Julian Eberius, Katrin Braunschweig, Markus Hentsch, Maik Thiele, Ahmad Ahmadov, Wolfgang Lehner

Building the Dresden Web Table Corpus: A Classification Approach

Erstveröffentlichung in / First published in:

2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC). Limassol, 07.-10.12.2015. IEEE, S. 41-50. ISBN 978-0-7695-5696-3.

DOI: http://dx.doi.org/10.1109/BDC.2015.30

Building the Dresden Web Table Corpus: A Classification Approach

Julian Eberius, Katrin Braunschweig, Markus Hentsch, Maik Thiele, Ahmad Ahmadov and Wolfgang Lehner
Technische Universität Dresden
Faculty of Computer Science, Database Systems Group
01062 Dresden, Germany
Email: firstname.lastname@tu-dresden.de

***Abstract*—In recent years, researchers have recognized relational tables on the Web as an important source of information. To assist this research we developed the Dresden Web Tables Corpus (DWTC), a collection of about 125 million data tables extracted from the Common Crawl (CC) which contains 3.6 billion web pages and is 266TB in size. As the vast majority of HTML tables are used for layout purposes and only a small share contains genuine tables with different surface forms, accurate table detection is essential for building a large-scale Web table corpus. Furthermore, correctly recognizing the table structure (e.g. horizontal listings, matrices) is important in order to understand the role of each table cell, distinguishing between label and data cells. In this paper, we present an extensive table layout classification that enables us to identify the main layout categories of Web tables with very high precision. We therefore identify and develop a plethora of table features, different feature selection techniques and several classification algorithms. We evaluate the effectiveness of the selected features and compare the performance of various state-of-the-art classification algorithms. Finally, the winning approach is employed to classify millions of tables resulting in the Dresden Web Table Corpus (DWTC).**

***Keywords*-Data preprocessing, Web mining, Machine learning**

I. Introduction

The Web has developed into a comprehensive resource not only for unstructured or semi-structured data, but also for relational data. Millions of relational tables embedded in HTML pages or published in the course of Open Data/Open Government initiatives provide extensive information on entities and their relationships from almost every domain. Researchers have recognized these Web tables as an important source of information for applications such as factual search [1], entity augmentation [2], [3], [4] and ontology enrichment [5].

Whereas in the past only big search engine companies where able to crawl and make use of these large volume data, the situation changed with the advent of the Common Crawl Foundation, a non-profit foundation that crawls the Web and regularly publishes the resulting Web corpora for public usage [6]. The data is hosted on Amazon S3, and could thus be easily processed using EC2 instances for example. We exploit this new opportunities and develop the Dresden Web Table Corpus [7] (DWTC) a large corpus consisting of 125 million unique tables extracted from the WARC files of the July 2014 version[1] of the Common Crawl (CC). While the implementation of the Web table extractor[2] can be understood by checking out the code we provide on GitHub, this paper focuses on the conceptional challenges of table detection and classification.

Accurate table detection is essential for the extraction of table data from the Web, as the vast majority of table structures in HTML pages are used for layout purposes, and only a small share contains genuine tables. These tables, however, are not uniformly structured. Instead, different layout types such as vertical lists or matrices are used to represent the data, depending on the content and purpose of the table. Layout classification allows us to identify the main layout categories of Web tables. Based on these categories, we can then make more accurate assumptions about table characteristics, such as the location of the header, and, ultimately, the meaning of the table. However, in the literature, table layout classification has received only little attention, whereas Web table detection has been studied in greater detail. Instead of distinguishing between different layout types, many table analysis and interpretation techniques simply assume a uniform layout across all tables. In many cases, a simple layout similar to the layout commonly used for database tables, with attribute labels at the top of each column and no designated stub, is expected. It is clear that assuming a single uniform layout either excludes a substantial number of tables from the extraction process or leads to inaccurate results.

Therefore, this paper focuses on incorporating layout classification in the Web table extraction process. In particular, we study two alternative approaches. The first, which has been proposed in the literature, combines layout classification with table detection into a single classification task. The second treats both problems as separate consecutive classification tasks. We conduct a comparative evaluation on real-world data to establish, which approach is more effective. In detail, we address the following aspects:

- **Classification Scheme:** Incorporating lessons learned from related work, we propose a classification scheme for genuine Web tables, that distinguishes three main

[1]http://blog.commoncrawl.org/2014/08/july-2014-crawl-data-available/
[2]https://github.com/JulianEberius/dwtc-extractor

layout types: vertical listings, horizontal listings, and matrix tables.

- **Feature Selection:** We consolidate and extend a wide range of features proposed in the literature for each of the classification tasks. Using correlation-based feature selection, we evaluated the relevance of each feature with respect to the classification problems.
- **Experimental Evaluation:** We conduct an experimental evaluation on a corpus of Web tables extracted from the Common Crawl. We evaluated our different approaches, comparing various classification algorithms.

This paper is organized as follows: In Section II we review related work in the field of table detection as well as table layout classification for Web tables. We then formally define the classification problems in Section III. In Section IV we consolidate the various table features proposed in the literature and extend them and for Web tables to facilitate an accurate classification. In an experimental evaluation ((Section V)) on real-world data, we evaluate the effectiveness of the selected features and compare the performance of various state-of-the-art classification algorithms. Finally, we apply the best classification approach to build our DWTC and conclude with a summary of our findings (Section VI).

