interpret the presented information, but it is rather hard to do the same algorithmically. As a result, we are constrained to cumbersome approaches that limit the potential reuses of data maintained in these files. A typical problem that arises in most enterprises is that due to the lack of visibility the data stored in spreadsheets is not available for enterprise-wide data analysis or reuse.

Our goal is to overcome these limitations by developing a method that allows to discover tables in spreadsheets, infer their layout and other implicit information. We believe that this approach can provide the means to extract a richer and more structured representation of data from spreadsheets. This representation will act as the base for transforming the data into other formats, such as a relational table/s or a JSON documents.

In this paper we discuss an extended version of our work, which was first introduced at [1]. Here, we focus as well on layout inference via cell classification, describing all the important aspect of our approach. However, we also put emphasis on the post-classification process, where we attempt to correct incorrect predictions.

The paper is organized as follows: In Sect. 2, we define the classification problem for layout inference in spreadsheets. We present the dataset used as ground truth, in Sect. 3. We describe, in Sect. 4, our cell classification approach. Here, we list all defined cell features, explain how the most promising were selected, and provide the evaluation results from the classification experiments. A thorough report of our misclassification repairing approach can be found Sect. 5. Finally, we review related work on table identification and layout discovery in Sect. 6.

2 The Classification Problem

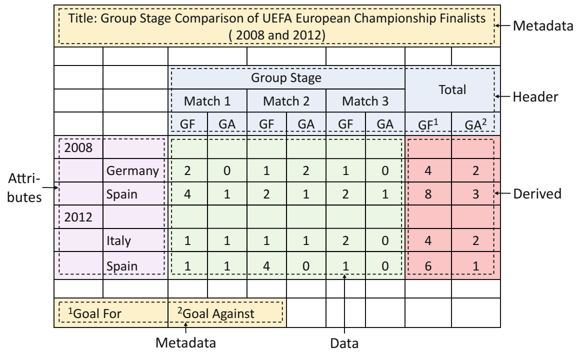
The objective of capturing the tabular data embedded in spreadsheets can be treated as a classification problem where the individual sections of a table have to be identified. In this section, initially, we define these sections or building blocks that form our classes. Subsequently, we specify the granularity on which the classification task will be performed.

2.1 Spreadsheet Layout Building Blocks

Considering that tables embedded in spreadsheets vary in shape and layout, it is rather challenging to directly recognize them as a whole. Thus, we opt to recognize their building blocks, instead.

We define five building blocks for spreadsheet tables: Headers, Attributes, Metadata, Data and Derived (see Fig. 1). A "Header" (H) cell represents the label of a column and can be flat or hierarchical (stacked). Hierarchical structures

Final edited form was published in "International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. Porto 2016", S. 78 - 100. ISBN: 978-3-319-99701-8
https://doi.org/10.1007/978-3-319-99701-8_4



Title: Group Stage Comparison of UEFA European Championship Finalists (2008 and 2012)									
		Group Stage						Total	
		Match 1		Match 2		Match 3			
		GF	GA	GF	GA	GF	GA	GF^1	GA^2
2008									
	Germany	2	0	1	2	1	0	4	2
	Spain	4	1	2	1	2	1	8	3
2012									
	Italy	1	1	1	1	2	0	4	2
	Spain	1	1	4	0	1	0	6	1
^{1}Goal For		^{2}Goal Against							

Fig. 1. The building blocks [1].

can be also found in the left-most or right-most columns of a table, which we call "Attributes" (A), a term first introduced at [2]. Attributes can be seen as instances from the same or different (relational) dimensions placed in one or multiple columns in a way that conveys the existence of a hierarchy. We label cells as "Metadata" (M) when they provide additional information regarding the worksheet as a whole or specific sections. Examples of Metadata are the table name, creation date, and the unit of the values for a column. The remaining cells form the actual payload of the table and are labeled as "Data". Additionally, we use the label "Derived" (B) to distinguish those cells that are aggregations of other Data cells' values. Derived cells can have a different structure from the core Data cells, therefore we need to treat them separately. Figure 1 provides examples of all the aforementioned building blocks.

2.2 Working at the Cell Granularity

One potential solution for the table identification and layout recognition tasks is to operate under some assumptions about the structure of spreadsheet tables. That means expecting spreadsheets to contain one or more tables with typical layouts that are well separated from each other. In such scenario we could define simple rules and heuristics to recognize the different parts. For example, the top row could be marked as Header when it contains mostly string values. Additionally, cells containing the string "Table:" are most probably Metadata. However, this approach can not scale to handle arbitrary spreadsheet tables. Since, the corpora we have considered include spreadsheets from various domains, we need to find a more accurate and more general solution.

For this reason, our approach focuses on the smallest structural unit of a spreadsheet, namely the cell. At this granularity we are able to identify arbitrary layout structures, which might be neglected otherwise. For instance, it is tricky to classify rows when multiple tables are stacked horizontally. The same applies for the cases when Metadata are intermingled with Header or Data. Nevertheless, we acknowledge that the probability of having misclassifications

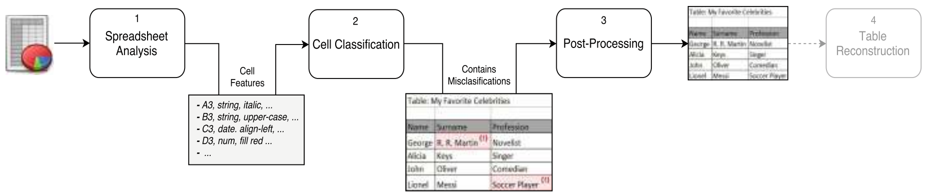


Fig. 2. The cell classification process [1].

increases when working with cells instead of composite structures such as rows or columns. Therefore, our aim is to come up with novel solutions that mitigate this drawback.

Figure 2 illustrates the three high-level tasks that compose our cell classification process. Initially, the application reads the spreadsheet file and extracts the features of each non-blank cell. Here we considered different aspects of the cell, summarized in Sects. 4.1 and 4.2. In the next step, cells are classified with high accuracy (see Sect. 4.3) based on their features. Finally, a post-processing step improves even further the classification results. Using rules and machine learning techniques we identify cells that are most probably misclassified, and attempt to infer their true label (i.e., layout role).

To complete the picture, Fig. 2 also includes the Table Reconstruction task, which forms a separate topic, and is therefore left as future work.