II. Related Work

Identifying genuine tables and discriminating between different table layouts are the key for building a high-quality Web table corpus. Both tasks can be regarded as classification tasks, yet each with a very different objective. In this section we collect and consolidate various table features proposed in the literature and evaluate different classification algorithms in order to come to a reliable solution.

A. Genuine Table Detection

A number of table detection and analysis approaches have been proposed that specifically target tables on the Web. [8] address the detection of HTML tables by proposing a set of heuristic rules and cell similarity measures that distinguish relational tables from tables used for layout purposes. These simple rules eliminate tables with less than two cells as well as tables that contain a significant amount of hyperlinks, forms or figures. A set of cell similarity measures are used to filter out any remaining layout tables. An overall F-measure of 86.5% is reported.

Similar heuristic rules are implemented by [9] for the detection of genuine Web tables. A table is regarded as genuine, if it is a leaf table (i.e. it does not contain another table in a cell), contains multiple rows and columns, and the size of each table entry is below a predefined threshold. Furthermore, genuine tables do not contain lists, forms, images or other non-text formatting tags. This detection approach achieves an F-measure of 88.01%. The authors also point out that syntactic and semantic coherency within the rows or columns of a table are important characteristics to identify genuine tables. However, no specific measures for coherency are proposed and these characteristics are not included in the detection algorithm.

[10] are the first to apply machine learning techniques to the detection of genuine Web tables. Decision trees as well as support vector machines (SVM) are considered for the task. A large set of features, including structural features and content type features, are utilized. In addition, a *word group feature* is proposed that treats tables as text documents and genuine table detection as a document categorization problem. An overall evaluation reports a maximal F-measure of 95.88% for decision trees and 95.89% for support vector machines with an RBF kernel. With an F-measure of 95.73% achieved using only structural and content type features we can conclude that the complex word group feature are only of minor relevance.

The authors in [11] combine simple rules and statistical classifiers to detect relational tables in a huge corpus of 14.1 billion HTML tables extracted from a Web crawl. The amount of genuine tables is estimated to be only 1.1% of the entire corpus, however attribute/value tables are not included, as they are not regarded as relational tables. Similar to previous techniques, the majority of obviously non-relational tables (89.4%) is eliminated using a set of simple heuristic rules. The remaining tables are classified using rule-based classifiers trained on 7 simple features inspired by the work of [10]. Tuning their classifier to maximize recall at the prospect of loss of precision, [11] report an average precision of 69% and an average recall of 84% for a subset of the corpus, consisting of several thousand tables. The rules and features utilized in this approach do not present novel contributions. However, [11] are the first to apply a table detection algorithm to a corpus at Web scale.

A significantly different classification approach is proposed by [12], who utilize the structural information provided by the DOM tree of a Web page to detect genuine Web tables. The DOM tree of the table as well as the DOM tree of the surrounding document are directly used as features in the classification tasks. A specialized parse tree kernel is proposed for SVM-based classification. Additionally, the content type features proposed by [10] are used to incorporate content coherency. On a test corpus of several thousand tables, [12] report an F-measure of 98.58% for an SVM-based classification combining structural and content type features. This approach outperforms any previous technique by incorporating the structural characteristics of the DOM trees. While previously proposed features were mostly applicable to tables independent of the file format, the features proposed here are specific to HTML tables.

B. Table Layout Classification

In addition to distinguishing between genuine and non-genuine tables, more fine-grained classification schemes for Web tables have been proposed, which take the layout and

Final edited form was published in "2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC). Limassol 2015", S. 41-50, ISBN 978-0-7695-5696-3
http://dx.doi.org/10.1109/BDC.2015.30

structure of the tables into account.
The first classification scheme has been proposed by [13]. At the highest level, Web tables are categorized as either *relational knowledge* or *layout* tables. Relational tables are further divided into seven categories: listings (vertical and horizontal), attribute/value tables, matrix tables, calendars, enumerations (i.e. lists), and forms. Layout tables are divided into two categories: navigational tables and formatting tables. A wide range of features characterizing the structure and cell content are used to classify the tables. Before classifying the tables, [13] apply a simple rule-based filter to eliminate tables that are obviously not relational. The applied rules are similar to those proposed by [9], and filter out tables with less than two rows or columns, and cells with more than 100 characters. The authors report a reduction by more than 80%, with 93% of the eliminated tables identified as layout tables. Remaining tables are classified using a *gradient boosted decision tree* model achieving an overall accuracy of 75.2%.
The work presented by [13] has been further extended by [14], who consider two layers of classification. The first layer is similar to the classification scheme proposed before, classifying tables into one of five categories: vertical, horizontal, matrix, formatting and navigational. Again, the classification scheme includes genuine and layout tables, but does not consider lists or forms. A secondary classification scheme further classifies genuine tables based on structural characteristics, such as the occurrence of merged cells or nested tables. The authors consider a set of 25 features, with 20 features adopted from [13] and 5 features added to address multivalued tables. A neural network is used to classify the Web tables. The authors note an increase in classification performance for all categories, except for matrix tables.
Both table layout classification approaches are very similar and incorporate the detection of genuine relational tables into the classification of table layouts. The classifiers achieve good results for both classification aspects. However, no comparison to a two-stage classification approach that performs each task separately is provided.

III. Classification Problem

After reviewing previous work related to Web table detection and layout classification, we now take a closer look at the specific classification approaches we wish to compare. First, we recall the tasks we want to carry out as part of the classification process: (1) the detection of genuine tables, and (2) the identification of the layout type of these tables.

A. Genuine Tables and Table Layout

The objective of table detection is to identify tables that represent actual genuine tables. These tables frequently contain simple strings or numeric values in the table cells. Moreover, they feature syntactic similarities between values belonging to the same attribute domain, reflected by coherent rows or columns. In the detection process, candidate tables are classified as either *genuine* or *non-genuine*. Consequently, table detection can be regarded as a binary classification problem. In the context of Web pages, non-relational-like tables are mostly HTML tables used for layout purposes or to represent menu structures.
The objective of layout type identification is the analysis of a table's logical structure. Although table structures on the Web are very heterogeneous, several prominent structures can be identified, representing the main layout types. Specifically, we consider three main layout types: horizontal listings, vertical listings and matrix tables. Most genuine tables can be assigned to one of these layout types. The layout types we consider are based on the alignment of values of the same attribute within a table. Consequently, characteristic features to identify each type are the location of attribute labels and the coherence of values per row or column.

B. Classification Methodology

To carry out both of these tasks, we consider two different classification approaches, a *single-layer* and a *double-layer* approach. The *single-layer approach* combines both classification problems in a single classification task. Therefore, non-genuine or layout tables are regarded as one class in the classification scheme, similar to the layout classes *Vertical*, *Horizontal* and *Matrix*. An additional class *Other* is included for all tables that do not fit into any of the previous classes. The same features, classifier and training data are used to classify all Web tables in this approach. Consequently, selected features must be suitable for table detection as well as layout identification.
In contrast, a *double-layer approach* performs two separate classification tasks consecutively, using the output of the table detection task as the basis for table layout identification. As only the layout type of genuine tables is of interest, non-genuine tables are discarded after the first classification step. As both tasks are performed independently, different features, classifiers and training data can be applied. In Section V-C3 we compare both approaches regarding their accuracy.

IV. Feature Specification

We consider features at two different levels of granularity: global features and local features taking into account structural as well as content features, utilizing the presence of HTML markup if applicable. The features address different aspects of table detection, layout classification or both. We do not specifically distinguish between features suitable for table detection and layout classification, respectively, as one of the approaches we intent to evaluate incorporates both tasks into a single classification problem.

A. Global Features

Global features describe the table as a whole and, thus, are computed once per table. As global features, we take into

Final edited form was published in "2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC). Limassol 2015", S. 41-50, ISBN 978-0-7695-5696-3
http://dx.doi.org/10.1109/BDC.2015.30

account the general table structure of rows and columns, the overall consistency of cell entries, the distribution of different data types, as well as the occurrence of designated header tags. These features incorporate and extend the features proposed by [13] as well as [10].

1) Table Structure: Table structure features describe the size and orientation of a table. They take into account the extent of rows, column and cells. As global features, we consider the maximal extent as well as the average extent across the table:

- `MAX_ROWS`: Maximal number of cells per row, which are not created by a `<SPAN>` tag.
- `MAX_COLS`: Maximal number of cells per column, which are not created by a `<SPAN>` tag.
- `MAX_CELL_LENGTH`: Maximal number of characters per cell.
- `AVG_ROWS`: Number of cells per row, averaged across all rows.
- `AVG_COLS`: Number of cells per column, averaged across all columns.
- `AVG_CELL_LENGTH`: Average number of characters per cell.

These features provide a first indication whether a candidate table has the regular structure that is common for genuine tables. Furthermore, they detect very small tables that are unlikely to represent relational content.

2) Consistency and Variation: In addition to the previous features, which describe the general extent of a table, we also consider the variation encountered in the extent of different table segments. From this variation, we can derive a more precise measure of the regularity of the table structure. In particular, we consider the standard deviation of the size of rows, columns and cells:

- `STD_DEV_ROWS`: Standard deviation of the number of cells per row.
- `STD_DEV_COLS`: Standard deviation of the number of cells per column.
- `STD_DEV_CELL_LENGTH`: Standard deviation of the number of characters per cell.

In addition to the variance in the table structure, we consider the consistency of the table entries with respect to their size. For each cell c, we take the size s as the number of characters and compute the *cumulative length consistency (CLC)* per row or column as follows, where s_{avg_i} is the average cell size of table segment (i.e. column or row) i:

$$CTC_i = \sum_c 0.5 - x_i, \text{ where } x_i = \min\left(\frac{|s_c - s_{avg_i}|}{s_{avg_i}}, 1\right) \tag{1}$$

These consistency scores are averaged across all rows (CLC_R) and columns (CLC_C) and the maximum is returned as the global length consistency feature.

3) Content Ratio: The content ratio features identify what kind of content or data type is predominant in a table. Layout tables often contain many images or hyperlinks, while relational tables rather contain simple data types such as character strings or numeric values. We consider five different content types: images, forms, hyperlinks, alphabetic characters and numeric characters. In addition, we look for empty cells. An additional category *Other* is added to account for cell entries that do not match any of the previous categories. The following list shows all content ratio features we are considering:

- `RATIO_IMG`: Cells containing `<IMG>` tag.
- `RATIO_FORM`: Cells containing `<FORM>` tag.
- `RATIO_HYPERLINK`: Cells containing `<A>` tag.
- `RATIO_ALPHABETIC`: Cells with predominantly alphabetic characters.
- `RATIO_DIGIT`: Cells with predominantly numeric characters.
- `RATIO_EMPTY`: Empty cells.
- `RATIO_OTHER`: Cells not matching above categories.