3 The Gold Standard

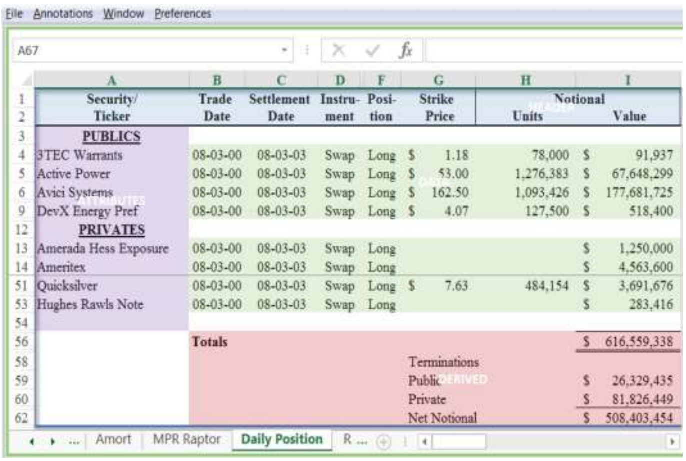
The supervised classification processes requires a ground truth dataset, which is used for both training and validation. In Sect. 3.1 we briefly describe the three spreadsheet corpora used to extract a representative set of spreadsheets. To create the training data we developed a spreadsheet labeling tool (see Sect. 3.2), which provides the means to annotate ranges of cells. Given that tool, we randomly selected and annotated files from the three corpora for which we provide statistics in Sect. 3.3.

3.1 Spreadsheet Corpora and Training Data

For our experiments we have considered spreadsheets from three different sources. EUSES [3] is one of the oldest and most frequently used corpora. It has 4,498 unique spreadsheets, which are gathered through Google search, using keywords such as “financial” and “inventory”. The ENRON corpus [4] contains over 15,000 spreadsheets, extracted from the Enron email archive. This corpus is of a particular interest, since it provides access to real-world enterprise spreadsheets. The third corpus is FUSE [5] that contains 249,376 unique spreadsheets, extracted from Common Crawl. Each spreadsheet in FUSE is accompanied by a JSON file that contains metadata and statistics. Unlike the other two corpora, FUSE can be reproduced and extended.

3.2 The Annotation Tool

Using the Eclipse SWT[1] library we developed an interactive desktop application that ensures good quality annotations. The original Excel spreadsheet is embedded into a Java window and protected from user alteration. To create an annotation, the user selects a range of cells (region) and then chooses the appropriate predefined label. A rectangle, which is filled with the color associated to the label, covers the annotated region. The application evaluates the annotations and rejects the inappropriate ones. For example, the user cannot annotate ranges that are empty or overlapping with existing annotations. The data from all the created annotations are stored in a new sheet, named "Range_Annotation_Data". The sheet is protected and hidden once the file is closed. Figure 3 provides an example of an annotated sheet.



	A	B	C	D	F	G	H	I
1	Security/	Trade	Settlement	Instru-	Posi-	Strike	Notional	
2	Ticker	Date	Date	ment	tion	Price	Units	Value
3	PUBLICS							
4	3TEC Warrants	08-03-00	08-03-03	Swap	Long	$ 1.18	78,000	$ 91,937
5	Active Power	08-03-00	08-03-03	Swap	Long	$ 53.00	1,276,383	$ 67,648,299
6	Avici Systems	08-03-00	08-03-03	Swap	Long	$ 162.50	1,093,426	$ 177,681,725
9	DevX Energy Pref	08-03-00	08-03-03	Swap	Long	$ 4.07	127,500	$ 518,400
12	PRIVATES							
13	Amerada Hess Exposure	08-03-00	08-03-03	Swap	Long			$ 1,250,000
14	Ameritex	08-03-00	08-03-03	Swap	Long			$ 4,563,600
51	Quicksilver	08-03-00	08-03-03	Swap	Long	$ 7.63	484,154	$ 3,691,676
53	Hughes Rawls Note	08-03-00	08-03-03	Swap	Long			$ 283,416
54								
56		Totals						$ 616,559,338
58						Terminations		
59						Public		$ 26,329,435
60						Private		$ 81,826,449
62						Net Notional		$ 508,403,454

Fig. 3. Annotated sheet [1]. (Color figure online)

In addition to the building block described in Sect. 2, we have introduced the possibility to annotate a region (area) that represents a whole table. A rectangle with thick blue borders marks its boundaries.

We need the "Table" annotation for two main reasons: Firstly, we can govern the labeling process, to assure valid annotations. For example, Data can only exist inside a Table. However, Metadata can be left outside when they provide information relevant to multiple Tables. Secondly, these annotations will help us evaluate the Table Reconstruction task, in the future.

[1] https://www.eclipse.org/swt/.

3.3 Annotation Statistics

The graphs below provide an overview of the collected annotations and the contribution of each corpus. We considered each corpus individually and assigned a unique number to their files. Using a random number generator we extracted subsets of files. From these, we annotated a total of 465 worksheets (216 files) and 898 tables (Fig. 4).

In Fig. 5 we examine the annotated cells. The total number of cells for each label (class) is placed at the top of the column bar. There was a small amount of cells that did not match any of the defined labels. These are usually random notes that do not have a clear role, and do not provide additional context (information) about the table. We decided to omit such cells.

As can be seen in Fig. 5, the number of Data cells is by orders of magnitude larger than the other label numbers. To adjust the class distribution we undersampled the Data class. We consider from each table in our dataset the first, the last row, and three random rows in between. By applying this technique the Data class was reduced to 32, 875 instead of 808, 179 cells. Considering also the other four classes the final gold standard consists of 52, 948 cells in total.

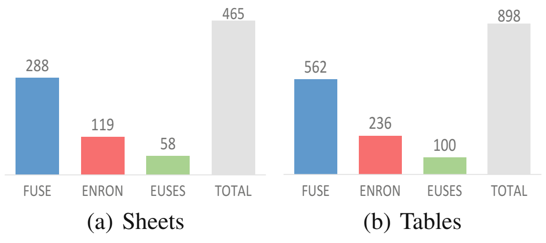


Fig. 4. Annotation statistics [1].

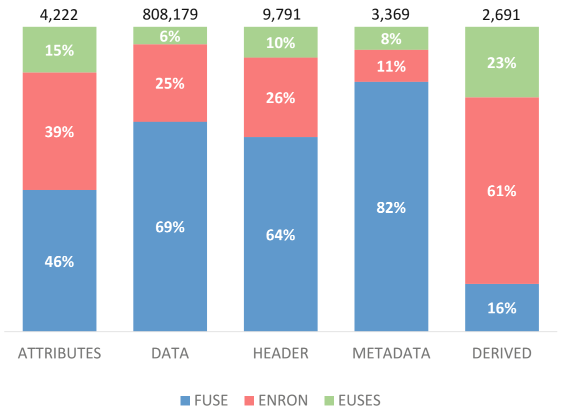


Fig. 5. Annotated cells [1].