The ratio of cells containing content of a specific type t is then defined as follows, where n is the number of cells in a table and t_i is the content type of cell i.

$$\texttt{RATIO} = \frac{1}{n}\sum_{i=1}^{n} x_i, \text{ where } x_i = \begin{cases} 1, & \text{if } t_i = t \\ 0, & \text{else} \end{cases} \tag{2}$$

In addition to the content ratio, we also consider the general content type consistency. The consistency is first analyzed per row or column and then averaged across the table. For each table segment, i.e. row or column, we compute the *cumulative type consistency (CTC)* as follows, where dt_i is the dominant content type in segment S_i:

$$CTC_i = \sum_{c \in S_i} x_c, \text{ where } x_c = \begin{cases} 1, & \text{if } t_c = dt_i \\ -1, & \text{else} \end{cases} \tag{3}$$

The consistency scores are then averaged across all rows (CTC_R) and columns (CTC_C). As the global content consistency feature, we take the maximum of these scores.

4) Header: In HTML tables, the designated `<TH>` tag can be used to define header cells Layout tables generally do not contain a header. Therefore, the presence of markup for header cells is a good indicator for genuine tables.The corresponding feature is defined as follows:

$$\texttt{HAS\_HEADER} = \begin{cases} 1, & \text{if table contains header markup} \\ 0, & \text{else} \end{cases} \tag{4}$$

B. Local Features

In addition to global features, we consider a number of local features, which are computed for subsets of the table. We follow the approach proposed by [13] and consider only

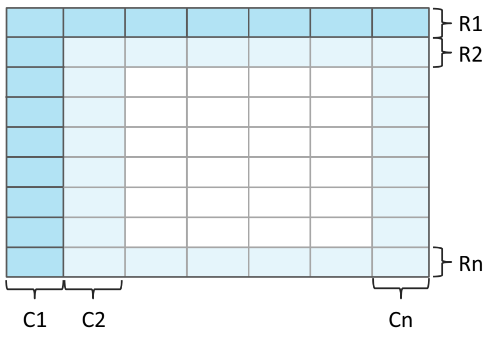

Figure 1. Local features are computed for the first two rows and columns as well as the last row and column of a table.

the first two rows, the first two columns, the last row and the last column of a table as segments for local features (see Figure 1). The first row and the first column of a table are potential locations for the label cells. Computing features for these segments and comparing them to their neighboring segments enables recognizing the orientation of headers in the tables, which is an important indicator for the layout type. Furthermore, considering the cell content at the beginning and end of each column or row provides an estimate of the coherence of the content as well as the orientation of the table. As local features, we again consider structural features as well as the content ratio.

1) Structural Features: As structural features of a table segment S_i, we take into account the average and variance of the size s of cells in the segment. Each of these features is computed once for each of the selected segments.

$$\texttt{LOCAL\_LENGTH\_AVG} = \frac{1}{|S_i|} \sum_{c \in S_i} s_c \tag{5}$$

$$\texttt{LOCAL\_LENGTH\_VARIANCE} = \frac{1}{|S_i|} \sum_{c \in S_i} (s_c - s_{avg_i})^2 \tag{6}$$

Additionally, we consider the distribution of virtual cells in the segments. Virtual cells are created within a table cell via `<SPAN>` tags. A single table cell can contain multiple `<SPAN>` tags. We compute the local span ratio as another feature for each table segment, considering all physical and virtual cells c in a segment.

$$\texttt{LOCAL\_SPAN\_RATIO} = \frac{1}{|S_i|} \sum_{c \in S_i} x_c$$

$$\text{, where } x_c = \begin{cases} 1, & \text{if } c \text{ is created by a } \texttt{<SPAN>} \text{ tag} \\ 0, & \text{else} \end{cases} \tag{7}$$

2) Content Ratio: Similar to the global content ratio features, we also consider the content type of cells in each local segment. We consider a wide range of content types, based on the presence of special tags or characters. The ratio is computed as before. However, in contrast to the global features, the content types considered here are not mutually exclusive. Multiple types can be assigned to the content of a single cell. The following list describes the content types and corresponding features:

- `LOCAL_RATIO_HEADER`: Cells containing `<TH>` tag.
- `LOCAL_RATIO_ANCHOR`: Cells containing `<A>` tag.
- `LOCAL_RATIO_IMAGE`: Cells containing `<IMG>` tag.
- `LOCAL_RATIO_INPUT`: Cells containing `<INPUT>` tag.
- `LOCAL_RATIO_SELECT`: Cells containing `<SELECT>` tag.
- `LOCAL_RATIO_FONT`: Cells containing `<B>`, `<I>`, `<U>` or `<FONT>` tags.
- `LOCAL_RATIO_BR`: Cells containing `<BR>` tag.
- `LOCAL_RATIO_COLON`: Cells containing colons.
- `LOCAL_RATIO_CONTAINS_NUMBER`: Cells containing numeric characters.
- `LOCAL_RATIO_IS_NUMBER`: Cells containing *only* numeric characters.
- `LOCAL_RATIO_NON_EMPTY`: Cells that are not empty.
- `LOCAL_RATIO_UNORDERED_LIST`: Cells containing `<UL>` tag.
- `LOCAL_RATIO_ORDERED_LIST`: Cells containing `<OL>` tag.
- `LOCAL_RATIO_COMMA`: Cells containing commas.
- `LOCAL_RATIO_BRACKET`: Cells containing brackets "(" or ")".