4 Cell Classification

4.1 Feature Specification

We have grouped the defined features into 5 categories: content, cell style, font style, reference, and spatial. The *content* features describe the cell value, but not its format. The *cell style* features capture the formatting applied to the cell, excluding the font formatting. The later is recorded by *font style* features. *Reference* features explore the Excel formulas and their references in the same or other worksheets. The last group, *spatial* features, describe the "neighborhood" of the cell (i.e., the adjacent cells). Here, we do not define the individual features, instead more details can be found at [1].

4.2 Feature Selection

We used Weka[2], a well known tool for machine learning tasks, for feature selection and classification. Initially, we binarized nominal features with more than two values, which gave us 219 features in total. We used the "RemoveUseless" option to remove the features that do not vary at all or vary too much. Additionally, we manually removed those features that are practically constant (i.e., at least 99.9% of cases the value is the same). Furthermore, we decided to exclude from the final set features that check the style and content type of the neighbors. Although, these features are promising, they need further refinement. Thus, we plan perform thorough experiments with them in the future.

The remaining 88 features were evaluated using the *InfoGainAttribute*, *GainRatioAttribute*, *ChiSquaredAttribute*, *ConsitencySubset*, and *CfsSubset* feature selection methods. For each one of them we performed 10 folds (runs). A bidirectional Best First search was used for *ConsitencySubset* and *CfsSubset*, while the other methods can only be coupled with Ranker search.

From the results we considered features that score high in all these selection methods. Although, we were predominantly influenced by *ConsistencySubset* results, since, when tested, they provide the highest classification accuracy. We also included in the final set features that are strong indicators despite the fact that they describe small number of instances. "Words_Like_Table" is an example of such features, where 48 out of total 49 positive (true) cases are instances of the Metadata class.

Tables 1 and 2 list the selected features, 43 in total. Those suffixed with ? represent boolean features. While, those suffixed with # represent numeric features.

In general spreadsheets exhibit different characteristics depending on the domain they come from. Therefore, we expect that some of this features might not works as well for other spreadsheet datasets. For example, reference features are more important for industrial rather than for Web spreadsheets, since the former are characterized by heavier use of formulas. We note that an independent feature selection might be required for other spreadsheet datasets, in order to achieve near optimum accuracy.

[2] http://www.cs.waikato.ac.nz/ml/weka/.

Final edited form was published in "International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. Porto 2016", S. 78 - 100. ISBN: 978-3-319-99701-8
https://doi.org/10.1007/978-3-319-99701-8_4

Table 1. Selected content and style features [1].

Content	Cell style
LENGTH#	INDENTATIONS#
NUMBER_OF_TOKENS#	H_ALIGNMENT_DEFAULT?
LEADING_SPACES#	H_ALIGNMENT_CENTER?
IS_NUMERIC?	V_ALIGNMENT_BOTTOM?
IS_FORMULA?	FILL_PATTERN_DEFAULT?
STARTS_WITH_NUMBER?	IS_WRAPTEXT?
STARTS_WITH_SPECIAL?	NUMBER_OF_CELLS#
IS_CAPITALIZED?	NONE_TOP_BORDER?
IS_UPPER_CASE?	THIN_TOP_BORDER?
IS_ALPHABETIC?	NONE_BOTTOM_BORDER?
CONTAINS_SPECIAL_CHARS?	NONE_LEFT_BORDER?
CONTAINS_PUNCTUATIONS?	NONE_RIGHT_BORDER?
CONTAINS_COLON?	MEDIUM_RIGHT_BORDER?
WORDS_LIKE_TOTAL?	HAS_NO_DEFINED_BORDERS?
WORDS_LIKE_TABLE?	
IN_YEAR_RANGE?	

Table 2. Selected font, reference and spatial features [1].

Font	Reference	Spatial
FONT_SIZE#	IS_AGGREGATION_FORMULA?	ROW_NUMBER#
FONT_COLOR_DEFAULT?	REFERENCE_VALUE_NUMERIC?	COLUMN_NUMBER#
IS_BOLD?		HAS_0_NEIGHBORS?
NONE_UNDERLINE?		HAS_1_NEIGHBOR?
		HAS_2_NEIGHBORS?
		HAS_3_NEIGHBORS?
		HAS_4_NEIGHBORS?

4.3 Cell Classifiers

In our evaluation, we considered various classification algorithms, most of which have been successfully applied to similar tasks in the literature. Specifically, we considered CART [6] (*SimpleCART* in Weka), C4.5 [7] (*J48* in Weka), Random Forest [8] and support vector machines [9] (*SMO* in Weka). The latter uses the sequential minimal optimization algorithm developed by [10] to train the classifier. Here, with SMO we considered both polynomial kernel and RBF kernel.

We evaluated the classification performance using 10-fold cross validation. The Random Forest (RF) gave the highest overall accuracy of 98.2%. Also, RF outperformed the other algorithms, in all the defined classes. To provide a more

Final edited form was published in "International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. Porto 2016", S. 78 - 100. ISBN: 978-3-319-99701-8
https://doi.org/10.1007/978-3-319-99701-8_4

concrete picture on the classification accuracy, we also tested Random Forest against the full dataset of annotated cells (828, 252). There was a slight decrease in performance specifically for Metadata and Derived, but the general accuracy did not suffer. More detailed results can be found at [1].

5 Post-processing

In this section, we discuss techniques for handling the misclassifications that occur during the cell classification process. It is in our benefit to revise these incorrect predictions as it will make the subsequent tasks, such as table identification and schema extraction in spreadsheets, much more easier and accurate. In the following paragraphs, we discuss two approaches. We start with what can be considered a naïve approach, and afterwards we motivate and describe our more sophisticated solution.

5.1 Naïve Approach

Our initial empirical analysis of the classification results hinted that neighboring cells could be potentially used to recover some of the misclassifications. Intuitively, the label assigned to a cell should match, in most of the scenarios, at least that of the neighboring cells in the same row and/or column.

Here, we define the neighborhood as a 3-by-3 window around a cell, shown in Fig. 6. The red cell in the center, marked with "x", represents a misclassification, surrounded by 8 neighboring cells.

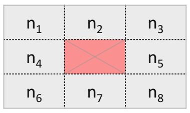


Fig. 6. A 3 × 3 cell neighborhood. (Color figure online)

Not necessarily all neighboring cells have a label. For example, empty and hidden cells do not get labeled, since we omit them from the classification process. Also, another special case are the misclassified cells at the boundaries (extremes) of the worksheet (i.e., the minimum and maximum allowed row/column in the spreadsheet application). Such cells have less than eight neighbors, since one of the neighboring columns and/or rows does not exist.