Images, hyperlinks or forms are not frequently found in genuine relational Web tables and, consequently, indicate that a table is used for layout purposes. Lists, commas or line breaks suggest that the content of a cell is not atomic. Yet, well-formed relational tables predominantly contain atomic values. Other tags, such as text formatting tags, can be used to estimate the coherence and orientation of a table, as attribute values from the same domain generally feature similar formats.

C. Pre-Selection Filters

Similar to previous Web table detection approaches, we can eliminate candidate tables, where it is obvious that they do not represent genuine tables, using a set of simple rules (see Section II). Similar to the rules proposed by [9], we expect genuine tables to feature *at least* two rows and two columns. Otherwise, the table resembles a list or single cell, which we do not consider a genuine table. Additionally, we remove tables that are invalid, i.e. the table structure does not form a valid HTML segment, as well as tables that cannot be displayed correctly, which indicates a low

Table II
DISTRIBUTION OF TABLE AND LAYOUT CLASSES IN THE CC.

Class	# of Tables
Layout	54.0%
Genuine	45.0%
Vertical Listings	34.0%
Horizontal Listings	46.7%
Matrix	3.7%
Others	15.6%

quality of the table in general. Surprisingly, the majority of HTML tables on the Web does not pass these simple filters, which amplifies the importance of accurate table detection algorithms to identify relevant, high quality tables amongst all these potential candidates.

V. EXPERIMENTAL EVALUATION

In order to establish, which of the two classification approaches we described in Section III achieves a higher overall accuracy for Web table detection and layout classification, we evaluate both approaches on a corpus of tables extracted from the CC. Therefore, in Section V-B we consider feature selection to initially reduce the dimensionality of the problem and ensure that we only use effective features for each classification task. The feature selection also provides insights into the different characteristics of table detection and layout identification. Finally, in Section V-C we compare the results of different classification algorithms to ensure that any difference in performance between the two approaches is not simply due to the suitability of the selected algorithm.

A. Gold Standard Definition

For the test corpus, we utilize the *Common Crawl* published in October 2014. From a subset of the crawl, we randomly extracted $26,654$ HTML tables. From these tables, we identified $24,623$ tables as obviously non-relational using the simple filter rules described in Section IV-C. After this initial filter step, $2,022$ tables remained in the corpus, which were manually labeled and used for evaluation. Table II shows the frequency with which each table class occurs in the corpus. In total, layout tables make up about 96% of all tables that we extracted from the Web. This percentage initially seems very high, but matches the estimate of 98.9% reported by [11]. Since matrix tables are significantly less frequent on the Web than tables with other layouts, our test set contains only a few instances of matrix tables. The small sample size for this class can potentially impact the performance of any classification algorithm. A similar issue has been reported by [14].

B. Feature Selection

In Section IV, we specified a large set of features for our classification problems, leading to a high-dimensional feature space. In total, we consider 127 features per table. Depending on the classification algorithm used, a large number of features often requires more training data to achieve good prediction results and the separation of classes can be more challenging in high-dimensional spaces. To reduce the dimensionality and to ensure that no redundant features are considered, we first perform feature selection.

We employ *correlation based feature selection (CFS)*, developed by [15], which is a filter approach that is independent of any specific classification algorithm. CFS recursively selects features that increase the so-called *merit* of the feature set, until no additional feature adds any benefit. To specify the merit of the feature set, CFS uses Pearson's correlation coefficient, which is biased towards features that are highly correlated with a class variable, but uncorrelated with other features in the set. As a result, irrelevant and redundant features are removed from the original feature set without loss of classification accuracy [15]. We use the implementation of CFS available in the WEKA machine learning toolkit [16] and apply it to each classification problem individually, i.e. table detection and layout classification, as well as to the consolidated classification task. The selection will indicate if any features are only relevant for one of the tasks.
From the initial set of 127 features, the CFS algorithm reduced the number of features to 29 for the table detection problem, to 23 for the layout identification problem and to 31 for the combined classification task. This means a significant reduction in dimensionality for each of the classification problems. Table I shows the features selected for each task, with content ratio features combined. Four features, namely `MAX_ROWS`, `MAX_CELL_LENGTH`, `STD_DEV_CELL_LENGTH` and `CUMULATIVE_LENGTH_CONSISTENCY` are not selected for any of the classification problems, most likely because they are correlated with other features. There are apparent differences in the feature sets selected for each problem. While, for instance, the presence of a header or <span> tags is useful to detect genuine tables, the average cell size as well as column and row sizes are more relevant for the identification of a table's layout. Similarly, content ratio features selected for each classification problem differ, as well. While the ratio of cells containing images or forms is relevant for table detection, it is not relevant for layout identification. Instead, the local ratio of header cells and cells containing colons, which also indicates label cells in some tables, are selected. The reduced feature sets clearly reflect the different objectives of the classification problems. Consequently, the features selected for the combined classification task appear to be a combination of features selected for the individual tasks. The different feature sets selected by CFS suggest that a double-layer classification approach has the potential to outperform a single-layer approach, by tailoring each classification

Final edited form was published in "2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC). Limassol 2015", S. 41-50, ISBN 978-0-7695-5696-3
http://dx.doi.org/10.1109/BDC.2015.30

Table I
FEATURES SELECTED FOR EACH CLASSIFICATION TASK.