We would like to standardize the number of neighbors for any arbitrary cell to eight. We accomplish this by adding two artificial labels: "Empty" and "Imaginary". The latter shall be used for (non-existing) neighbors outside the boundaries of the worksheet, while the first for all the other cases of un-labeled neighboring cells.

Having seven labels in total, we implemented a script to find distinct arrangements of labels in the neighborhood of incorrectly classified cells. From $1,237$ misclassifications, resulting from the classifications in the full dataset ($828,252$ cells), there were 672 unique neighborhood label arrangements.

We identified the first 40 most repeated arrangements and manually inferred generic rules (i.e., not bound to specific labels) from them. These rules are divided into two sets: identification and relabeling. Intuitively, we use the first rule-set to identify incorrect classifications, and afterwards we relabel them using the second rule-set. Using this technique we managed to repair 152 misclassified, but lose 14 correctly classified cells. Here, we do not provide details about the individual rules, instead the complete list can be found at [1].

Though this method is able to recover a number of incorrect classifications, it has several limitations. Our subsequent experiments revealed that in almost half of the cases there are misclassified cells in the neighborhood itself, as illustrated in Fig. 7. Beside these observations, there are other good reasons to look further than the immediate neighborhood. For example, a Header cell that is relatively far from all Data cells in a worksheet is probably a misclassification. Another example comes from tables with missing values, which translates at Data cells having empty immediate neighbors. In such scenario, we need to look at more distant neighbors to determine if a Data cell is misclassified or not.

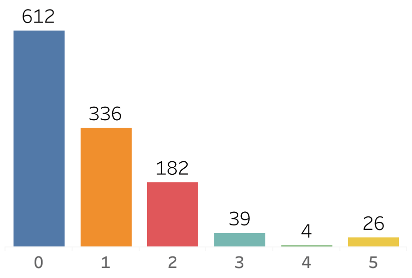


Fig. 7. Occurrences of misclassifications in the immediate neighborhood.

In the following section we propose a new approach for detecting and fixing misclassified cells. This approach is based on a good mixture of features, extracted from the immediate and the distant neighbors. The results from our evaluation are very encouraging and confirm the advantages of taking into account a broader neighborhood context.

5.2 Region Based Approach

In this section we present our region-based approach (RBA) for handling misclassifications. At the core of RBA, is the intuition that grouping adjacent cells of the same label should form regions of rectangular shape. That is because tables have well separated sections, such as the Data and Header sections, which tend to be rectangular instead of an arbitrary rectilinear polygonal shape. Here, we consider as adjacent cells only immediate neighbors on the left, right, top, and bottom (see Fig. 6). Additionally, we define rectangular region as a well formed matrix of cells of the same label. Intuitively, in such a matrix, all rows have the same number of cells. The same is true when we examine the columns of the matrix.

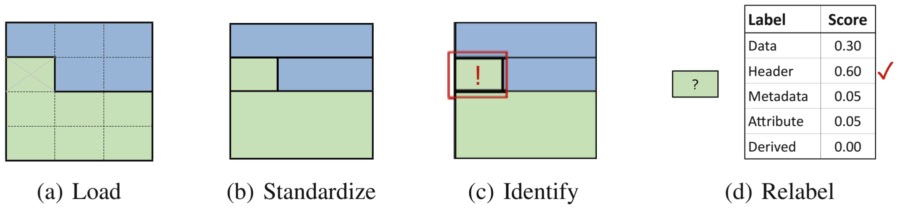


Fig. 8. Region based approach. (Color figure online)

We claim that non-rectangular cell regions point towards misclassifications. In other words, some cell/s break the regularity of the region. For us these cells are potential misclassifications. Hence, the aim of our approach becomes to isolate them, and subsequently determine the correctness of their label. Figure 8a illustrates how a misclassification, the cell marked with an x, can introduce irregularities.

Also, in Fig. 8 we introduce the tasks that compose RBA. Initially, we load the classification results. We then create strictly rectangular regions per each label. In the subsequent step, we identify incorrect classifications. Finally, we predict the true label for those regions flagged as misclassified.

Standardization and Confinement. This task starts by building what we call Row Label Intervals (RLI). We define these intervals as a sequence of cells of the same label in a row. Figure 9a displays the intervals from the example shown in Fig. 8a. The first, third, and fourth row contain one interval each, while the second row contains two intervals of different labels.

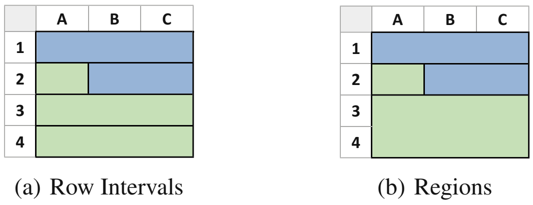


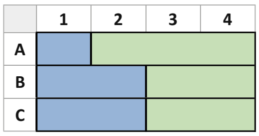
Fig. 9. Original worksheet. (Color figure online)

Final edited form was published in "International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. Porto 2016", S. 78 - 100. ISBN: 978-3-319-99701-8
https://doi.org/10.1007/978-3-319-99701-8_4

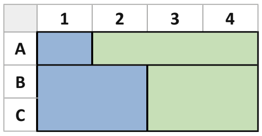
As we emphasized in the previous section, we are interested in strictly rectangular (cell) regions. We try to achieve this by merging RLIs of same label. This is only possible when the RLIs in adjacent rows have the same start and end. Otherwise we introduce shape irregularities. Based on this reasoning, for our running example, we merge the intervals in the third and fourth row, as illustrated in Fig. 9b. Using this technique we manage to isolate the single-cell region that prevented the rest of the green cells to form a well-shaped cluster.

We were not able to merge the blue row intervals from 1st and 2nd row, since they have a different start column. However, clearly there is the potential to build a larger blue region, that is B1:C2. This would have isolated the cell A1. The latter is desirable, since at this phase we aim at pinpointing irregularities.

One way to tackle this challenge is by creating regions column-wise, in addition to row-wise. We pivot (transpose) the worksheet, so that the columns of the original worksheet become the rows of the transformed worksheet. Afterwards, it is easy to construct row intervals, following the steps described previously. The results are shown in Fig. 10a. Once we attempt to merge the RLIs, we get the output shown in Fig. 10b. As intended we create a large blue region, and isolate the blue interval that does not fit well with the rest.



(a) Pivot Intervals



(b) Regions

Fig. 10. Pivoted worksheet. (Color figure online)

Our standardization procedure produces two alternative partitioning strategies for the labeled cells. For some worksheets the optimum partitions might come from one of the directions (i.e., row-wise or column-wise). However, for others we need both directions to ensure that for each misclassified cell there is at least a region that confines it from correct classifications. Thus, we have to keep both outputs, which has the drawback of augmenting the number of regions in the worksheet. One possible and required action for reducing this number is to filter out duplicate regions from the outputs. Figure 11 summarizes the overall standardization procedure, which includes the duplicates filtering step.

Confinement Assessment. To assess the validity of this procedure we decided to evaluate it on the classification results. We divide the resulting regions into three categories based on the ratio of misclassified cells they contain. We call "Correct" those regions that do not contain any misclassification. For those that only contain misclassifications we use the term "Misclassified". The remaining cases, regions that contain both correct and incorrect classifications, we call "Mixed". Figure 12a provides the number of regions per category.

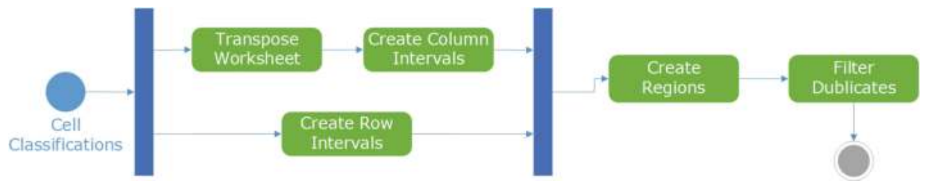


Fig. 11. Standardization procedure.

Mixed regions might raise concerns at first glance, since they occur a noticeable number of times. However, they are a natural by-product of the procedure. Consider again Fig. 10b, the first cell of the green region (interval) at the top row is misclassified, as pointed out in Fig. 8a. In this example, the Mixed region occurs when processing the transformed (pivoted) worksheet. However, in the general case both variants of the worksheet can produce such regions.

Additionally, we have performed a more detailed analysis of Mixed regions. The results are displayed in Fig. 12b. We note that for the majority of cases the number of correctly classified cells is greater than that of misclassified cells. Also, there are 69 cases where the numbers are equal, and an insignificant number of cases with more misclassified cells.

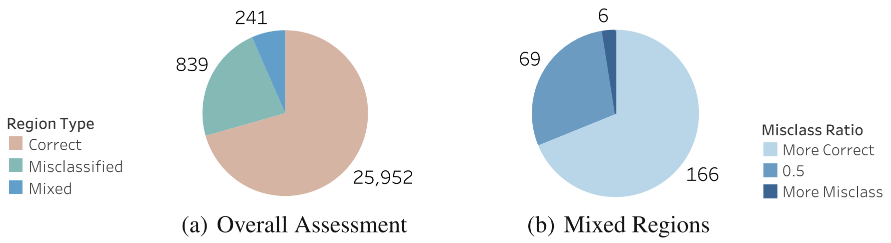


Fig. 12. Region analysis.

To simplify our subsequent operations, we decided to maintain only two categories of regions. Those that mostly contain incorrect classifications (>0.5) are marked as Misclassified, the rest as Correct. These brings the number of regions per category to 845 and 26, 187 respectively.

Filtering by Size. As mentioned before, one of the drawbacks of our standardization procedure is the considerable number of outputted regions. Ideally, we would like to keep only those that have the most potential to be Misclassified. Therefore, we analyzed the Misclassified regions, with the purpose of identifying some of their typical characteristics. Our analysis revealed that these regions exhibit small sizes (i.e., number of cells in the region), as shown in Fig. 13. Clearly, the larger is the size of a Misclassified region the least are the occurrences.

Final edited form was published in "International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. Porto 2016", S. 78 - 100. ISBN: 978-3-319-99701-8
https://doi.org/10.1007/978-3-319-99701-8_4

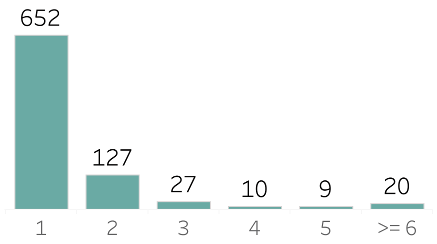


Fig. 13. Size occurrences in misclassified regions.

Based on the results in Fig. 13, we decided to consider only regions of size 1–3 from the procedure outputs, since they have significant number of occurrences. After, performing this filter, for our dataset, we get 12,724 regions. Out of these, 806 are Misclassified, and 11,918 are Correct regions. We utilize this reduced dataset for the tasks described in the following sections.

Detecting and Repairing Misclassifications. We have defined misclassification detection and repairing (relabeling) as supervised learning tasks. For this, we have created a set of features, which are formally described in the following paragraphs. Most of these features are used for both detection and repairing.

Region Features. Table 3 summarizes the features that are used to characterize each rectangular region. We use the same convention as in Sect. 4.2 to distinguish numerical features from boolean ones. We note that *predicted_label* does not fit in any of these categories, since it is a nominal feature. Here, we have additionally introduced the categories "Simple" and "Compound". As their formal definition will show, compound features are derived from multiple simple ones (some of them not explicitly listed in the table).

All features, introduced in this section, are based on the conception that a region can be solely represented with the rectangle that bounds its cells and the label of these cells. The worksheet itself can be seen as a Cartesian Coordinate system, where the point $(1, 1)$ is at the top left corner. The values of the x-axis increase column-wise, while for y-axis they increase row-wise.

Having such coordinate system, it is relatively easy to convert the regions into abstract rectangles. The coordinates for the top-left vertex of the rectangle are the column and row number of the top-left cell in the region. Its width and height can be calculated by counting respectively the number of cells in a column and in a row of the region. For example, the large green region in Fig. 9b will be represented with the rectangle having as top-left vertex the point $(1, 3)$, $width = 3$, and $height = 2$.

The *simple features* characterize various aspects of the rectangle (region). A rectangle is horizontal when $width > height$, is vertical when $width < height$, and is square when $width = height$. The feature *count_cell* describes how many cells are in the region (i.e., the area of the rectangle). Additionally, we count the

Final edited form was published in "International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. Porto 2016", S. 78 - 100. ISBN: 978-3-319-99701-8
https://doi.org/10.1007/978-3-319-99701-8_4

Table 3. Region features.

Nr.	Simple	Nr.	Compound
1	IS_HORIZONTAL?	9	SIMILARITY_{TOP,BOTTOM,LEFT,RIGHT}#
2	IS_VERTICAL?	10	DISSIMILARITY_{TOP,BOTTOM,LEFT,RIGHT}#
3	IS_SQUARE?	11	EMPTINESS_{TOP,BOTTOM,LEFT,RIGHT}#
4	COUNT_CELLS#	12	INFLUENCE_{TOP,BOTTOM,LEFT,RIGHT}#
5	COUNT_ITS_KIND#		
6	DISTANCE_FROM_ITS_KIND#		
7	DISTANCE_FROM_ANY_KIND#		
8	PREDICTED_LABEL		

number of regions in the worksheet having the same label (i.e., its own kind) as the current region. Simple feature number 6 and 7 respectively capture the smallest Euclidean distance of this region to a region of the same label and to a region of any label. Finally, the *predicted_label* stores the label (i.e., assigned from the cell classification task as described in Sect. 4) of the cells in the region.