Feature	Table Detection	Layout Identification	Combined
MAX_COLS		×	×
AVG_COLS		×	×
AVG_ROWS	×		
AVG_CELL_LENGTH		×	
STD_DEV_COLS		×	
STD_DEV_ROWS		×	
RATIO_X	×	×	×
CUMULATIVE_CONTENT_CONSISTENCY	×	×	×
HAS_HEADER	×		
LOCAL_LENGTH_AVG	×	×	×
LOCAL_LENGTH_VARIANCE	×	×	×
LOCAL_SPAN_RATIO	×		×
LOCAL_RATIO_X	×	×	×

step to the characteristics of the classification problem at hand. In the next section, we evaluate each approach using different classification algorithms to see if we can confirm this hypothesis.

C. Classifiers

To provide a comprehensive comparison and evaluation of the different processing approaches for table detection and layout classification, we conduct a range of experiments. We evaluate the classification performance using repeated random sub-sampling. We randomly split the dataset, using 90% of the data for training and 10% for validation. All results are averaged over 100 iterations. A common metric for classification performance is *accuracy*, which measures the number of correct predictions divided by the number of all predictions. However, especially for unbalanced datasets, were one category is predominant, the accuracy metric is often not sufficient to evaluate the prediction quality of a model. As our goal is to achieve a high classification performance for the less frequently represented classes *Vertical Listing*, *Horizontal Listing* and *Matrix*, we require more suitable metrics. Therefore, we use Precision, Recall and F-Measure to evaluate classification performance.
In our evaluation, we consider various classification algorithms, most of which have been successfully applied to similar tasks in the literature. As the first class of classification algorithms, we consider *decision trees*, which have been successfully applied to similar tasks by [10] as well as [13]. Specifically, we consider CART [17] (*SimpleCART* in WEKA), C4.5 [18] (*J48* in WEKA) and Random Forest [19]. As a second class of classifiers, we consider support vector machines (SVM) [20] that have been used before by [10] as well as [12] to detect relational HTML tables. In our experiments, we use an implementation of support vector machines provided in WEKA named *SMO*, which uses the sequential minimal optimization algorithm developed by [21]to train the classifier. We consider both a polynomial kernel and an RBF kernel.

1) Classification Algorithms: In the first set of experiments, we study the suitability of different classification algorithms with respect to the individual classification problems. Therefore, we evaluate the table detection and layout identification tasks individually as well as combined in a single classification task. For each classification problem, we use the full set of features and measure precision, recall and F_1 per class. Additionally, we measure the weighted average for each metric, using the class frequencies as weights to account for the unbalanced distribution of classes in the corpus.
First, we evaluate the table detection task, where we distinguish between layout tables and genuine tables. The performance measures are presented in Table III. Overall, the results show a very similar performance for all tested classification algorithms, with the Random Forest classifier performing best with respect to F_1. For all classifiers, the prediction quality for layout tables is much better regarding precision and recall than for genuine tables, which is a very heterogeneous class due to the different layout types. However, with 87.3% we achieve a significantly higher precision for genuine tables compared to 41% precision achieved by the approach proposed by [22], which is frequently used by other researchers.
Next, we study the quality of the layout identification using a Random Forest classifier. All layout tables are removed from the corpus, and we only consider classes *Vertical Listing*, *Horizontal Listing*, *Matrix* and *Other*. The results are shown in Table IV and confirm that layout classification is more challenging compared to table detection. Overall, we observe significantly more variation between the different classes. We achieve good results for vertical and horizontal listings, yet achieve only low precision and recall for matrix tables. This issue, which has also been reported by [14], is mainly due

Table III
EVALUATION OF THE TABLE DETECTION TASK: ALL MEASURES ARE REPORTED FOR THE FOLLOWING CLASSES: LAYOUT (L) AND GENUINE TABLES(G).

Classifier	Metric	L	G	Weight. Avg.
J48	Precision	95.11	82.64	89.38
	Recall	94.94	83.17	89.53
	F1	95.02	82.82	89.41
SimpleCART	Precision	95.17	81.61	88.93
	Recall	94.55	83.38	89.41
	F1	94.85	82.39	89.12
Random Forest	Precision	94.92	**87.30**	**91.42**
	Recall	**96.53**	82.12	89.90
	F1	**95.71**	**84.55**	**90.58**
SMO (Poly)	Precision	94.74	81.99	88.87
	Recall	94.79	81.81	88.82
	F1	94.75	81.81	88.80
SMO (RBF)	Precision	**95.43**	83.58	89.98
	Recall	95.19	**84.23**	**90.15**
	F1	95.30	83.81	90.02

Table IV
EVALUATION OF THE LAYOUT IDENTIFICATION TASK: ALL MEASURES ARE REPORTED FOR THE FOLLOWING CLASSES: VERTICAL LISTINGS (V), HORIZONTAL LISTINGS (H), MATRIX (M) AND OTHER (O).