With the *compound features* we analyze the neighborhood of a region, similarly to the naïve approach (see Sect. 5.1). However, this time the neighborhood is made of other regions, instead of cells. Therefore, below we refine some of the previously used concepts and add some new ones.

- **Current Region (R):** The region whose neighborhood we are studying.
- **Direction (D):** Can be Top, Bottom, Left, or Right.
- **Neighbor (N):** Any region other than the current one.
- **Nearest neighbors (NNs):** The neighboring regions with the smallest Euclidean distance from the current region in the specified direction.
- **Similar neighbors (SNs):** Neighbors that have the same label as the current region.
- **Dissimilar neighbors (DNs):** Neighbors that have different label from the current region.

As shown in the above list, we study the neighborhood in four directions, omitting intermediate directions like top-right. Moreover, we use the concept of nearest neighbor, to distinguish from other neighbors in the vicinity of the current region. However, as we shall see in the following paragraphs, we consider more distant neighbors as well. Additionally, we examine the label of the neighbors, to determine if they are similar or dissimilar to the current region. Further more, we are interested in regions with specific label, as we later show in the definition of *influence.*

It is important to emphasize that unlike the naïve approach, the number of neighboring regions varies. Moreover, they might come in different sizes (considering both width and height). Therefore, we need a method to weight the importance of each neighbor. For this we utilize two measures: *overlap-ratio* and *distance*. The former quantifies how much of the specified direction is dominated by a neighbor. The latter how far or close is the neighbor. Clearly, a nearer neighbor should be given more weight.

Equation 1, illustrates how the overlap ratio is calculated. Here, r stands for the current rectangle, n for the selected neighbor, and d for the current direction. For a neighbor at the top and bottom we measure the overlap by projecting it and the region itself to the x-axis. The length of the segment shared by both these regions represents the overlap. For left and right neighbors we do the same, but instead using the projections to the y-axis. We transform the overlap into a ratio by dividing it with the width or the height (i.e., respectively, the length of the vertical or horizontal edge) of the current region.

$$\begin{gathered} OverlapRatio(r, n_i, d) = \frac{Overlap(r, n_i, d)}{EdgeLength(r, d)} \\ \text{where} \quad n_i \in Neighbors(r, d) \quad \text{and} \quad d \in Directions \end{gathered} \tag{1}$$

Once we have the overlap ratio and the distance to the neighbor, we can calculate its weight as shown in Eq. 2. In the denominator, we add one to the distance to account for cases where the latter is zero. Clearly, this equation captures the intuition that the weight for a neighbor should increase for smaller distances and bigger overlap ratios.

$$weight_i = OverlapRatio(r, n_i, d) \cdot \frac{1}{1 + Distance(r, n_i)} \tag{2}$$

We can now define the *similarity* for a region and its neighbor, as shown in Eq. 3. Similarity takes a value greater than zero when the neighbor is a SN and is one of the NNs. In such scenario, the value of the *similarity* equals the weight of the neighbor.

$$similarity_i = \begin{cases} 0 & Label(r) \neq Label(n_i) \;\; \vee \;\; n_i \notin Nearest(r, d) \\ weight_i & otherwise \end{cases} \tag{3}$$

$$dissimilarity_i = \begin{cases} 0 & Label(r) = Label(n_i) \;\; \vee \;\; n_i \notin Nearest(r, d) \\ weight_i & otherwise \end{cases} \tag{4}$$

Likewise we calculate the *dissimilarity* for a neighbor, as shown in Eq. 4. The only difference from the definition of *similarity* is that here the neighbor must be a DN, in addition to being a NN.

Influence goes beyond the immediate neighborhood (i.e., the nearest neighbors). It quantifies how much distant neighbors influence the current region. For example, this can be useful in the scenario where two Correct regions of the same label are separated by a Misclassified region. Knowing that there is considerable influence from a SN might save the region from accidentally being marked as Misclassified. Good influence can come also from other labels. For instance, a strong Data influence from the bottom neighborhood, can reinforce the belief that Header is the most plausible label for the current region.

Influence is tightly coupled with the selected label at that time, as shown in Eq. 5. Here, l stands for the label. Also, we have updated the function *Nearest* by adding the optional parameter *label*. When this parameter is set, the function returns only the nearest neighbors for a specific label in the given direction. Influence gets a value greater than zero when there exist at least one neighbor with the requested label. When there are multiple such neighbors, we prefer the influence from the nearest ones.

$$influence_i = \begin{cases} 0 & Label(n_i) \neq l \ \vee \ n_i \notin Nearest(r, d, l) \\ weight_i & otherwise \end{cases} \tag{5}$$

All the previous equations hint that there can be more than one nearest neighbor. In order to get the total value of a feature for a direction, we need to sum up the values for the individual NNs. Equations 6 and 7 respectively show how to perform this for *similarity* and *influence*. We can calculate the total *dissimilarity* for a direction the same way.

$$total\_similarity_d = \sum_{i=1}^{|Nearest(r,d,Label(r))|} Similarity(r, n_i, d) \tag{6}$$

$$total\_influence_{d,l} = \sum_{i=1}^{|Nearest(r,d,l)|} Influence(r, n_i, d, l) \tag{7}$$

Emptiness, the last compound feature, is the feature that tries to capture the (partial or complete) non-existence of nearest neighbors in a direction. Emptiness takes the maximum value when there are no neighbors in a direction. When the NNs partially overlap with the current region, *emptiness* takes a value between zero and one. Equation 8 illustrates how to calculate the value of this feature for a specific direction. Note in this equation that we do not set the label parameter for the *Nearest* function. Thus, it returns the complete set of NNs.

$$total\_emptiness_d = 1 - \sum_{i=1}^{|Nearest(r,d)|} OverlapRatio(r, n_i, d) \tag{8}$$

We can add additional flavors to the compound features by aggregating them to the level of row (left and right), column (top and bottom), and overall neighborhood (all four directions). Equation 9 illustrates how to calculate the overall value, using as example the *similarity* feature. As shown, we normalize the value from a direction using the ratio of the edge length (in that direction) to the perimeter of the current region.

$$overall\_similarity = \sum_{j=1}^{|Directions|} \left(\frac{EdgeLength(r, d_j)}{Perimeter(r)} \cdot similarity_{d_j} \right) \tag{9}$$

Misclassification Identification. We define Misclassification Identification (MI) as a machine learning task, whose goal is to distinguish real Misclassified regions from the dataset of candidate regions. For this binary classification problem we have considered all the simple features mentioned in the previous section. Additionally, we use the compound features for all four directions and their three flavors (i.e., row, column, overall). However, for *influence* we define a special version. We only consider the influence from neighbors of the same label, and omit the influence from the rest. This brings the total number of features, used for MI, to 36 (i.e., 8 simple + 28 compound).

Table 4. Comparing classifiers for misclassification identification task.

	Rand. forest *-I 100*	SMO RBF *-C 19.0, -G 0.1*	Logistic *-R 1.0E-14*	JRIP *-N 10.0*
F1 measure	0.97	0.96	0.95	0.96
True negative rate	0.64	0.58	0.47	0.60
False negative rate	0.03	0.03	0.04	0.04

For our evaluation we experimented with several classification algorithms. Again, we used the Weka tool. We firsed tuned the parameters of the individual classifiers. Subsequently, we used Weka Experimenter[3] to run 10 fold cross-validation with 10 repetitions. The results are displayed in Table 4. The values represent the average of all runs. Random forest achieves the highest F-measure and simultaneously has the highest true negative rate. With what regards false negative rate, there is not substantial difference between the classifiers. Considering these results, we selected the Random Forest classifier for our subsequent analysis.

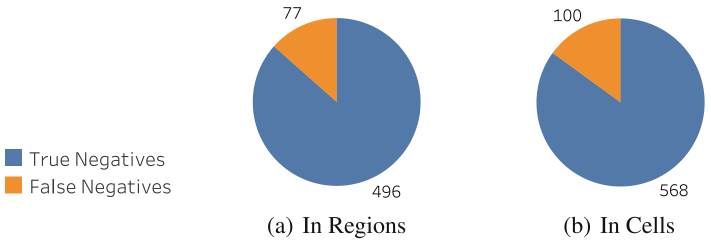


Fig. 14. Misclassification identification results.

In Fig. 14 we display the results from one of the cross-validation repetitions (*seed* = 1). We have provided the numbers in terms of regions and in terms

[3] https://sourceforge.net/projects/weka/files/documentation/3.8.x/

Final edited form was published in "International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. Porto 2016", S. 78 - 100. ISBN: 978-3-319-99701-8
https://doi.org/10.1007/978-3-319-99701-8_4

of individual cells. We get more False Negative cells (i.e., wrongly flagged as Misclassified) in comparison to the naïve approach. However, the number of True Negative cells (i.e., correctly predicted Misclassified) is several times bigger for the RBA approach.

Relabeling. We define the task of relabeling as that of predicting the most plausible true label for regions flagged as Misclassified. For this task we use all the simple features, except of *predicted_label*. From the compound features we use only *influence*, capturing it for each label and direction. We add to the set of directions also the three combined variants (flavors): row, column, and overall. With this addition, the total number of features used for relabeling becomes 42 (i.e., 7 simple + 35 influences).

The dataset to train our model for relabeling comes from the original annotated cells (i.e., the ground truth). Similarly to what we did with the predicted labels, we construct rectangular regions from the annotations. At the end, we keep only those of size three or smaller for training. This brings the total number of regions in this dataset to 11, 934.

Table 5. Relabeling: trained on annotated regions.

	Rand. forest *-I 350*	SMO RBF *-C 16.0 -G 1.0*	Logistic *-R 1.0E-8*	JRIP *-N 2.0*
F1 measure	0.64	0.59	0.67	0.49
True negative rate	0.65	0.59	0.67	0.49
False negative rate	0.11	0.13	0.10	0.16

For our evaluation we used the same classification algorithms as for misclassification identification. In a similar fashion, we first tuned the parameters of the classifiers on the training datasets. Subsequently, we evaluated their performance on the 573 regions identified as Misclassified. The results are provided in Table 5.

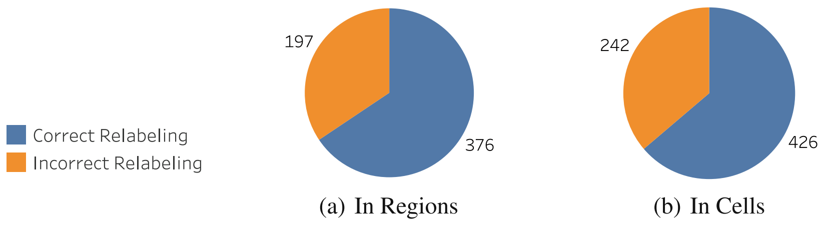


Fig. 15. Relabeling results.

Figure 15 displays the relabeling results for Logistic Regression (LR) classifier. We pick this classifier, since as shown in Table 5 it achieves the best results.

Final edited form was published in "International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. Porto 2016", S. 78 - 100. ISBN: 978-3-319-99701-8
https://doi.org/10.1007/978-3-319-99701-8_4

Again, we provide the numbers in regions and in cells. Although, we managed to predict the true label for most of the regions flagged Misclassified, there is a considerable number of wrong predictions.

One possible solution to decrease the number of incorrect predictions is to use the predicted class (label) probabilities, instead of fixed membership. By default, the LR classifier assigns the class with the highest probability to an instance. We can interfere in this process, and only relabel those regions for which the prediction has high confidence. Effectively, this means setting a threshold for the class probability.

We assessed the validity of this approach by analyzing the probabilities (scores) assigned by the LR classifier during the relabeling task. For each region we have recorded the highest predicted class score. Then we created the distinct list of these scores from the whole task. For each value in the list we identify the regions that got a score greater than or equal. Then we calculate the *difference* between the number of correctly relabeled and the number of incorrectly relabeled. The results are provided in Fig. 16. The largest *difference* is 192, and is achieved for score 0.59. For this score we get 363 correct versus 171 incorrect predictions. However, we decided to be more conservative and set the (score) threshold 0.83. We get a better trade off, since we only get 113 wrong region (re-)labels, and a considerable number of 300 correct region (re-)labels.

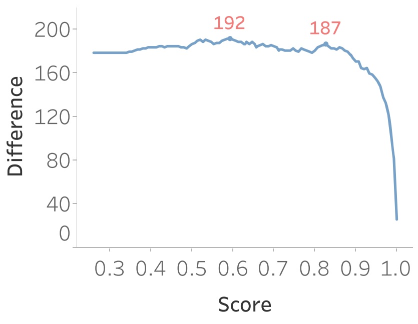


Fig. 16. Confidence score analysis.