Metric	V	H	M	O	Weight. Avg.
Precision	71.22	90.02	35.70	**80.98**	**80.18**
Recall	86.87	89.24	17.93	56.90	80.71
F1	77.98	**89.50**	21.69	**65.87**	79.35

to low number of matrix tables in the dataset. As a result, there are not enough training samples for this class to build a reliable model and make accurate predictions. That means a larger training set is necessary to improve the prediction performance for matrix tables.
Finally, we measure the performance of the combined classification task including the *Layout* class (see Table V). The prediction quality per class is similar to the results reported by [14]. The prediction of layout tables is very accurate, while we observe a lower precision and recall for horizontal and vertical listings. Again, we experience issues with matrix tables, due to their low frequency in the dataset. So far we have utilized the complete set of features for all classification problems. In the next set of experiments, we evaluate the impact of feature selection for each task.

2) Impact of Feature Selection: As detailed in Section V-B, we applied correlation-based feature selection (CFS) to identify the most relevant features for each classification problem and reduce the dimensionality by removing irrelevant or redundant features. We repeated all previous experiments for all classifiers, using only the selected features. In Table VI, we compare the weighted average F_1 measures for each experiment to evaluate the impact of the feature selection. In most cases, the performance measures achieved with only the selected features are very similar to the values achieved with all features. This confirms that the selection algorithm is successful in removing irrelevant and redundant features. We attribute these changes mainly to the selection of kernel parameters. Due to the reduced dimensionality, the optimal kernel parameters most likely differ from the previous settings, and a more comprehensive parameter optimization is required to achieve the best results. Apart from this, the dimensionality reduction has only little impact on the prediction performance. However, feature selection also reduces the computational costs for the classification tasks, which is especially relevant for the processing of huge Web corpora.

3) Single-Layer vs. Double-Layer Classification: After evaluating all individual classification tasks as well as the impact of feature selection, we now compare the single-layer and the double-layer classification approaches The single-layer approach corresponds to the combined classification task in the previous experiments. For the double-layer approach, we perform table detection and layout identification consecutively, using the tables classified as *genuine* in the first step as input for the second step. We use the same training set to train both classifiers. We then evaluate the test set and measure the overall prediction quality.
As the previous results indicated that Random Forest algorithms performs best we only use this classifier in the next experiments. Additionally, for each task, we only use the features selected by the CFS algorithm.
The results of this experiment are presented in Table VII. In general, we can observe that both approaches achieve very similar results, which means that there is no clear winner. For both approaches, we observe the best prediction performance for layout tables and slightly lower measures for vertical and horizontal listings. Also, both approach show a weak performance for matrix tables. In general, the results do not confirm the initial assumption that a double-layer approach achieves more accurate results. Comparing the results to the performance measures achieved by layout identification separately, we can see that errors from the table detection task propagate through the classification process and impact the precision in the layout identification step.
Although the single-layer and double-layer approaches achieve a very similar prediction quality overall, if we look at the individual measures in detail, we can identify some differences in their respective performances. While the single-layer approach achieves a higher *precision* for genuine tables, the double-layer approach achieves a higher *recall* for most of the classes. Depending on the target application of the extracted tables, we may favor one over the other. The authors in [22], for instance, explicitly favor recall for the extraction

Table V
EVALUATION OF COMBINED CLASSIFICATION PROBLEM. ALL MEASURES ARE REPORTED FOR THE FOLLOWING CLASSES: LAYOUT (L), VERTICAL LISTINGS (V), HORIZONTAL LISTINGS (H), MATRIX (M) AND OTHER (O).

Metric	L	V	H	M	O	Weight. Avg.
Precision	93.12	67.51	**84.78**	35.83	**76.39**	**85.13**
Recall	**97.34**	67.28	71.03	14.65	48.50	**82.06**
F1	**95.17**	67.09	**77.09**	19.31	58.40	**82.95**

Table VI
WEIGHTED F_1-MEASURE ACHIEVED USING ALL FEATURES COMPARED TO FEATURES SELECTED BY CFS ALGORITHM.

Classifier	Table Detection		Layout Identification		Combined	
	All	CFS	All	CFS	All	CFS
J48	89.41	89.22	77.54	78.46	80.67	80.69
SimpleCART	89.12	89.56	79.43	77.92	80.30	80.16
Random Forest	**90.58**	90.52	79.35	**80.56**	**82.95**	82.93
SMO (Poly)	88.80	89.08	75.86	73.42	81.28	80.79
SMO (RBF)	90.02	88.63	77.43	72.86	81.88	79.44

Table VII
COMPARISON OF THE SINGLE-LAYER AND DOUBLE-LAYER APPROACHES, USING RANDOM FOREST AS THE CLASSIFICATION ALGORITHM AND THE FEATURES SELECTED BY CFS.

Approach	Metric	L	V	H	M	O	Weight. Avg.
Single-layer	Precision	93.57	**66.27**	**81.32**	**39.91**	**76.65**	**84.45**
	Recall	**96.60**	69.60	71.95	**17.83**	50.83	82.44
	F1	95.06	67.50	76.10	**22.67**	**60.14**	82.93
Double-layer	Precision	**95.29**	64.35	78.51	26.63	68.15	83.72
	Recall	96.16	**73.26**	**73.93**	15.86	**53.38**	**83.35**
	F1	**95.72**	**68.52**	**76.15**	19.88	59.87	**83.38**

of genuine tables, as the precision can be further improved in subsequent processing steps.
In addition, there are further characteristics that need to be taken into account when selecting one of the classification approaches. On the one hand, a single-layer approach is less computationally expensive, as it involves only a single classification step. On the other hand, the double-layer approach is more flexible, providing opportunities for the training data, feature and classifier selection to be adjusted to the task at hand.