6 Related Work

6.1 Spreadsheet Layout Inference

Comparing with related work, we have exclusively focused on the cell level. In this way we can infer the layout of arbitrary tables and arrangements in spreadsheets. Related work proposes approaches that work with larger structures, such as rows and columns. These fail to recognize that the contained cells could have different layout roles. In other words, these approaches lose important information, by assuming homogeneous structures. Furthermore, working on a cell level

Final edited form was published in "International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. Porto 2016", S. 78 - 100. ISBN: 978-3-319-99701-8
https://doi.org/10.1007/978-3-319-99701-8_4

enables us to make use of many features, not considered before by related work. Currently, there is no scheme that allows to aggregate these features for larger structures without compromising the accuracy. Above all, our approach is automatic, using machine learning techniques. Thus we overcome the cumbersome task of manually defining heuristics (as some of the related work do), which are limited by the human nature.

At [2] the authors present their work on what they call data frame spreadsheets (i.e., containing attributes or metadata regions on the top/left and a block of numeric values). Using linear-chain, conditional random field (CRF), they perform a sequential classification of rows in the worksheet, in order to infer its layout. Their next immediate focus is extracting the hierarchies found on the top (Header) and left (Attribute) regions. This aspect of their work is further extended at [11]. Additionally, in their first paper, they have experimented with the extraction of the data in the form of relational tuples. They do this based on the information they inferred about the structure of a data frame.

At [12], the authors present their work on schema extraction for Web tabular data, including spreadsheets. They extensively evaluated various methods for table layout inference, all operating at the row level. The CRF classifier combined with their novel approach for encoding cell features into row features (called "logarithmic binning") achieves the highest scores.

The paper [13] presents work on header and unit inference for spreadsheets. Unlike us, the authors take a more software engineering perspective. They utilize the inferred table structure to identify unit errors. The authors have defined a set of heuristics-based spatial-analysis algorithms, and a framework that allows them to combine the results from these algorithms. Additionally, they have evaluated their approach in two datasets, containing 10 and 12 spreadsheets, respectively.

At [14], the authors present DeExcelerator, a framework which takes as input partially structured documents, including spreadsheets, and automatically transforms them into first normal form relations. For spreadsheets, their approach works based on a set of simple rules and heuristics that resulted from a manual study on real-world examples. Their framework operates on different granularity levels (i.e., row, column, and cell), considering the content, formating, and location of the cell/s. They evaluated their system on a sample of 50 spreadsheets extracted from data.gov, using human judges (10 database students).

6.2 HTML Tables

One of the typical ways to present information (facts) is by organizing data in a tabular format. As a result, the problems of table recognition and layout discovery have been encountered by various research communities. Some of the most recent studies are related to HTML (Web) tables. In [15], decision trees and support vector machines (SVM) are considered to differentiate between genuine and non-genuine Web tables. The authors defined structural, content type, and a word group features. The [16] reports the study of large sample of Web tables, which yielded a taxonomy of table layouts. It also discusses heuristics, which are based on features similar to the paper above, to classify Web tables into the proposed taxonomy. In [17], the authors describe the creation of the Dresden Web Table Corpus, by proposing a classification approach that works on the level of different table layout classes.

7 Conclusions

In conclusion, we have presented an updated version of our work, first introduced at [1]. Via cell classification, we aim at identifying the layout and structure of the data in spreadsheets. We defined five labels (classes), based on literature review and our thorough empirical analysis of spreadsheets coming from various domains. Using our specialized tool we initially annotated a considerable sample of worksheets, and subsequently extracted a big variety of predefined features for each annotated cell. The latter, composes our gold standard, which we used for feature selection and for evaluating classifiers. Our experiments show that with the selected features and a Random Forest classifier we can achieve high overall accuracy.

Moreover, we have devised a strategy to fix some of the incorrect classification. Our aim is to get rid of random noise (misclassifications) that might occur in worksheets where we mainly make correct predictions. We attempt to go beyond the rather simplistic approach we discussed at [1], by proposing a three-step process. Initially we cluster our cells into rectangular regions. Similarly to the cell classification, we characterize these regions with a variety of sophisticated features. Subsequently, we use a classifier to identify regions that only contain misclassified cells. Later, we attempt to predict their true label, using another specialized classifier. Our evaluation shows that we perform well at the misclassification identification task, but not as good for the relabeling task. In the future, we plan to define more features for the latter task, and experiment with other classification algorithms as well.

Acknowledgments. This research has been funded by the European Commission through the Erasmus Mundus Joint Doctorate "Information Technologies for Business Intelligence - Doctoral College" (IT4BI-DC).

References

1. Koci, E., Thiele, M., Romero, O., Lehner, W.: A machine learning approach for layout inference in spreadsheets. In: Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016), KDIR, Porto, Portugal, 9–11 November 2016, vol. 1, pp. 77–88 (2016)
2. Chen, Z., Cafarella, M.: Automatic web spreadsheet data extraction. In: SSW 2013, p. 1. ACM (2013)
3. Barik, T., Lubick, K., Smith, J., Slankas, J., Murphy-Hill, E.: FUSE: a reproducible, extendable, internet-scale corpus of spreadsheets. In: MSR 2015 (2015)
4. Hermans, F., Murphy-Hill, E.: Enron's spreadsheets and related emails: a dataset and analysis. In: Proceedings of ICSE 2015. IEEE (2015)

5. Fisher, M., Rothermel, G.: The EUSES spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. In: SIGSOFT 2005, vol. 30, pp. 1–5. ACM (2005)
6. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth, Belmont (1984)
7. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., Boston (1993)
8. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001)
9. Vapnik, V.: Estimation of Dependences Based on Empirical Data. Springer Series in Statistics. Springer-Verlag New York, Inc., New York (1982)
10. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods - Support Vector Learning. MIT Press (1998)
11. Chen, Z., Cafarella, M.: Integrating spreadsheet data via accurate and low-effort extraction. In: SIGKDD 2014, pp. 1126–1135. ACM (2014)
12. Adelfio, M.D., Samet, H.: Schema extraction for tabular data on the web. In: VLDB 2013, vol. 6, pp. 421–432 (2013)
13. Abraham, R., Erwig, M.: Header and unit inference for spreadsheets through spatial analyses. In: VL/HCC 2004, pp. 165–172. IEEE (2004)
14. Eberius, J., Werner, C., Thiele, M., Braunschweig, K., Dannecker, L., Lehner, W.: DeExcelerator: a framework for extracting relational data from partially structured documents. In: CIKM 2013, pp. 2477–2480. ACM (2013)
15. Wang, Y., Hu, J.: A machine learning based approach for table detection on the web. In: WWW 2002, pp. 242–250. ACM (2002)
16. Crestan, E., Pantel, P.: Web-scale table census and classification. In: WSDM 2011, pp. 545–554. ACM (2011)
17. Eberius, J., Braunschweig, K., Hentsch, M., Thiele, M., Ahmadov, A., Lehner, W.: Building the dresden web table corpus: a classification approach. In: BDC 2015. IEEE/ACM (2015)