VI. CONCLUSION & BUILDING DWTC

Our comprehensive experiments show that layout types observed in Web tables are not equally frequent on the Web. Especially matrix tables are relatively rare. Consequently, a large sample size is required, when the training data is randomly sampled from the Web, in order to include a sufficient number of matrix tables to build the classification model. Additionally, layout tables represent the vast majority of tables on the Web. Thus, the accuracy with which these tables are identified and filtered, has a significant impact on the precision of detecting genuine tables with different layout types.
Overall, we achieve good results for the detection of genuine Web tables and the identification of their main layout types. As a result, the double-layer classification approach we proposed, has been employed to classify millions of tables for the Dresden Web Table Corpus. For processing the Common Crawl corpora on Amazon S3 we used 100 Amazon EC2 c3.2xlarge machines with 8 vCPUs and 15 GB main memory each. The total compute time for the whole corpus was 10.238 hours which took more than 4 days using the 100 EC2 instances. From a monetary point of view, extracting the Web table corpus from the July 2014 Common Crawl required a total machine rental fee of \$102.38 using Amazon spot instances. The DWTC is provided by 500 GZip compressed text files[3], each line of text containing one JSON document representing one extracted table and its metadata. For more statistics on our Web table corpus as well as additional details on the extractor we refer the reader to [7].

[3]http://wwwdb.inf.tu-dresden.de/misc/dwtc/data_feb15/dwtc-XXX.json.gz

References

[1] X. Yin, W. Tan, and C. Liu, "Facto: A fact lookup engine based on web tables," in *Proceedings of the 20th International Conference on World Wide Web*, 2011, pp. 507–516. [Online]. Available: http://doi.acm.org/10.1145/1963405.1963477

[2] J. Eberius, M. Thiele, K. Braunschweig, and W. Lehner, "Top-k entity augmentation using consistent set covering," in *Proceedings of the 27th International Conference on Scientific and Statistical Database Management*, ser. SSDBM '15. New York, NY, USA: ACM, 2015, pp. 8:1–8:12. [Online]. Available: http://doi.acm.org/10.1145/2791347.2791353

[3] ——, "Drillbeyond: Enabling business analysts to explore the web of open data," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1978–1981, 2012. [Online]. Available: http://dx.doi.org/10.14778/2367502.2367552

[4] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri, "Infogather: Entity augmentation and attribute discovery by holistic matching with web tables," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. ACM, 2012, pp. 97–108. [Online]. Available: http://doi.acm.org/10.1145/2213836.2213848

[5] V. Mulwad, T. Finin, and A. Joshi, "Generating linked data by inferring the semantics of tables," in *Proceedings of the 1st International Workshop on Searching and Integrating New Web Data Sources - Very Large Data Search*, 2011, pp. 17–22.

[6] R. Meusel, P. Petrovski, and C. Bizer, "The webdatacommons microdata, rdfa and microformat dataset series," in *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I*, 2014, pp. 277–292. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-11964-9_18

[7] "Dresden Web Table Corpus," https://wwwdb.inf.tu-dresden.de/misc/dwtc/, accessed: 2015-08-06.

[8] H.-H. Chen, S.-C. Tsai, and J.-H. Tsai, "Mining tables from large scale html texts," in *Proceedings of the 18th Conference on Computational Linguistics*, vol. 1. ACL, 2000, pp. 166–172. [Online]. Available: http://dx.doi.org/10.3115/990820.990845

[9] G. Penn, J. Hu, H. Luo, and R. T. McDonald, "Flexible web document analysis for delivery to narrow-bandwidth devices," in *Proceedings of the 6th International Conference on Document Analysis and Recognition*, 2001, pp. 1074–1078.

[10] Y. Wang and J. Hu, "A machine learning based approach for table detection on the web," in *Proceedings of the 11th International Conference on World Wide Web*, 2002, pp. 242–250. [Online]. Available: http://doi.acm.org/10.1145/511446.511478

[11] M. J. Cafarella, A. Y. Halevy, Y. Zhang, D. Z. Wang, and E. Wu, "Uncovering the relational web," in *Proceedings of the 11th International Workshop on the Web and Databases: In Conjunction with the 2008 ACM SIGMOD International Conference on Management of Data*, 2008.

[12] J.-W. Son and S.-B. Park, "Web table discrimination with composition of rich structural and content information," *Applied Soft Computing*, vol. 13, no. 1, pp. 47–57, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.asoc.2012.07.025

[13] E. Crestan and P. Pantel, "Web-scale table census and classification," in *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*. ACM, 2011, pp. 545–554. [Online]. Available: http://doi.acm.org/10.1145/1935826.1935904

[14] L. R. Lautert, M. M. Scheidt, and C. F. Dorneles, "Web table taxonomy and formalization," *SIGMOD Record*, vol. 42, no. 3, pp. 28–33, 2013. [Online]. Available: http://doi.acm.org/10.1145/2536669.2536674

[15] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, The University of Waikato, 1999.

[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[17] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Wadsworth, 1984.

[18] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

[19] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1010933404324

[20] V. Vapnik, *Estimation of Dependences Based on Empirical Data*, ser. Springer Series in Statistics. Springer-Verlag New York, Inc., 1982.

[21] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=68391

[22] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "Webtables: Exploring the power of tables on the web," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 538–549, 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1453856.1453916